Package 'nlme'

October 31, 2026

Version 3.1-172

Date 2026-10-31

Priority recommended

Title Linear and Nonlinear Mixed Effects Models

Contact see 'MailingList'

Description Fit and compare Gaussian linear and nonlinear mixed-effects models.

Depends R (>= 3.6.0)

Imports graphics, stats, utils, lattice

Suggests MASS

LazyData yes

Encoding UTF-8

License GPL (>= 2)

BugReports https://bugs.r-project.org

MailingList R-help@r-project.org

URL https://svn.r-project.org/R-packages/trunk/nlme/

NeedsCompilation yes

Author José Pinheiro [aut] (S version),
Douglas Bates [aut] (up to 2007),
Saikat DebRoy [ctb] (up to 2002),
Deepayan Sarkar [ctb] (up to 2005),
EISPACK authors [ctb] (src/rs.f),
Siem Heisterkamp [ctb] (Author fixed sigma),
Bert Van Willigen [ctb] (Programmer fixed sigma),
Johannes Ranke [ctb] (varConstProp()),
R Core Team [aut, cre] (ROR: <https://ror.org/02zz1nj61>)

Maintainer R Core Team <R-core@R-project.org>

Repository CRAN

Date/Publication 2026-10-31 09:07:01 UTC

Contents

ACF . . . 8
ACF.gls . . . 9
ACF.lme . . . 10
Alfalfa . . . 11
allCoef . . . 12
anova.gls . . . 13
anova.lme . . . 15
as.matrix.corStruct . . . 18
as.matrix.pdMat . . . 19
as.matrix.reStruct . . . 20
asOneFormula . . . 21
Assay . . . 21
asTable . . . 22
augPred . . . 23
balancedGrouped . . . 25
bdf . . . 26
BodyWeight . . . 27
Cefamandole . . . 28
Coef . . . 29
coef.corStruct . . . 30
coef.gnls . . . 31
coef.lme . . . 32
coef.lmList . . . 33
coef.modelStruct . . . 35
coef.pdMat . . . 36
coef.reStruct . . . 37
coef.varFunc . . . 38
collapse . . . 39
collapse.groupedData . . . 40
compareFits . . . 42
comparePred . . . 43
corAR1 . . . 44
corARMA . . . 46
corCAR1 . . . 47
corClasses . . . 49
corCompSymm . . . 50
corExp . . . 51
corFactor . . . 53
corFactor.corStruct . . . 54
corGaus . . . 55
corLin . . . 57
corMatrix . . . 58
corMatrix.corStruct . . . 59
corMatrix.pdMat . . . 61
corMatrix.reStruct . . . 62
corNatural . . . 63

corRatio . . . 64
corSpatial . . . 66
corSpher . . . 67
corSymm . . . 69
Covariate . . . 71
Covariate.varFunc . . . 72
Dialyzer . . . 73
Dim . . . 74
Dim.corSpatial . . . 74
Dim.corStruct . . . 75
Dim.pdMat . . . 77
Earthquake . . . 77
ergoStool . . . 78
Fatigue . . . 79
fdHess . . . 80
fitted.glsStruct . . . 81
fitted.gnlsStruct . . . 82
fitted.lme . . . 83
fitted.lmeStruct . . . 84
fitted.lmList . . . 85
fitted.nlmeStruct . . . 86
fixed.effects . . . 87
fixef.lmList . . . 88
formula.pdBlocked . . . 89
formula.pdMat . . . 90
formula.reStruct . . . 91
gapply . . . 92
Gasoline . . . 93
getCovariate . . . 94
getCovariate.corStruct . . . 95
getCovariate.data.frame . . . 96
getCovariate.varFunc . . . 97
getCovariateFormula . . . 98
getData . . . 98
getData.gls . . . 99
getData.lme . . . 100
getData.lmList . . . 101
getGroups . . . 102
getGroups.corStruct . . . 103
getGroups.data.frame . . . 104
getGroups.gls . . . 105
getGroups.lme . . . 106
getGroups.lmList . . . 107
getGroups.varFunc . . . 108
getGroupsFormula . . . 109
getResponse . . . 110
getResponseFormula . . . 110
getVarCov . . . 111

gls . . . 112
glsControl . . . 114
glsObject . . . 116
glsStruct . . . 117
Glucose . . . 118
Glucose2 . . . 118
gnls . . . 119
gnlsControl . . . 121
gnlsObject . . . 122
gnlsStruct . . . 124
groupedData . . . 125
gsummary . . . 127
Gun . . . 129
IGF . . . 129
Initialize . . . 130
Initialize.corStruct . . . 131
Initialize.glsStruct . . . 132
Initialize.lmeStruct . . . 132
Initialize.reStruct . . . 133
Initialize.varFunc . . . 134
intervals . . . 135
intervals.gls . . . 136
intervals.lme . . . 137
intervals.lmList . . . 139
isBalanced . . . 140
isInitialized . . . 141
LDEsysMat . . . 142
lme . . . 143
lme.groupedData . . . 146
lme.lmList . . . 148
lmeControl . . . 150
lmeObject . . . 152
lmeStruct . . . 154
lmList . . . 155
lmList.groupedData . . . 156
logDet . . . 157
logDet.corStruct . . . 158
logDet.pdMat . . . 159
logDet.reStruct . . . 160
logLik.corStruct . . . 161
logLik.glsStruct . . . 162
logLik.gnls . . . 163
logLik.gnlsStruct . . . 164
logLik.lme . . . 165
logLik.lmeStruct . . . 166
logLik.lmList . . . 167
logLik.reStruct . . . 168
logLik.varFunc . . . 169

Machines . . . 170
MathAchieve . . . 170
MathAchSchool . . . 171
Matrix . . . 171
Matrix.pdMat . . . 172
Matrix.reStruct . . . 173
Meat . . . 174
Milk . . . 174
model.matrix.reStruct . . . 175
Muscle . . . 176
Names . . . 177
Names.formula . . . 178
Names.pdBlocked . . . 179
Names.pdMat . . . 180
Names.reStruct . . . 181
needUpdate . . . 182
needUpdate.modelStruct . . . 182
Nitrendipene . . . 183
nlme . . . 184
nlme.nlsList . . . 187
nlmeControl . . . 189
nlmeObject . . . 191
nlmeStruct . . . 193
nlsList . . . 194
nlsList.selfStart . . . 196
Oats . . . 197
Orthodont . . . 198
Ovary . . . 199
Oxboys . . . 199
Oxide . . . 200
pairs.compareFits . . . 201
pairs.lme . . . 202
pairs.lmList . . . 203
PBG . . . 205
pdBlocked . . . 205
pdClasses . . . 207
pdCompSymm . . . 208
pdConstruct . . . 209
pdConstruct.pdBlocked . . . 210
pdDiag . . . 212
pdFactor . . . 214
pdFactor.reStruct . . . 215
pdIdent . . . 216
pdLogChol . . . 217
pdMat . . . 219
pdMatrix . . . 220
pdMatrix.reStruct . . . 221
pdNatural . . . 222

pdSymm . . . 223
Phenobarb . . . 225
phenoModel . . . 226
Pixel . . . 227
plot.ACF . . . 227
plot.augPred . . . 228
plot.compareFits . . . 229
plot.gls . . . 230
plot.intervals.lmList . . . 232
plot.lme . . . 233
plot.lmList . . . 235
plot.nffGroupedData . . . 236
plot.nfnGroupedData . . . 238
plot.nmGroupedData . . . 240
plot.ranef.lme . . . 242
plot.ranef.lmList . . . 244
plot.Variogram . . . 245
pooledSD . . . 246
predict.gls . . . 247
predict.gnls . . . 248
predict.lme . . . 249
predict.lmList . . . 250
predict.nlme . . . 252
print.summary.pdMat . . . 253
print.varFunc . . . 254
qqnorm.gls . . . 255
qqnorm.lme . . . 256
Quinidine . . . 258
quinModel . . . 259
Rail . . . 260
random.effects . . . 261
ranef.lme . . . 261
ranef.lmList . . . 263
RatPupWeight . . . 265
recalc . . . 265
recalc.corStruct . . . 266
recalc.modelStruct . . . 267
recalc.reStruct . . . 268
recalc.varFunc . . . 269
Relaxin . . . 270
Remifentanil . . . 270
residuals.gls . . . 272
residuals.glsStruct . . . 273
residuals.gnlsStruct . . . 274
residuals.lme . . . 275
residuals.lmeStruct . . . 276
residuals.lmList . . . 277
residuals.nlmeStruct . . . 278

reStruct . . . 279
simulate.lme . . . 281
solve.pdMat . . . 282
solve.reStruct . . . 283
Soybean . . . 284
splitFormula . . . 284
Spruce . . . 285
summary.corStruct . . . 286
summary.gls . . . 287
summary.lme . . . 288
summary.lmList . . . 289
summary.modelStruct . . . 291
summary.nlsList . . . 292
summary.pdMat . . . 293
summary.varFunc . . . 294
Tetracycline1 . . . 295
Tetracycline2 . . . 296
update.modelStruct . . . 296
update.varFunc . . . 297
varClasses . . . 298
varComb . . . 299
varConstPower . . . 300
varConstProp . . . 301
VarCorr . . . 304
varExp . . . 305
varFixed . . . 306
varFunc . . . 307
varIdent . . . 308
Variogram . . . 309
Variogram.corExp . . . 310
Variogram.corGaus . . . 311
Variogram.corLin . . . 312
Variogram.corRatio . . . 313
Variogram.corSpatial . . . 314
Variogram.corSpher . . . 315
Variogram.default . . . 316
Variogram.gls . . . 317
Variogram.lme . . . 319
varPower . . . 322
varWeights . . . 323
varWeights.glsStruct . . . 324
varWeights.lmeStruct . . . 325
Wafer . . . 326
Wheat . . . 326
Wheat2 . . . 327
[.pdMat . . . 327

Index **329**

ACF *Autocorrelation Function*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: `gls` and `lme`.

Usage

```
ACF(object, maxLag, ...)
```

Arguments

`object`	any object from which an autocorrelation function can be obtained. Generally an object resulting from a model fit, from which residuals can be extracted.
`maxLag`	maximum lag for which the autocorrelation should be calculated.
`...`	some methods for this generic require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <Bates@stat.wisc.edu>

References

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`ACF.gls`, `ACF.lme`, `plot.ACF`

ACF.gls *Autocorrelation Function for gls Residuals*

Description

This method function calculates the empirical autocorrelation function for the residuals from a gls fit. If a grouping variable is specified in form, the autocorrelation values are calculated using pairs of residuals within the same group; otherwise all possible residual pairs are used. The autocorrelation function is useful for investigating serial correlation models for equally spaced data.

Usage

```
## S3 method for class 'gls'
ACF(object, maxLag, resType, form, na.action, ...)
```

Arguments

object	an object inheriting from class "gls", representing a generalized least squares fitted model.
maxLag	an optional integer giving the maximum lag for which the autocorrelation should be calculated. Defaults to maximum lag in the residuals.
resType	an optional character string specifying the type of residuals to be used. If "response", the "raw" residuals (observed - fitted) are used; else, if "pearson", the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if "normalized", the normalized residuals (standardized residuals pre-multiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to "pearson".
form	an optional one sided formula of the form ~ t, or ~ t | g, specifying a time covariate t and, optionally, a grouping factor g. The time covariate must be integer valued. When a grouping factor is present in form, the autocorrelations are calculated using residual pairs within the same group. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes ACF.gls to print an error message and terminate if there are any incomplete observations.
...	some methods for this generic require additional arguments.

Value

a data frame with columns lag and ACF representing, respectively, the lag between residuals within a pair and the corresponding empirical autocorrelation. The returned value inherits from class ACF.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

ACF.lme, plot.ACF

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary)
ACF(fm1, form = ~ 1 | Mare)

# Pinheiro and Bates, p. 255-257
fm1Dial.gls <- gls(rate ~
  (pressure+I(pressure^2)+I(pressure^3)+I(pressure^4))*QB,
                  Dialyzer)

fm2Dial.gls <- update(fm1Dial.gls,
                 weights = varPower(form = ~ pressure))

ACF(fm2Dial.gls, form = ~ 1 | Subject)
```

ACF.lme *Autocorrelation Function for lme Residuals*

Description

This method function calculates the empirical autocorrelation function for the within-group residuals from an lme fit. The autocorrelation values are calculated using pairs of residuals within the innermost group level. The autocorrelation function is useful for investigating serial correlation models for equally spaced data.

Usage

```
## S3 method for class 'lme'
ACF(object, maxLag, resType, ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model.
maxLag	an optional integer giving the maximum lag for which the autocorrelation should be calculated. Defaults to maximum lag in the within-group residuals.

resType — an optional character string specifying the type of residuals to be used. If "response", the "raw" residuals (observed - fitted) are used; else, if "pearson", the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if "normalized", the normalized residuals (standardized residuals pre-multiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to "pearson".

... — some methods for this generic require additional arguments – not used.

Value

a data frame with columns lag and ACF representing, respectively, the lag between residuals within a pair and the corresponding empirical autocorrelation. The returned value inherits from class ACF.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

ACF.gls, plot.ACF

Examples

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time),
           Ovary, random = ~ sin(2*pi*Time) | Mare)
ACF(fm1, maxLag = 11)

# Pinheiro and Bates, p240-241
fm1Over.lme <- lme(follicles  ~ sin(2*pi*Time) +
           cos(2*pi*Time), data=Ovary,
     random=pdDiag(~sin(2*pi*Time)) )
(ACF.fm1Over <- ACF(fm1Over.lme, maxLag=10))
plot(ACF.fm1Over, alpha=0.01)
```

Alfalfa — *Split-Plot Experiment on Varieties of Alfalfa*

Description

The Alfalfa data frame has 72 rows and 4 columns.

Format

This data frame contains the following columns:

Variety a factor with levels `Cossack`, `Ladak`, and `Ranger`

Date a factor with levels `None` `S1` `S20` `O7`

Block a factor with levels `1` `2` `3` `4` `5` `6`

Yield a numeric vector

Details

These data are described in Snedecor and Cochran (1980) as an example of a split-plot design. The treatment structure used in the experiment was a $3x4$ full factorial, with three varieties of alfalfa and four dates of third cutting in 1943. The experimental units were arranged into six blocks, each subdivided into four plots. The varieties of alfalfa (*Cossac*, *Ladak*, and *Ranger*) were assigned randomly to the blocks and the dates of third cutting (*None*, *S1*—September 1, *S20*—September 20, and *O7*—October 7) were randomly assigned to the plots. All four dates were used on each block.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.1)

Snedecor, G. W. and Cochran, W. G. (1980), *Statistical Methods (7th ed)*, Iowa State University Press, Ames, IA

`allCoef` *Extract Coefficients from a Set of Objects*

Description

The extractor function is applied to each object in `...`, with the result being converted to a vector. A `map` attribute is included to indicate which pieces of the returned vector correspond to the original objects in `dots`.

Usage

```
allCoef(..., extract)
```

Arguments

`...`	objects to which `extract` will be applied. Generally these will be model components, such as `corStruct` and `varFunc` objects.
`extract`	an optional extractor function. Defaults to `coef`.

Value

a vector with all elements, generally coefficients, obtained by applying `extract` to the objects in `....`

Author(s)

José' Pinheiro and Douglas Bates

See Also

lmeStruct,nlmeStruct

Examples

```
cs1 <- corAR1(0.1)
vf1 <- varPower(0.5)
allCoef(cs1, vf1)
```

---

anova.gls *Compare Likelihoods of Fitted Objects*

Description

When only one fitted model object is present, a data frame with the numerator degrees of freedom, F-values, and P-values for Wald tests for the terms in the model (when Terms and L are NULL), a combination of model terms (when Terms in not NULL), or linear combinations of the model coefficients (when L is not NULL). Otherwise, when multiple fitted objects are being compared, a data frame with the degrees of freedom, the (restricted) log-likelihood, the Akaike Information Criterion (AIC), and the Bayesian Information Criterion (BIC) of each object is returned. If test=TRUE, whenever two consecutive objects have different number of degrees of freedom, a likelihood ratio statistic with the associated p-value is included in the returned data frame.

Usage

```
## S3 method for class 'gls'
anova(object, ..., test, type, adjustSigma, Terms, L, verbose)
```

Arguments

object	an object inheriting from class "gls", representing a generalized least squares fit.
...	other optional fitted model objects inheriting from classes "gls", "gnls", "lm", "lme", "lmList", "nlme", "nlsList", or "nls".
test	an optional logical value controlling whether likelihood ratio tests should be used to compare the fitted models represented by object and the objects in .... Defaults to TRUE.
type	an optional character string specifying the type of sum of squares to be used in F-tests for the terms in the model. If "sequential", the sequential sum of squares obtained by including the terms in the order they appear in the model is used; else, if "marginal", the marginal sum of squares obtained by deleting a term from the model at a time is used. This argument is only used when a single fitted object is passed to the function. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to "sequential".

`adjustSigma` an optional logical value. If `TRUE` and the estimation method used to obtain `object` was maximum likelihood, the residual standard error is multiplied by $\sqrt{n_{obs}/(n_{obs} - n_{par})}$, converting it to a REML-like estimate. This argument is only used when a single fitted object is passed to the function. Default is `TRUE`.

`Terms` an optional integer or character vector specifying which terms in the model should be jointly tested to be zero using a Wald F-test. If given as a character vector, its elements must correspond to term names; else, if given as an integer vector, its elements must correspond to the order in which terms are included in the model. This argument is only used when a single fitted object is passed to the function. Default is `NULL`.

`L` an optional numeric vector or array specifying linear combinations of the coefficients in the model that should be tested to be zero. If given as an array, its rows define the linear combinations to be tested. If names are assigned to the vector elements (array columns), they must correspond to coefficients names and will be used to map the linear combination(s) to the coefficients; else, if no names are available, the vector elements (array columns) are assumed in the same order as the coefficients appear in the model. This argument is only used when a single fitted object is passed to the function. Default is `NULL`.

`verbose` an optional logical value. If `TRUE`, the calling sequences for each fitted model object are printed with the rest of the output, being omitted if `verbose = FALSE`. Defaults to `FALSE`.

Value

a data frame inheriting from class `"anova.lme"`.

Note

Likelihood comparisons are not meaningful for objects fit using restricted maximum likelihood and with different fixed effects.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

See Also

`gls`, `gnls`, `nlme`, `lme`, `logLik.gls`, `AIC`, `BIC`, `print.anova.lme`

Examples

```
# AR(1) errors within each Mare
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
```

```
anova(fm1)
# variance changes with a power of the absolute fitted values?
fm2 <- update(fm1, weights = varPower())
anova(fm1, fm2)

# Pinheiro and Bates, p. 251-252
fm1Orth.gls <- gls(distance ~ Sex * I(age - 11), Orthodont,
                correlation = corSymm(form = ~ 1 | Subject),
                weights = varIdent(form = ~ 1 | age))
fm2Orth.gls <- update(fm1Orth.gls,
                corr = corCompSymm(form = ~ 1 | Subject))
anova(fm1Orth.gls, fm2Orth.gls)

# Pinheiro and Bates, pp. 215-215, 255-260
#p. 215
fm1Dial.lme <-
  lme(rate ~(pressure + I(pressure^2) + I(pressure^3) + I(pressure^4))*QB,
      Dialyzer, ~ pressure + I(pressure^2))
# p. 216
fm2Dial.lme <- update(fm1Dial.lme,
                  weights = varPower(form = ~ pressure))
# p. 255
fm1Dial.gls <- gls(rate ~ (pressure +
     I(pressure^2) + I(pressure^3) + I(pressure^4))*QB,
        Dialyzer)
fm2Dial.gls <- update(fm1Dial.gls,
                 weights = varPower(form = ~ pressure))
anova(fm1Dial.gls, fm2Dial.gls)
fm3Dial.gls <- update(fm2Dial.gls,
                    corr = corAR1(0.771, form = ~ 1 | Subject))
anova(fm2Dial.gls, fm3Dial.gls)
# anova.gls to compare a gls and an lme fit
anova(fm3Dial.gls, fm2Dial.lme, test = FALSE)

# Pinheiro and Bates, pp. 261-266
fm1Wheat2 <- gls(yield ~ variety - 1, Wheat2)
fm3Wheat2 <- update(fm1Wheat2,
      corr = corRatio(c(12.5, 0.2),
        form = ~ latitude + longitude, nugget = TRUE))
# Test a specific contrast
anova(fm3Wheat2, L = c(-1, 0, 1))
```

---

anova.lme *Compare Likelihoods of Fitted Objects*

Description

When only one fitted model object is present, a data frame with the numerator degrees of freedom, denominator degrees of freedom, F-values, and P-values for Wald tests for the terms in the model

(when Terms and L are NULL), a combination of model terms (when Terms in not NULL), or linear combinations of the model coefficients (when L is not NULL). Otherwise, when multiple fitted objects are being compared, a data frame with the degrees of freedom, the (restricted) log-likelihood, the Akaike Information Criterion (AIC), and the Bayesian Information Criterion (BIC) of each object is returned. If test=TRUE, whenever two consecutive objects have different number of degrees of freedom, a likelihood ratio statistic with the associated p-value is included in the returned data frame.

Usage

```
## S3 method for class 'lme'
anova(object, ..., test, type, adjustSigma, Terms, L, verbose)
## S3 method for class 'anova.lme'
print(x, verbose, ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model.
...	other optional fitted model objects inheriting from classes "gls", "gnls", "lm", "lme", "lmList", "nlme", "nlsList", or "nls".
test	an optional logical value controlling whether likelihood ratio tests should be used to compare the fitted models represented by object and the objects in .... Defaults to TRUE.
type	an optional character string specifying the type of sum of squares to be used in F-tests for the terms in the model. If "sequential", the sequential sum of squares obtained by including the terms in the order they appear in the model is used; else, if "marginal", the marginal sum of squares obtained by deleting a term from the model at a time is used. This argument is only used when a single fitted object is passed to the function. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to "sequential".
adjustSigma	an optional logical value. If TRUE and the estimation method used to obtain object was maximum likelihood, the residual standard error is multiplied by $\sqrt{n_{obs}/(n_{obs} - n_{par})}$, converting it to a REML-like estimate. This argument is only used when a single fitted object is passed to the function. Default is TRUE.
Terms	an optional integer or character vector specifying which terms in the model should be jointly tested to be zero using a Wald F-test. If given as a character vector, its elements must correspond to term names; else, if given as an integer vector, its elements must correspond to the order in which terms are included in the model. This argument is only used when a single fitted object is passed to the function. Default is NULL.
L	an optional numeric vector or array specifying linear combinations of the coefficients in the model that should be tested to be zero. If given as an array, its rows define the linear combinations to be tested. If names are assigned to the vector elements (array columns), they must correspond to coefficients names and will be used to map the linear combination(s) to the coefficients; else, if no names are available, the vector elements (array columns) are assumed in the same order as

	the coefficients appear in the model. This argument is only used when a single fitted object is passed to the function. Default is NULL.
x	an object inheriting from class "anova.lme"
verbose	an optional logical value. If TRUE, the calling sequences for each fitted model object are printed with the rest of the output, being omitted if verbose = FALSE. Defaults to FALSE.

Value

a data frame inheriting from class "anova.lme".

Note

Likelihood comparisons are not meaningful for objects fit using restricted maximum likelihood and with different fixed effects.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

gls, gnls, nlme, lme, AIC, BIC, print.anova.lme, logLik.lme,

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
anova(fm1)
fm2 <- update(fm1, random = pdDiag(~age))
anova(fm1, fm2)

## Pinheiro and Bates, pp. 251-254 ----------------------------------------
fm1Orth.gls <- gls(distance ~ Sex * I(age - 11), Orthodont,
     correlation = corSymm(form = ~ 1 | Subject),
     weights = varIdent(form = ~ 1 | age))
fm2Orth.gls <- update(fm1Orth.gls,
        corr = corCompSymm(form = ~ 1 | Subject))
## anova.gls examples:
anova(fm1Orth.gls, fm2Orth.gls)
fm3Orth.gls <- update(fm2Orth.gls, weights = NULL)
anova(fm2Orth.gls, fm3Orth.gls)
fm4Orth.gls <- update(fm3Orth.gls, weights = varIdent(form = ~ 1 | Sex))
anova(fm3Orth.gls, fm4Orth.gls)
# not in book but needed for the following command
fm3Orth.lme <- lme(distance ~ Sex*I(age-11), data = Orthodont,
                   random = ~ I(age-11) | Subject,
                   weights = varIdent(form = ~ 1 | Sex))
```

```
# Compare an "lme" object with a "gls" object (test would be non-sensical!)
anova(fm3Orth.lme, fm4Orth.gls, test = FALSE)

## Pinheiro and Bates, pp. 222-225 ----------------------------------------
op <- options(contrasts = c("contr.treatment", "contr.poly"))
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight, random = ~ Time)
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# Test a specific contrast
anova(fm2BW.lme, L = c("Time:Diet2" = 1, "Time:Diet3" = -1))

## Pinheiro and Bates, pp. 352-365 ----------------------------------------
fm1Theo.lis <- nlsList(
     conc ~ SSfol(Dose, Time, lKe, lKa, lCl), data=Theoph)
fm1Theo.lis
fm1Theo.nlme <- nlme(fm1Theo.lis)
fm2Theo.nlme <- update(fm1Theo.nlme, random= pdDiag(lKe+lKa+lCl~1) )
fm3Theo.nlme <- update(fm2Theo.nlme, random= pdDiag(    lKa+lCl~1) )

# Comparing the 3 nlme models
anova(fm1Theo.nlme, fm3Theo.nlme, fm2Theo.nlme)

options(op) # (set back to previous state)
```

`as.matrix.corStruct` *Matrix of a corStruct Object*

Description

This method function extracts the correlation matrix, or list of correlation matrices, associated with `object`.

Usage

```
## S3 method for class 'corStruct'
as.matrix(x, ...)
```

Arguments

`x`	an object inheriting from class "`corStruct`", representing a correlation structure.
`...`	further arguments passed from other methods.

Value

If the correlation structure includes a grouping factor, the returned value will be a list with components given by the correlation matrices for each group. Otherwise, the returned value will be a matrix representing the correlation structure associated with `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

See Also

corClasses, corMatrix

Examples

```
cst1 <- corAR1(form = ~1|Subject)
cst1 <- Initialize(cst1, data = Orthodont)
as.matrix(cst1)
```

---

as.matrix.pdMat *Matrix of a pdMat Object*

Description

This method function extracts the positive-definite matrix represented by x.

Usage

```
## S3 method for class 'pdMat'
as.matrix(x, ...)
```

Arguments

x	an object inheriting from class "pdMat", representing a positive-definite matrix.
...	further arguments passed from other methods.

Value

a matrix corresponding to the positive-definite matrix represented by x.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

See Also

pdMat, corMatrix

Examples

```
as.matrix(pdSymm(diag(4)))
```

`as.matrix.reStruct` *Matrices of an reStruct Object*

Description

This method function extracts the positive-definite matrices corresponding to the `pdMat` elements of `object`.

Usage

```
## S3 method for class 'reStruct'
as.matrix(x, ...)
```

Arguments

`x`	an object inheriting from class `"reStruct"`, representing a random effects structure and consisting of a list of `pdMat` objects.
`...`	further arguments passed from other methods.

Value

a list with components given by the positive-definite matrices corresponding to the elements of `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

See Also

as.matrix.pdMat, reStruct, pdMat

Examples

```
rs1 <- reStruct(pdSymm(diag(3), ~age+Sex, data = Orthodont))
as.matrix(rs1)
```

asOneFormula *Combine Formulas of a Set of Objects*

Description

The names of all variables used in the formulas extracted from the objects defined in ... are converted into a single linear formula, with the variables names separated by +.

Usage

```
asOneFormula(..., omit)
```

Arguments

...	objects, or lists of objects, from which a formula can be extracted.
omit	an optional character vector with the names of variables to be omitted from the returned formula. Defaults to c(".", "pi").

Value

a one-sided linear formula with all variables named in the formulas extracted from the objects in ..., except the ones listed in omit.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

formula, all.vars

Examples

```
asOneFormula(y ~ x + z | g, list(~ w, ~ t * sin(2 * pi)))
```

Assay *Bioassay on Cell Culture Plate*

Description

The Assay data frame has 60 rows and 4 columns.

Format

This data frame contains the following columns:

Block an ordered factor with levels 2 < 1 identifying the block where the wells are measured.

sample a factor with levels a to f identifying the sample corresponding to the well.

dilut a factor with levels 1 to 5 indicating the dilution applied to the well

logDens a numeric vector of the log-optical density

Details

These data, courtesy of Rich Wolfe and David Lansky from Searle, Inc., come from a bioassay run on a 96-well cell culture plate. The assay is performed using a split-block design. The 8 rows on the plate are labeled A–H from top to bottom and the 12 columns on the plate are labeled 1–12 from left to right. Only the central 60 wells of the plate are used for the bioassay (the intersection of rows B–G and columns 2–11). There are two blocks in the design: Block 1 contains columns 2–6 and Block 2 contains columns 7–11. Within each block, six samples are assigned randomly to rows and five (serial) dilutions are assigned randomly to columns. The response variable is the logarithm of the optical density. The cells are treated with a compound that they metabolize to produce the stain. Only live cells can make the stain, so the optical density is a measure of the number of cells that are alive and healthy.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.2)

---

asTable *Convert groupedData to a matrix*

Description

Create a tabular representation of the response in a balanced groupedData object.

Usage

```
asTable(object)
```

Arguments

object	A balanced groupedData object

Details

A balanced groupedData object can be represented as a matrix or table of response values corresponding to the values of a primary covariate for each level of a grouping factor. This function creates such a matrix representation of the data in object.

Value

A matrix. The data in the matrix are the values of the response. The columns correspond to the distinct values of the primary covariate and are labelled as such. The rows correspond to the distinct levels of the grouping factor and are labelled as such.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

See Also

groupedData, isBalanced, balancedGrouped

Examples

```
asTable(Orthodont)

# Pinheiro and Bates, p. 109
ergoStool.mat <- asTable(ergoStool)
```

---

augPred *Augmented Predictions*

Description

Predicted values are obtained at the specified values of `primary`. If `object` has a grouping structure (i.e. `getGroups(object)` is not `NULL`), predicted values are obtained for each group. If `level` has more than one element, predictions are obtained for each level of the `max(level)` grouping factor. If other covariates besides `primary` are used in the prediction model, their average (numeric covariates) or most frequent value (categorical covariates) are used to obtain the predicted values. The original observations are also included in the returned object.

Usage

```
augPred(object, primary, minimum, maximum, length.out, ...)
## S3 method for class 'lme'
augPred(object, primary = NULL,
        minimum = min(primary), maximum = max(primary),
        length.out = 51, level = Q, ...)
```

Arguments

object	a fitted model object from which predictions can be extracted, using a predict method.
primary	an optional one-sided formula specifying the primary covariate to be used to generate the augmented predictions. By default, if a covariate can be extracted from the data used to generate object (using getCovariate), it will be used as primary.
minimum	an optional lower limit for the primary covariate. Defaults to min(primary).
maximum	an optional upper limit for the primary covariate. Defaults to max(primary).
length.out	an optional integer with the number of primary covariate values at which to evaluate the predictions. Defaults to 51.
level	an optional integer vector specifying the desired prediction levels. Levels increase from outermost to innermost grouping, with level 0 representing the population (fixed effects) predictions. Defaults to the innermost level.
...	some methods for the generic may require additional arguments.

Value

a data frame with four columns representing, respectively, the values of the primary covariate, the groups (if object does not have a grouping structure, all elements will be 1), the predicted or observed values, and the type of value in the third column: original for the observed values and predicted (single or no grouping factor) or predict.groupVar (multiple levels of grouping), with groupVar replaced by the actual grouping variable name (fixed is used for population predictions). The returned object inherits from class "augPred".

Note

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: gls, lme, and lmList.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

See Also

plot.augPred, getGroups, predict

Examples

```
fm1 <- lme(Orthodont, random = ~1)
augPred(fm1, length.out = 2, level = c(0,1))
```

balancedGrouped *Create a groupedData object from a matrix*

Description

Create a groupedData object from a data matrix. This function can be used only with balanced data. The opposite conversion, from a groupedData object to a matrix, is done with asTable.

Usage

```
balancedGrouped(form, data, labels=NULL, units=NULL)
```

Arguments

form	A formula of the form y ~ x | g giving the name of the response, the primary covariate, and the grouping factor.
data	A matrix or data frame containing the values of the response grouped according to the levels of the grouping factor (rows) and the distinct levels of the primary covariate (columns). The dimnames of the matrix are used to construct the levels of the grouping factor and the primary covariate.
labels	an optional list of character strings giving labels for the response and the primary covariate. The label for the primary covariate is named x and that for the response is named y. Either label can be omitted.
units	an optional list of character strings giving the units for the response and the primary covariate. The units string for the primary covariate is named x and that for the response is named y. Either units string can be omitted.

Value

A balanced groupedData object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

See Also

groupedData, isBalanced, asTable

Examples

```
OrthoMat <- asTable( Orthodont )
Orth2 <- balancedGrouped(distance ~ age | Subject, data = OrthoMat,
    labels = list(x = "Age",
                  y = "Distance from pituitary to pterygomaxillary fissure"),
    units = list(x = "(yr)", y = "(mm)"))
Orth2[ 1:10, ]        ## check the first few entries

# Pinheiro and Bates, p. 109
ergoStool.mat <- asTable(ergoStool)
balancedGrouped(effort~Type|Subject,
                data=ergoStool.mat)
```

---

bdf *Language scores*

Description

The bdf data frame has 2287 rows and 25 columns of language scores from grade 8 pupils in elementary schools in The Netherlands.

Usage

```
bdf
```

Format

schoolNR a factor denoting the school.

pupilNR a factor denoting the pupil.

IQ.verb a numeric vector of verbal IQ scores

IQ.perf a numeric vector of IQ scores.

sex Sex of the student.

Minority a factor indicating if the student is a member of a minority group.

repeatgr an ordered factor indicating if one or more grades have been repeated.

aritPRET a numeric vector

classNR a numeric vector

aritPOST a numeric vector

langPRET a numeric vector

langPOST a numeric vector

ses a numeric vector of socioeconomic status indicators.

denomina a factor indicating of the school is a public school, a Protestant private school, a Catholic private school, or a non-denominational private school.

schoolSES a numeric vector

satiprin a numeric vector

natitest a factor with levels `0` and `1`

meetings a numeric vector

currmeet a numeric vector

mixedgra a factor indicating if the class is a mixed-grade class.

percmino a numeric vector

aritdiff a numeric vector

homework a numeric vector

classsiz a numeric vector

groupsiz a numeric vector

Source

'`http://stat.gamma.rug.nl/snijders/multilevel.htm`', the first edition of http://www.stats.ox.ac.uk/~snijders/mlbook.htm.

References

Snijders, Tom and Bosker, Roel (1999), *Multilevel Analysis: An Introduction to Basic and Advanced Multilevel Modeling*, Sage.

Examples

```
summary(bdf)

## More examples, including lme() fits  reproducing parts in the above
## book, are available in the R script files
system.file("mlbook", "ch04.R", package ="nlme") # and
system.file("mlbook", "ch05.R", package ="nlme")
```

BodyWeight *Rat weight over time for different diets*

Description

The `BodyWeight` data frame has 176 rows and 4 columns.

Format

This data frame contains the following columns:

weight a numeric vector giving the body weight of the rat (grams).

Time a numeric vector giving the time at which the measurement is made (days).

Rat an ordered factor with levels `2` < `3` < `4` < `1` < `8` < `5` < `6` < `7` < `11` < `9` < `10` < `12` < `13` < `15` < `14` < `16` identifying the rat whose weight is measured.

Diet a factor with levels `1` to `3` indicating the diet that the rat receives.

Details

Hand and Crowder (1996) describe data on the body weights of rats measured over 64 days. These data also appear in Table 2.4 of Crowder and Hand (1990). The body weights of the rats (in grams) are measured on day 1 and every seven days thereafter until day 64, with an extra measurement on day 44. The experiment started several weeks before "day 1." There are three groups of rats, each on a different diet.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.3)

Crowder, M. and Hand, D. (1990), *Analysis of Repeated Measures*, Chapman and Hall, London.

Hand, D. and Crowder, M. (1996), *Practical Longitudinal Data Analysis*, Chapman and Hall, London.

Cefamandole *Pharmacokinetics of Cefamandole*

Description

The `Cefamandole` data frame has 84 rows and 3 columns.

Format

This data frame contains the following columns:

Subject a factor giving the subject from which the sample was drawn.

Time a numeric vector giving the time at which the sample was drawn (minutes post-injection).

conc a numeric vector giving the observed plasma concentration of cefamandole (mcg/ml).

Details

Davidian and Giltinan (1995, 1.1, p. 2) describe data obtained during a pilot study to investigate the pharmacokinetics of the drug cefamandole. Plasma concentrations of the drug were measured on six healthy volunteers at 14 time points following an intraveneous dose of 15 mg/kg body weight of cefamandole.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.4)

Davidian, M. and Giltinan, D. M. (1995), *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, London.

Examples

```
plot(Cefamandole)
fm1 <- nlsList(SSbiexp, data = Cefamandole)
summary(fm1)
```

`Coef` *Assign Values to Coefficients*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include all "pdMat", "corStruct" and "varFunc" classes, "reStruct", and "modelStruct".

`coefficients<-` is an *alias* for `coef<-`.

Usage

```
coef(object, ...) <- value

coefficients(object, ...) <- value
```

Arguments

`object`	any object representing a fitted model, or, by default, any object with a `coef` component.
`...`	some methods for this generic function may require additional arguments.
`value`	a value to be assigned to the coefficients associated with `object`.

Value

will depend on the method function; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`coef`

`coef.corStruct` *Coefficients of a corStruct Object*

Description

This method function extracts the coefficients associated with the correlation structure represented by `object`.

Usage

```
## S3 method for class 'corStruct'
coef(object, unconstrained, ...)
## S3 replacement method for class 'corStruct'
coef(object, ...) <- value
```

Arguments

`object`	an object inheriting from class "`corStruct`", representing a correlation structure.
`unconstrained`	a logical value. If `TRUE` the coefficients are returned in unconstrained form (the same used in the optimization algorithm). If `FALSE` the coefficients are returned in "natural", possibly constrained, form. Defaults to `TRUE`.
`value`	a vector with the replacement values for the coefficients associated with `object`. It must be a vector with the same length of `coef{object}` and must be given in unconstrained form.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the coefficients corresponding to `object`.

SIDE EFFECTS

On the left side of an assignment, sets the values of the coefficients of `object` to `value`. `Object` must be initialized (using `Initialize`) before new values can be assigned to its coefficients.

Author(s)

José Pinheiro and Douglas Bates

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

See Also

corAR1, corARMA, corCAR1, corCompSymm, corExp, corGaus, corLin, corRatio, corSpatial, corSpher, corSymm,Initialize

Examples

```
cst1 <- corARMA(p = 1, q = 1)
coef(cst1)
```

---

coef.gnls *Extract gnls Coefficients*

Description

The estimated coefficients for the nonlinear model represented by `object` are extracted.

Usage

```
## S3 method for class 'gnls'
coef(object, ...)
```

Arguments

`object`	an object inheriting from class "gnls", representing a generalized nonlinear least squares fitted model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the estimated coefficients for the nonlinear model represented by `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gnls

Examples

```
fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            weights = varPower())
coef(fm1)
```

coef.lme *Extract lme Coefficients*

Description

The estimated coefficients at level i are obtained by adding together the fixed effects estimates and the corresponding random effects estimates at grouping levels less or equal to i. The resulting estimates are returned as a data frame, with rows corresponding to groups and columns to coefficients. Optionally, the returned data frame may be augmented with covariates summarized over groups.

Usage

```
## S3 method for class 'lme'
coef(object, augFrame, level, data, which, FUN,
       omitGroupingFactor, subset, ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model.
augFrame	an optional logical value. If TRUE, the returned data frame is augmented with variables defined in data; else, if FALSE, only the coefficients are returned. Defaults to FALSE.
level	an optional positive integer giving the level of grouping to be used in extracting the coefficients from an object with multiple nested grouping levels. Defaults to the highest or innermost level of grouping.
data	an optional data frame with the variables to be used for augmenting the returned data frame when augFrame = TRUE. Defaults to the data frame used to fit object.
which	an optional positive integer or character vector specifying which columns of data should be used in the augmentation of the returned data frame. Defaults to all columns in data.
FUN	an optional summary function or a list of summary functions to be applied to group-varying variables, when collapsing data by groups. Group-invariant variables are always summarized by the unique value that they assume within that group. If FUN is a single function it will be applied to each non-invariant variable by group to produce the summary for that variable. If FUN is a list of functions, the names in the list should designate classes of variables in the frame such as ordered, factor, or numeric. The indicated function will be applied to any group-varying variables of that class. The default functions to be used are mean for numeric factors, and Mode for both factor and ordered. The Mode function, defined internally in gsummary, returns the modal or most popular value of the variable. It is different from the mode function that returns the S-language mode of the variable.

omitGroupingFactor
: an optional logical value. When TRUE the grouping factor itself will be omitted from the group-wise summary of data but the levels of the grouping factor will continue to be used as the row names for the returned data frame. Defaults to FALSE.

subset
: an optional expression specifying a subset

...
: some methods for this generic require additional arguments. None are used in this method.

Value

a data frame inheriting from class "coef.lme" with the estimated coefficients at level level and, optionally, other covariates summarized over groups. The returned object also inherits from classes "ranef.lme" and "data.frame".

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York, esp. pp. 455-457.

See Also

lme, ranef.lme, plot.ranef.lme, gsummary

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
coef(fm1)
coef(fm1, augFrame = TRUE)
```

---

coef.lmList *Extract lmList Coefficients*

Description

The coefficients of each lm object in the object list are extracted and organized into a data frame, with rows corresponding to the lm components and columns corresponding to the coefficients. Optionally, the returned data frame may be augmented with covariates summarized over the groups associated with the lm components.

Usage

```
## S3 method for class 'lmList'
coef(object, augFrame, data, which, FUN,
   omitGroupingFactor, ...)
```

Arguments

object	an object inheriting from class "lmList", representing a list of lm objects with a common model.
augFrame	an optional logical value. If TRUE, the returned data frame is augmented with variables defined in the data frame used to produce object; else, if FALSE, only the coefficients are returned. Defaults to FALSE.
data	an optional data frame with the variables to be used for augmenting the returned data frame when augFrame = TRUE. Defaults to the data frame used to fit object.
which	an optional positive integer or character vector specifying which columns of the data frame used to produce object should be used in the augmentation of the returned data frame. Defaults to all variables in the data.
FUN	an optional summary function or a list of summary functions to be applied to group-varying variables, when collapsing the data by groups. Group-invariant variables are always summarized by the unique value that they assume within that group. If FUN is a single function it will be applied to each non-invariant variable by group to produce the summary for that variable. If FUN is a list of functions, the names in the list should designate classes of variables in the frame such as ordered, factor, or numeric. The indicated function will be applied to any group-varying variables of that class. The default functions to be used are mean for numeric factors, and Mode for both factor and ordered. The Mode function, defined internally in gsummary, returns the modal or most popular value of the variable. It is different from the mode function that returns the S-language mode of the variable.
omitGroupingFactor	an optional logical value. When TRUE the grouping factor itself will be omitted from the group-wise summary of data but the levels of the grouping factor will continue to be used as the row names for the returned data frame. Defaults to FALSE.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame inheriting from class "coef.lmList" with the estimated coefficients for each "lm" component of object and, optionally, other covariates summarized over the groups corresponding to the "lm" components. The returned object also inherits from classes "ranef.lmList" and "data.frame".

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York, esp. pp. 457-458.

See Also

lmList, fixed.effects.lmList, ranef.lmList, plot.ranef.lmList, gsummary

Examples

```
fm1 <- lmList(distance ~ age|Subject, data = Orthodont)
coef(fm1)
coef(fm1, augFrame = TRUE)
```

---

coef.modelStruct *Extract modelStruct Object Coefficients*

Description

This method function extracts the coefficients associated with each component of the modelStruct list.

Usage

```
## S3 method for class 'modelStruct'
coef(object, unconstrained, ...)
## S3 replacement method for class 'modelStruct'
coef(object, ...) <- value
```

Arguments

object	an object inheriting from class "modelStruct", representing a list of model components, such as "corStruct" and "varFunc" objects.
unconstrained	a logical value. If TRUE the coefficients are returned in unconstrained form (the same used in the optimization algorithm). If FALSE the coefficients are returned in "natural", possibly constrained, form. Defaults to TRUE.
value	a vector with the replacement values for the coefficients associated with object. It must be a vector with the same length of coef{object} and must be given in unconstrained form.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with all coefficients corresponding to the components of object.

SIDE EFFECTS

On the left side of an assignment, sets the values of the coefficients of object to value. Object must be initialized (using Initialize) before new values can be assigned to its coefficients.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Initialize

Examples

```
lms1 <- lmeStruct(reStruct = reStruct(pdDiag(diag(2), ~age)),
   corStruct = corAR1(0.3))
coef(lms1)
```

---

coef.pdMat *pdMat Object Coefficients*

Description

This method function extracts the coefficients associated with the positive-definite matrix represented by object.

Usage

```
## S3 method for class 'pdMat'
coef(object, unconstrained, ...)
## S3 replacement method for class 'pdMat'
coef(object, ...) <- value
```

Arguments

object	an object inheriting from class "pdMat", representing a positive-definite matrix.
unconstrained	a logical value. If TRUE the coefficients are returned in unconstrained form (the same used in the optimization algorithm). If FALSE the upper triangular elements of the positive-definite matrix represented by object are returned. Defaults to TRUE.
value	a vector with the replacement values for the coefficients associated with object. It must be a vector with the same length of coef{object} and must be given in unconstrained form.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the coefficients corresponding to object.

SIDE EFFECTS

On the left side of an assignment, sets the values of the coefficients of `object` to `value`.

Author(s)

José Pinheiro and Douglas Bates

References

Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289-296.

See Also

`pdMat`

Examples

```
coef(pdSymm(diag(3)))
```

---

coef.reStruct *reStruct Object Coefficients*

Description

This method function extracts the coefficients associated with the positive-definite matrix represented by `object`.

Usage

```
## S3 method for class 'reStruct'
coef(object, unconstrained, ...)
## S3 replacement method for class 'reStruct'
coef(object, ...) <- value
```

Arguments

`object`	an object inheriting from class `"reStruct"`, representing a random effects structure and consisting of a list of `pdMat` objects.
`unconstrained`	a logical value. If `TRUE` the coefficients are returned in unconstrained form (the same used in the optimization algorithm). If `FALSE` the coefficients are returned in "natural", possibly constrained, form. Defaults to `TRUE`.
`value`	a vector with the replacement values for the coefficients associated with `object`. It must be a vector with the same length of `coef(object)` and must be given in unconstrained form.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the coefficients corresponding to `object`.

SIDE EFFECTS

On the left side of an assignment, sets the values of the coefficients of `object` to `value`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`coef.pdMat`, `reStruct`, `pdMat`

Examples

```
rs1 <- reStruct(list(A = pdSymm(diag(1:3), form = ~Score),
  B = pdDiag(2 * diag(4), form = ~Educ)))
coef(rs1)
```

---

coef.varFunc *varFunc Object Coefficients*

Description

This method function extracts the coefficients associated with the variance function structure represented by `object`.

Usage

```
## S3 method for class 'varFunc'
coef(object, unconstrained, allCoef, ...)
## S3 replacement method for class 'varIdent'
coef(object, ...) <- value
```

Arguments

`object`	an object inheriting from class "`varFunc`" representing a variance function structure.
`unconstrained`	a logical value. If `TRUE` the coefficients are returned in unconstrained form (the same used in the optimization algorithm). If `FALSE` the coefficients are returned in "natural", generally constrained form. Defaults to `TRUE`.
`allCoef`	a logical value. If `FALSE` only the coefficients which may vary during the optimization are returned. If `TRUE` all coefficients are returned. Defaults to `FALSE`.

value	a vector with the replacement values for the coefficients associated with object. It must be have the same length of coef{object} and must be given in unconstrained form. Object must be initialized before new values can be assigned to its coefficients.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the coefficients corresponding to object.

SIDE EFFECTS

On the left side of an assignment, sets the values of the coefficients of object to value.

Author(s)

José Pinheiro and Douglas Bates

See Also

varFunc

Examples

```
vf1 <- varPower(1)
coef(vf1)
coef(vf1) <- 2
```

---

collapse *Collapse According to Groups*

Description

This function is generic; method functions can be written to handle specific classes of objects. Currently, only a groupedData method is available.

Usage

```
collapse(object, ...)
```

Arguments

object	an object to be collapsed, usually a data frame.
...	some methods for the generic may require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

collapse.groupedData

---

collapse.groupedData *Collapse a groupedData Object*

Description

If object has a single grouping factor, it is returned unchanged. Else, it is summarized by the values of the displayLevel grouping factor (or the combination of its values and the values of the covariate indicated in preserve, if any is present). The collapsed data is used to produce a new groupedData object, with grouping factor given by the displayLevel factor.

Usage

```
## S3 method for class 'groupedData'
collapse(object, collapseLevel, displayLevel,
       outer, inner, preserve, FUN, subset, ...)
```

Arguments

object	an object inheriting from class groupedData, generally with multiple grouping factors.
collapseLevel	an optional positive integer or character string indicating the grouping level to use when collapsing the data. Level values increase from outermost to innermost grouping. Default is the highest or innermost level of grouping.
displayLevel	an optional positive integer or character string indicating the grouping level to use as the grouping factor for the collapsed data. Default is collapseLevel.
outer	an optional logical value or one-sided formula, indicating covariates that are outer to the displayLevel grouping factor. If equal to TRUE, the displayLevel element attr(object, "outer") is used to indicate the outer covariates. An outer covariate is invariant within the sets of rows defined by the grouping factor. Ordering of the groups is done in such a way as to preserve adjacency of groups with the same value of the outer variables. Defaults to NULL, meaning that no outer covariates are to be used.
inner	an optional logical value or one-sided formula, indicating a covariate that is inner to the displayLevel grouping factor. If equal to TRUE, attr(object, "outer") is used to indicate the inner covariate. An inner covariate can change within the sets of rows defined by the grouping factor. Defaults to NULL, meaning that no inner covariate is present.

`preserve` an optional one-sided formula indicating a covariate whose levels should be preserved when collapsing the data according to the `collapseLevel` grouping factor. The collapsing factor is obtained by pasting together the levels of the `collapseLevel` grouping factor and the values of the covariate to be preserved. Default is `NULL`, meaning that no covariates need to be preserved.

`FUN` an optional summary function or a list of summary functions to be used for collapsing the data. The function or functions are applied only to variables in `object` that vary within the groups defined by `collapseLevel`. Invariant variables are always summarized by group using the unique value that they assume within that group. If `FUN` is a single function it will be applied to each non-invariant variable by group to produce the summary for that variable. If `FUN` is a list of functions, the names in the list should designate classes of variables in the data such as `ordered`, `factor`, or `numeric`. The indicated function will be applied to any non-invariant variables of that class. The default functions to be used are `mean` for numeric factors, and `Mode` for both `factor` and `ordered`. The `Mode` function, defined internally in `gsummary`, returns the modal or most popular value of the variable. It is different from the `mode` function that returns the S-language mode of the variable.

`subset` an optional named list. Names can be either positive integers representing grouping levels, or names of grouping factors. Each element in the list is a vector indicating the levels of the corresponding grouping factor to be preserved in the collapsed data. Default is `NULL`, meaning that all levels are used.

`...` some methods for this generic require additional arguments. None are used in this method.

Value

a `groupedData` object with a single grouping factor given by the `displayLevel` grouping factor, resulting from collapsing `object` over the levels of the `collapseLevel` grouping factor.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`groupedData`, `plot.nmGroupedData`

Examples

```
# collapsing by Dog
collapse(Pixel, collapseLevel = 1)
# same as collapse(Pixel, collapseLevel = "Dog")
```

compareFits *Compare Fitted Objects*

Description

The columns in object1 and object2 are put together in matrices which allow direct comparison of the individual elements for each object. Missing columns in either object are replaced by NAs.

Usage

```
compareFits(object1, object2, which)
```

Arguments

object1, object2	data frames, or matrices, with the same row names, but possibly different column names. These will usually correspond to coefficients from fitted objects with a grouping structure (e.g. lme and lmList objects).
which	an optional integer or character vector indicating which columns in object1 and object2 are to be used in the returned object. Defaults to all columns.

Value

a three-dimensional array, with the third dimension given by the number of unique column names in either object1 or object2. To each column name there corresponds a matrix with as many rows as the rows in object1 and two columns, corresponding to object1 and object2. The returned object inherits from class compareFits.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

plot.compareFits, pairs.compareFits, comparePred, coef, random.effects

Examples

```
fm1 <- lmList(Orthodont)
fm2 <- lme(fm1)
(cF12 <- compareFits(coef(fm1), coef(fm2)))
```

comparePred *Compare Predictions*

Description

Predicted values are obtained at the specified values of primary for each object. If either object1 or object2 have a grouping structure (i.e. getGroups(object) is not NULL), predicted values are obtained for each group. When both objects determine groups, the group levels must be the same. If other covariates besides primary are used in the prediction model, their group-wise averages (numeric covariates) or most frequent values (categorical covariates) are used to obtain the predicted values. The original observations are also included in the returned object.

Usage

```
comparePred(object1, object2, primary, minimum, maximum,
    length.out, level, ...)
```

Arguments

object1, object2	fitted model objects, from which predictions can be extracted using the predict method.
primary	an optional one-sided formula specifying the primary covariate to be used to generate the augmented predictions. By default, if a covariate can be extracted from the data used to generate the objects (using getCovariate), it will be used as primary.
minimum	an optional lower limit for the primary covariate. Defaults to min(primary), after primary is evaluated in the data used in fitting object1.
maximum	an optional upper limit for the primary covariate. Defaults to max(primary), after primary is evaluated in the data used in fitting object1.
length.out	an optional integer with the number of primary covariate values at which to evaluate the predictions. Defaults to 51.
level	an optional integer specifying the desired prediction level. Levels increase from outermost to innermost grouping, with level 0 representing the population (fixed effects) predictions. Only one level can be specified. Defaults to the innermost level.
...	some methods for the generic may require additional arguments.

Value

a data frame with four columns representing, respectively, the values of the primary covariate, the groups (if object does not have a grouping structure, all elements will be 1), the predicted or observed values, and the type of value in the third column: the objects' names are used to classify the predicted values and original is used for the observed values. The returned object inherits from classes comparePred and augPred.

Note

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: `gls`, `lme`, and `lmList`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

augPred, getGroups

Examples

```
fm1 <- lme(distance ~ age * Sex, data = Orthodont, random = ~ age)
fm2 <- update(fm1, distance ~ age)
comparePred(fm1, fm2, length.out = 2)
```

---

corAR1 *AR(1) Correlation Structure*

Description

This function is a constructor for the `corAR1` class, representing an autocorrelation structure of order 1. Objects created using this constructor must later be initialized using the appropriate `Initialize` method.

Usage

```
corAR1(value, form, fixed)
```

Arguments

`value`	the value of the lag 1 autocorrelation, which must be between -1 and 1. Defaults to 0 (no autocorrelation).
`form`	a one sided formula of the form `~ t`, or `~ t | g`, specifying a time covariate `t` and, optionally, a grouping factor `g`. A covariate for this correlation structure must be integer valued. When a grouping factor is present in `form`, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to `~ 1`, which corresponds to using the order of the observations in the data as a covariate, and no groups.
`fixed`	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to `FALSE`, in which case the coefficients are allowed to vary.

Value

an object of class corAR1, representing an autocorrelation structure of order 1.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 235, 397.

See Also

ACF.lme, corARMA, corClasses, Dim.corSpatial, Initialize.corStruct, summary.corStruct

Examples

```
## covariate is observation order and grouping factor is Mare
cs1 <- corAR1(0.2, form = ~ 1 | Mare)

# Pinheiro and Bates, p. 236
cs1AR1 <- corAR1(0.8, form = ~ 1 | Subject)
cs1AR1. <- Initialize(cs1AR1, data = Orthodont)
corMatrix(cs1AR1.)

# Pinheiro and Bates, p. 240
fm1Ovar.lme <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time),
                   data = Ovary, random = pdDiag(~sin(2*pi*Time)))
fm2Ovar.lme <- update(fm1Ovar.lme, correlation = corAR1())

# Pinheiro and Bates, pp. 255-258:  use in gls
fm1Dial.gls <-
  gls(rate ~(pressure + I(pressure^2) + I(pressure^3) + I(pressure^4))*QB,
      Dialyzer)
fm2Dial.gls <- update(fm1Dial.gls,
                  weights = varPower(form = ~ pressure))
fm3Dial.gls <- update(fm2Dial.gls,
                    corr = corAR1(0.771, form = ~ 1 | Subject))

# Pinheiro and Bates use in nlme:
# from p. 240 needed on p. 396
fm1Ovar.lme <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time),
                   data = Ovary, random = pdDiag(~sin(2*pi*Time)))
fm5Ovar.lme <- update(fm1Ovar.lme,
                 correlation = corARMA(p = 1, q = 1))
# p. 396
fm1Ovar.nlme <- nlme(follicles~
     A+B*sin(2*pi*w*Time)+C*cos(2*pi*w*Time),
```

```
   data=Ovary, fixed=A+B+C+w~1,
   random=pdDiag(A+B+w~1),
   start=c(fixef(fm5Ovar.lme), 1) )
# p. 397
fm2Ovar.nlme <- update(fm1Ovar.nlme,
         correlation=corAR1(0.311) )
```

---

corARMA *ARMA(p,q) Correlation Structure*

Description

This function is a constructor for the corARMA class, representing an autocorrelation-moving average correlation structure of order (p, q). Objects created using this constructor must later be initialized using the appropriate Initialize method.

Usage

```
corARMA(value, form, p, q, fixed)
```

Arguments

value	a vector with the values of the autoregressive and moving average parameters, which must have length p + q and all elements between -1 and 1. Defaults to a vector of zeros, corresponding to uncorrelated observations.
form	a one sided formula of the form ~ t, or ~ t | g, specifying a time covariate t and, optionally, a grouping factor g. A covariate for this correlation structure must be integer valued. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
p, q	non-negative integers specifying respectively the autoregressive order and the moving average order of the ARMA structure. Both default to 0, but at least one should be > 0.
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class corARMA, representing an autocorrelation-moving average correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 236, 397.

See Also

corAR1, corClasses Initialize.corStruct, summary.corStruct

Examples

```
## ARMA(1,2) structure, with observation order as a covariate and
## Mare as grouping factor
cs1 <- corARMA(c(0.2, 0.3, -0.1), form = ~ 1 | Mare, p = 1, q = 2)

# Pinheiro and Bates, p. 237
cs1ARMA <- corARMA(0.4, form = ~ 1 | Subject, q = 1)
cs1ARMA <- Initialize(cs1ARMA, data = Orthodont)
corMatrix(cs1ARMA)

cs2ARMA <- corARMA(c(0.8, 0.4), form = ~ 1 | Subject, p=1, q=1)
cs2ARMA <- Initialize(cs2ARMA, data = Orthodont)
corMatrix(cs2ARMA)

# Pinheiro and Bates use in nlme:
# from p. 240 needed on p. 396
fm1Ovar.lme <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time),
                   data = Ovary, random = pdDiag(~sin(2*pi*Time)))
fm5Ovar.lme <- update(fm1Ovar.lme,
                 correlation = corARMA(p = 1, q = 1))
# p. 396
fm1Ovar.nlme <- nlme(follicles~
     A+B*sin(2*pi*w*Time)+C*cos(2*pi*w*Time),
   data=Ovary, fixed=A+B+C+w~1,
   random=pdDiag(A+B+w~1),
   start=c(fixef(fm5Ovar.lme), 1) )
# p. 397
fm3Ovar.nlme <- update(fm1Ovar.nlme,
         correlation=corARMA(p=0, q=2) )
```

---

corCAR1 *Continuous AR(1) Correlation Structure*

Description

This function is a constructor for the corCAR1 class, representing an autocorrelation structure of order 1, with a continuous time covariate. Objects created using this constructor must be later initialized using the appropriate Initialize method.

Usage

```
corCAR1(value, form, fixed)
```

Arguments

value	the correlation between two observations one unit of time apart. Must be between 0 and 1. Defaults to 0.2.
form	a one sided formula of the form ~ t, or ~ t | g, specifying a time covariate t and, optionally, a grouping factor g. Covariates for this correlation structure need not be integer valued. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class corCAR1, representing an autocorrelation structure of order 1, with a continuous time covariate.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Jones, R.H. (1993) "Longitudinal Data with Serial Correlation: A State-space Approach", Chapman and Hall.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 236, 243.

See Also

corClasses, Initialize.corStruct, summary.corStruct

Examples

```
## covariate is Time and grouping factor is Mare
cs1 <- corCAR1(0.2, form = ~ Time | Mare)

# Pinheiro and Bates, pp. 240, 243
fm1Ovar.lme <- lme(follicles ~
           sin(2*pi*Time) + cos(2*pi*Time),
   data = Ovary, random = pdDiag(~sin(2*pi*Time)))
```

```
fm4Ovar.lme <- update(fm1Ovar.lme,
          correlation = corCAR1(form = ~Time))
```

---

corClasses *Correlation Structure Classes*

Description

Standard classes of correlation structures (corStruct) available in the **nlme** package.

Value

Available standard classes:

corAR1	autoregressive process of order 1.
corARMA	autoregressive moving average process, with arbitrary orders for the autoregressive and moving average components.
corCAR1	continuous autoregressive process (AR(1) process for a continuous time covariate).
corCompSymm	compound symmetry structure corresponding to a constant correlation.
corExp	exponential spatial correlation.
corGaus	Gaussian spatial correlation.
corLin	linear spatial correlation.
corRatio	Rational quadratics spatial correlation.
corSpher	spherical spatial correlation.
corSymm	general correlation matrix, with no additional structure.

Note

Users may define their own corStruct classes by specifying a constructor function and, at a minimum, methods for the functions corMatrix and coef. For examples of these functions, see the methods for classes corSymm and corAR1.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

corAR1, corARMA, corCAR1, corCompSymm, corExp, corGaus, corLin, corRatio, corSpher, corSymm, summary.corStruct

corCompSymm *Compound Symmetry Correlation Structure*

Description

This function is a constructor for the corCompSymm class, representing a compound symmetry structure corresponding to uniform correlation. Objects created using this constructor must later be initialized using the appropriate Initialize method.

Usage

```
corCompSymm(value, form, fixed)
```

Arguments

value	the correlation between any two correlated observations. Defaults to 0.
form	a one sided formula of the form ~ t, or ~ t | g, specifying a time covariate t and, optionally, a grouping factor g. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class corCompSymm, representing a compound symmetry correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Milliken, G. A. and Johnson, D. E. (1992) "Analysis of Messy Data, Volume I: Designed Experiments", Van Nostrand Reinhold.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 233-234.

See Also

corClasses, Initialize.corStruct, summary.corStruct

Examples

```
## covariate is observation order and grouping factor is Subject
cs1 <- corCompSymm(0.5, form = ~ 1 | Subject)
cs1 # Uninitialized ...


# Pinheiro and Bates, p. 225
cs1CompSymm <- corCompSymm(value = 0.3, form = ~ 1 | Subject)
cs2CompSymm <- corCompSymm(value = 0.3, form = ~ age | Subject)
cs1CompSymm <- Initialize(cs1CompSymm, data = Orthodont)
corMatrix(cs1CompSymm)
```

---

corExp *Exponential Correlation Structure*

Description

This function is a constructor for the "corExp" class, representing an exponential spatial correlation structure. Letting d denote the range and n denote the nugget effect, the correlation between two observations a distance r apart is $\exp(-r/d)$ when no nugget effect is present and $(1-n)\exp(-r/d)$ when a nugget effect is assumed. Objects created using this constructor must later be initialized using the appropriate Initialize method.

Usage

```
corExp(value, form, nugget, metric, fixed)
```

Arguments

value	an optional vector with the parameter values in constrained form. If nugget is FALSE, value can have only one element, corresponding to the "range" of the exponential correlation structure, which must be greater than zero. If nugget is TRUE, meaning that a nugget effect is present, value can contain one or two elements, the first being the "range" and the second the "nugget effect" (one minus the correlation between two observations taken arbitrarily close together); the first must be greater than zero and the second must be between zero and one. Defaults to numeric(0), which results in a range of 90% of the minimum distance and a nugget effect of 0.1 being assigned to the parameters when object is initialized.
form	a one sided formula of the form ~ S1+...+Sp, or ~ S1+...+Sp | g, specifying spatial covariates S1 through Sp and, optionally, a grouping factor g. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.

nugget	an optional logical value indicating whether a nugget effect is present. Defaults to FALSE.
metric	an optional character string specifying the distance metric to be used. The currently available options are "euclidean" for the root sum-of-squares of distances; "maximum" for the maximum difference; and "manhattan" for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to "euclidean".
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class "corExp", also inheriting from class "corSpatial", representing an exponential spatial correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

Littel, Milliken, Stroup, and Wolfinger (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. p. 238.

See Also

corClasses, Initialize.corStruct, summary.corStruct, dist

Examples

```
sp1 <- corExp(form = ~ x + y + z)

# Pinheiro and Bates, p. 238
spatDat <- data.frame(x = (0:4)/4, y = (0:4)/4)

cs1Exp <- corExp(1, form = ~ x + y)
cs1Exp <- Initialize(cs1Exp, spatDat)
corMatrix(cs1Exp)

cs2Exp <- corExp(1, form = ~ x + y, metric = "man")
cs2Exp <- Initialize(cs2Exp, spatDat)
corMatrix(cs2Exp)

cs3Exp <- corExp(c(1, 0.2), form = ~ x + y,
```

```
                  nugget = TRUE)
cs3Exp <- Initialize(cs3Exp, spatDat)
corMatrix(cs3Exp)

# example lme(..., corExp ...)
# Pinheiro and Bates, pp. 222-247
# p. 222
options(contrasts = c("contr.treatment", "contr.poly"))
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                   random = ~ Time)
# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p. 246
fm3BW.lme <- update(fm2BW.lme,
           correlation = corExp(form = ~ Time))
# p. 247
fm4BW.lme <-
      update(fm3BW.lme, correlation = corExp(form =  ~ Time,
                        nugget = TRUE))
anova(fm3BW.lme, fm4BW.lme)
```

`corFactor` *Factor of a Correlation Matrix*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include all `"corStruct"` classes, see 'corClasses'.

Usage

```
corFactor(object, ...)
```

Arguments

`object`	an object from which a correlation matrix can be extracted.
`...`	some methods for this generic function require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

corFactor.corStruct, recalc.corStruct

---

corFactor.corStruct *Factor of a corStruct Object Matrix*

Description

This method function extracts a transpose inverse square-root factor, or a series of transpose inverse square-root factors, of the correlation matrix, or list of correlation matrices, represented by `object`. Letting Σ denote a correlation matrix, a square-root factor of Σ is any square matrix L such that $\Sigma = L'L$. This method extracts L^{-t}.

Usage

```
## S3 method for class 'corStruct'
corFactor(object, ...)
```

Arguments

`object`	an object inheriting from class `"corStruct"` representing a correlation structure, which must have been initialized (using `Initialize`).
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

If the correlation structure does not include a grouping factor, the returned value will be a vector with a transpose inverse square-root factor of the correlation matrix associated with `object` stacked column-wise. If the correlation structure includes a grouping factor, the returned value will be a vector with transpose inverse square-root factors of the correlation matrices for each group, stacked by group and stacked column-wise within each group.

Note

This method function is used intensively in optimization algorithms and its value is returned as a vector for efficiency reasons. The `corMatrix` method function can be used to obtain transpose inverse square-root factors in matrix form.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

corFactor, corMatrix.corStruct, recalc.corStruct, Initialize.corStruct

Examples

```
cs1 <- corAR1(form = ~1 | Subject)
cs1 <- Initialize(cs1, data = Orthodont)
corFactor(cs1)
```

---

corGaus *Gaussian Correlation Structure*

Description

This function is a constructor for the corGaus class, representing a Gaussian spatial correlation structure. Letting d denote the range and n denote the nugget effect, the correlation between two observations a distance r apart is $\exp(-(r/d)^2)$ when no nugget effect is present and $(1 - n)\exp(-(r/d)^2)$ when a nugget effect is assumed. Objects created using this constructor must later be initialized using the appropriate ' Initialize method.

Usage

```
corGaus(value, form, nugget, metric, fixed)
```

Arguments

value	an optional vector with the parameter values in constrained form. If nugget is FALSE, value can have only one element, corresponding to the "range" of the Gaussian correlation structure, which must be greater than zero. If nugget is TRUE, meaning that a nugget effect is present, value can contain one or two elements, the first being the "range" and the second the "nugget effect" (one minus the correlation between two observations taken arbitrarily close together); the first must be greater than zero and the second must be between zero and one. Defaults to numeric(0), which results in a range of 90% of the minimum distance and a nugget effect of 0.1 being assigned to the parameters when object is initialized.
form	a one sided formula of the form ~ S1+...+Sp, or ~ S1+...+Sp | g, specifying spatial covariates S1 through Sp and, optionally, a grouping factor g. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
nugget	an optional logical value indicating whether a nugget effect is present. Defaults to FALSE.
metric	an optional character string specifying the distance metric to be used. The currently available options are "euclidean" for the root sum-of-squares of distances; "maximum" for the maximum difference; and "manhattan" for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to "euclidean".

`fixed` an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to `FALSE`, in which case the coefficients are allowed to vary.

Value

an object of class `corGaus`, also inheriting from class `corSpatial`, representing a Gaussian spatial correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

Littel, Milliken, Stroup, and Wolfinger (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`Initialize.corStruct`, `summary.corStruct`, `dist`

Examples

```
sp1 <- corGaus(form = ~ x + y + z)

# example lme(..., corGaus ...)
# Pinheiro and Bates, pp. 222-249
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                  random = ~ Time)
# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p 246
fm3BW.lme <- update(fm2BW.lme,
           correlation = corExp(form = ~ Time))
# p. 249
fm8BW.lme <- update(fm3BW.lme, correlation = corGaus(form = ~ Time))
```

corLin *Linear Correlation Structure*

Description

This function is a constructor for the corLin class, representing a linear spatial correlation structure. Letting d denote the range and n denote the nugget effect, the correlation between two observations a distance $r < d$ apart is $1 - (r/d)$ when no nugget effect is present and $(1 - n)(1 - (r/d))$ when a nugget effect is assumed. If $r \geq d$ the correlation is zero. Objects created using this constructor must later be initialized using the appropriate Initialize method.

Usage

```
corLin(value, form, nugget, metric, fixed)
```

Arguments

value	an optional vector with the parameter values in constrained form. If nugget is FALSE, value can have only one element, corresponding to the "range" of the linear correlation structure, which must be greater than zero. If nugget is TRUE, meaning that a nugget effect is present, value can contain one or two elements, the first being the "range" and the second the "nugget effect" (one minus the correlation between two observations taken arbitrarily close together); the first must be greater than zero and the second must be between zero and one. Defaults to numeric(0), which results in a range of 90% of the minimum distance and a nugget effect of 0.1 being assigned to the parameters when object is initialized.
form	a one sided formula of the form ~ S1+...+Sp, or ~ S1+...+Sp | g, specifying spatial covariates S1 through Sp and, optionally, a grouping factor g. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
nugget	an optional logical value indicating whether a nugget effect is present. Defaults to FALSE.
metric	an optional character string specifying the distance metric to be used. The currently available options are "euclidean" for the root sum-of-squares of distances; "maximum" for the maximum difference; and "manhattan" for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to "euclidean".
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class `corLin`, also inheriting from class `corSpatial`, representing a linear spatial correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

Littel, Milliken, Stroup, and Wolfinger (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`Initialize.corStruct`, `summary.corStruct`, `dist`

Examples

```
sp1 <- corLin(form = ~ x + y)

# example lme(..., corLin ...)
# Pinheiro and Bates, pp. 222-249
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                 random = ~ Time)
# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p 246
fm3BW.lme <- update(fm2BW.lme,
           correlation = corExp(form = ~ Time))
# p. 249
fm7BW.lme <- update(fm3BW.lme, correlation = corLin(form = ~ Time))
```

---

corMatrix *Extract Correlation Matrix*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include all `"corStruct"` classes, see 'corClasses'.

Usage

```
corMatrix(object, ...)
```

Arguments

object	an object for which a correlation matrix can be extracted.
...	some methods for this generic function require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

corMatrix.corStruct, corMatrix.pdMat, corMatrix.reStruct

---

corMatrix.corStruct *Matrix of a corStruct Object*

Description

This method function extracts the correlation matrix (or its transpose inverse square-root factor), or list of correlation matrices (or their transpose inverse square-root factors) corresponding to covariate and object. Letting Σ denote a correlation matrix, a square-root factor of Σ is any square matrix L such that $\Sigma = L'L$. When corr = FALSE, this method extracts L^{-t}.

Usage

```
## S3 method for class 'corStruct'
corMatrix(object, covariate, corr, ...)
```

Arguments

object	an object inheriting from class "corStruct" representing a correlation structure.
covariate	an optional covariate vector (matrix), or list of covariate vectors (matrices), at which values the correlation matrix, or list of correlation matrices, are to be evaluated. Defaults to getCovariate(object).
corr	a logical value. If TRUE the function returns the correlation matrix, or list of correlation matrices, represented by object. If FALSE the function returns a transpose inverse square-root of the correlation matrix, or a list of transpose inverse square-root factors of the correlation matrices.
...	some methods for this generic require additional arguments. None are used in this method.

Value

If `covariate` is a vector (matrix), the returned value will be an array with the corresponding correlation matrix (or its transpose inverse square-root factor). If the `covariate` is a list of vectors (matrices), the returned value will be a list with the correlation matrices (or their transpose inverse square-root factors) corresponding to each component of `covariate`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`corFactor.corStruct`, `Initialize.corStruct`

Examples

```
cs1 <- corAR1(0.3)
corMatrix(cs1, covariate = 1:4)
corMatrix(cs1, covariate = 1:4, corr = FALSE)

# Pinheiro and Bates, p. 225
cs1CompSymm <- corCompSymm(value = 0.3, form = ~ 1 | Subject)
cs1CompSymm <- Initialize(cs1CompSymm, data = Orthodont)
corMatrix(cs1CompSymm)

# Pinheiro and Bates, p. 226
cs1Symm <- corSymm(value = c(0.2, 0.1, -0.1, 0, 0.2, 0),
                   form = ~ 1 | Subject)
cs1Symm <- Initialize(cs1Symm, data = Orthodont)
corMatrix(cs1Symm)

# Pinheiro and Bates, p. 236
cs1AR1 <- corAR1(0.8, form = ~ 1 | Subject)
cs1AR1 <- Initialize(cs1AR1, data = Orthodont)
corMatrix(cs1AR1)

# Pinheiro and Bates, p. 237
cs1ARMA <- corARMA(0.4, form = ~ 1 | Subject, q = 1)
cs1ARMA <- Initialize(cs1ARMA, data = Orthodont)
corMatrix(cs1ARMA)

# Pinheiro and Bates, p. 238
spatDat <- data.frame(x = (0:4)/4, y = (0:4)/4)
cs1Exp <- corExp(1, form = ~ x + y)
cs1Exp <- Initialize(cs1Exp, spatDat)
corMatrix(cs1Exp)
```

corMatrix.pdMat *Extract Correlation Matrix from a pdMat Object*

Description

The correlation matrix corresponding to the positive-definite matrix represented by `object` is obtained.

Usage

```
## S3 method for class 'pdMat'
corMatrix(object, ...)
```

Arguments

`object`	an object inheriting from class `"pdMat"`, representing a positive definite matrix.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the correlation matrix corresponding to the positive-definite matrix represented by `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`as.matrix.pdMat`, `pdMatrix`

Examples

```
pd1 <- pdSymm(diag(1:4))
corMatrix(pd1)
```

corMatrix.reStruct *Extract Correlation Matrix from Components of an reStruct Object*

Description

This method function extracts the correlation matrices corresponding to the pdMat elements of object.

Usage

```
## S3 method for class 'reStruct'
corMatrix(object, ...)
```

Arguments

object	an object inheriting from class "reStruct", representing a random effects structure and consisting of a list of pdMat objects.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a list with components given by the correlation matrices corresponding to the elements of object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

as.matrix.reStruct, corMatrix, reStruct, pdMat

Examples

```
rs1 <- reStruct(pdSymm(diag(3), ~age+Sex, data = Orthodont))
corMatrix(rs1)
```

corNatural *General correlation in natural parameterization*

Description

This function is a constructor for the corNatural class, representing a general correlation structure in the "natural" parameterization, which is described under pdNatural. Objects created using this constructor must later be initialized using the appropriate Initialize method.

Usage

```
corNatural(value, form, fixed)
```

Arguments

value	an optional vector with the parameter values. Default is numeric(0), which results in a vector of zeros of appropriate dimension being assigned to the parameters when object is initialized (corresponding to an identity correlation structure).
form	a one sided formula of the form ~ t, or ~ t | g, specifying a time covariate t and, optionally, a grouping factor g. A covariate for this correlation structure must be integer valued. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class corNatural representing a general correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Initialize.corNatural, pdNatural, summary.corNatural

Examples

```
## covariate is observation order and grouping factor is Subject
cs1 <- corNatural(form = ~ 1 | Subject)
cs1 # Uninitialized ...
summary(Initialize(cs1, data = Orthodont))
```

corRatio *Rational Quadratic Correlation Structure*

Description

This function is a constructor for the corRatio class, representing a rational quadratic spatial correlation structure. Letting d denote the range and n denote the nugget effect, the correlation between two observations a distance r apart is $1/(1 + (r/d)^2)$ when no nugget effect is present and $(1 - n)/(1 + (r/d)^2)$ when a nugget effect is assumed. Objects created using this constructor need to be later initialized using the appropriate Initialize method.

Usage

```
corRatio(value, form, nugget, metric, fixed)
```

Arguments

value	an optional vector with the parameter values in constrained form. If nugget is FALSE, value can have only one element, corresponding to the "range" of the rational quadratic correlation structure, which must be greater than zero. If nugget is TRUE, meaning that a nugget effect is present, value can contain one or two elements, the first being the "range" and the second the "nugget effect" (one minus the correlation between two observations taken arbitrarily close together); the first must be greater than zero and the second must be between zero and one. Defaults to numeric(0), which results in a range of 90% of the minimum distance and a nugget effect of 0.1 being assigned to the parameters when object is initialized.
form	a one sided formula of the form ~ S1+...+Sp, or ~ S1+...+Sp | g, specifying spatial covariates S1 through Sp and, optionally, a grouping factor g. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
nugget	an optional logical value indicating whether a nugget effect is present. Defaults to FALSE.
metric	an optional character string specifying the distance metric to be used. The currently available options are "euclidean" for the root sum-of-squares of distances; "maximum" for the maximum difference; and "manhattan" for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to "euclidean".
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class `corRatio`, also inheriting from class `corSpatial`, representing a rational quadratic spatial correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

Littel, Milliken, Stroup, and Wolfinger (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`Initialize.corStruct`, `summary.corStruct`, `dist`

Examples

```
sp1 <- corRatio(form = ~ x + y + z)

# example lme(..., corRatio ...)
# Pinheiro and Bates, pp. 222-249
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                  random = ~ Time)
# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p 246
fm3BW.lme <- update(fm2BW.lme,
           correlation = corExp(form = ~ Time))
# p. 249
fm5BW.lme <- update(fm3BW.lme, correlation =
                  corRatio(form = ~ Time))

# example gls(..., corRatio ...)
# Pinheiro and Bates, pp. 261, 263
fm1Wheat2 <- gls(yield ~ variety - 1, Wheat2)
# p. 263
fm3Wheat2 <- update(fm1Wheat2, corr =
    corRatio(c(12.5, 0.2),
       form = ~ latitude + longitude,
             nugget = TRUE))
```

corSpatial *Spatial Correlation Structure*

Description

This function is a constructor for the corSpatial class, representing a spatial correlation structure. This class is "virtual", having four "real" classes, corresponding to specific spatial correlation structures, associated with it: corExp, corGaus, corLin, corRatio, and corSpher. The returned object will inherit from one of these "real" classes, determined by the type argument, and from the "virtual" corSpatial class. Objects created using this constructor must later be initialized using the appropriate Initialize method.

Usage

```
corSpatial(value, form, nugget, type, metric, fixed)
```

Arguments

value	an optional vector with the parameter values in constrained form. If nugget is FALSE, value can have only one element, corresponding to the "range" of the spatial correlation structure, which must be greater than zero. If nugget is TRUE, meaning that a nugget effect is present, value can contain one or two elements, the first being the "range" and the second the "nugget effect" (one minus the correlation between two observations taken arbitrarily close together); the first must be greater than zero and the second must be between zero and one. Defaults to numeric(0), which results in a range of 90% of the minimum distance and a nugget effect of 0.1 being assigned to the parameters when object is initialized.
form	a one sided formula of the form ~ S1+...+Sp, or ~ S1+...+Sp | g, specifying spatial covariates S1 through Sp and, optionally, a grouping factor g. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
nugget	an optional logical value indicating whether a nugget effect is present. Defaults to FALSE.
type	an optional character string specifying the desired type of correlation structure. Available types include "spherical", "exponential", "gaussian", "linear", and "rational". See the documentation on the functions corSpher, corExp, corGaus, corLin, and corRatio for a description of these correlation structures. Partial matching of arguments is used, so only the first character needs to be provided.Defaults to "spherical".
metric	an optional character string specifying the distance metric to be used. The currently available options are "euclidean" for the root sum-of-squares of distances; "maximum" for the maximum difference; and "manhattan" for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to "euclidean".

`fixed` an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to `FALSE`, in which case the coefficients are allowed to vary.

Value

an object of class determined by the `type` argument and also inheriting from class `corSpatial`, representing a spatial correlation structure.

Author(s)

José Pinheiro and Douglas Bates `<bates@stat.wisc.edu>`

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

Littel, Milliken, Stroup, and Wolfinger (1996) "SAS Systems for Mixed Models", SAS Institute.

See Also

`corExp`, `corGaus`, `corLin`, `corRatio`, `corSpher`, `Initialize.corStruct`, `summary.corStruct`, `dist`

Examples

```
sp1 <- corSpatial(form = ~ x + y + z, type = "g", metric = "man")
```

---

corSpher *Spherical Correlation Structure*

Description

This function is a constructor for the `corSpher` class, representing a spherical spatial correlation structure. Letting d denote the range and n denote the nugget effect, the correlation between two observations a distance $r < d$ apart is $1-1.5(r/d)+0.5(r/d)^3$ when no nugget effect is present and $(1-n)(1-1.5(r/d)+0.5(r/d)^3)$ when a nugget effect is assumed. If $r \geq d$ the correlation is zero. Objects created using this constructor must later be initialized using the appropriate `Initialize` method.

Usage

```
corSpher(value, form, nugget, metric, fixed)
```

Arguments

value	an optional vector with the parameter values in constrained form. If nugget is FALSE, value can have only one element, corresponding to the "range" of the spherical correlation structure, which must be greater than zero. If nugget is TRUE, meaning that a nugget effect is present, value can contain one or two elements, the first being the "range" and the second the "nugget effect" (one minus the correlation between two observations taken arbitrarily close together); the first must be greater than zero and the second must be between zero and one. Defaults to numeric(0), which results in a range of 90% of the minimum distance and a nugget effect of 0.1 being assigned to the parameters when object is initialized.
form	a one sided formula of the form ~ S1+...+Sp, or ~ S1+...+Sp | g, specifying spatial covariates S1 through Sp and, optionally, a grouping factor g. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
nugget	an optional logical value indicating whether a nugget effect is present. Defaults to FALSE.
metric	an optional character string specifying the distance metric to be used. The currently available options are "euclidean" for the root sum-of-squares of distances; "maximum" for the maximum difference; and "manhattan" for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to "euclidean".
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class corSpher, also inheriting from class corSpatial, representing a spherical spatial correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

Littel, Milliken, Stroup, and Wolfinger (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

Initialize.corStruct, summary.corStruct, dist

Examples

```
sp1 <- corSpher(form = ~ x + y)

# example lme(..., corSpher ...)
# Pinheiro and Bates, pp. 222-249
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                  random = ~ Time)
# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p 246
fm3BW.lme <- update(fm2BW.lme,
           correlation = corExp(form = ~ Time))
# p. 249
fm6BW.lme <- update(fm3BW.lme,
          correlation = corSpher(form = ~ Time))

# example gls(..., corSpher ...)
# Pinheiro and Bates, pp. 261, 263
fm1Wheat2 <- gls(yield ~ variety - 1, Wheat2)
# p. 262
fm2Wheat2 <- update(fm1Wheat2, corr =
   corSpher(c(28, 0.2),
     form = ~ latitude + longitude, nugget = TRUE))
```

---

corSymm *General Correlation Structure*

Description

This function is a constructor for the corSymm class, representing a general correlation structure. The internal representation of this structure, in terms of unconstrained parameters, uses the spherical parametrization defined in Pinheiro and Bates (1996). Objects created using this constructor must later be initialized using the appropriate Initialize method.

Usage

```
corSymm(value, form, fixed)
```

Arguments

value	an optional vector with the parameter values. Default is numeric(0), which results in a vector of zeros of appropriate dimension being assigned to the parameters when object is initialized (corresponding to an identity correlation structure).

form	a one sided formula of the form ~ t, or ~ t | g, specifying a time covariate t and, optionally, a grouping factor g. A covariate for this correlation structure must be integer valued. When a grouping factor is present in form, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to ~ 1, which corresponds to using the order of the observations in the data as a covariate, and no groups.
fixed	an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to FALSE, in which case the coefficients are allowed to vary.

Value

an object of class corSymm representing a general correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289-296.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

Initialize.corSymm, summary.corSymm

Examples

```
## covariate is observation order and grouping factor is Subject
cs1 <- corSymm(form = ~ 1 | Subject)

# Pinheiro and Bates, p. 225
cs1CompSymm <- corCompSymm(value = 0.3, form = ~ 1 | Subject)
cs1CompSymm <- Initialize(cs1CompSymm, data = Orthodont)
corMatrix(cs1CompSymm)

# Pinheiro and Bates, p. 226
cs1Symm <- corSymm(value =
        c(0.2, 0.1, -0.1, 0, 0.2, 0),
                   form = ~ 1 | Subject)
cs1Symm <- Initialize(cs1Symm, data = Orthodont)
corMatrix(cs1Symm)

# example gls(..., corSpher ...)
# Pinheiro and Bates, pp. 261, 263
fm1Wheat2 <- gls(yield ~ variety - 1, Wheat2)
# p. 262
fm2Wheat2 <- update(fm1Wheat2, corr =
```

```
   corSpher(c(28, 0.2),
     form = ~ latitude + longitude, nugget = TRUE))

# example gls(..., corSymm ... )
# Pinheiro and Bates, p. 251
fm1Orth.gls <- gls(distance ~ Sex * I(age - 11), Orthodont,
                   correlation = corSymm(form = ~ 1 | Subject),
                   weights = varIdent(form = ~ 1 | age))
```

`Covariate` *Assign Covariate Values*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include all "varFunc" classes, see 'varClasses'.

Usage

```
covariate(object) <- value
```

Arguments

`object`	any object with a `covariate` component.
`value`	a value to be assigned to the covariate associated with `object`.

Value

will depend on the method function; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

covariate<-.varFunc, getCovariate

Covariate.varFunc *Assign varFunc Covariate*

Description

The covariate(s) used in the calculation of the weights of the variance function represented by `object` is (are) replaced by `value`. If `object` has been initialized, `value` must have the same dimensions as `getCovariate(object)`.

Usage

```
## S3 replacement method for class 'varFunc'
covariate(object) <- value
```

Arguments

`object`	an object inheriting from class "`varFunc`", representing a variance function structure.
`value`	a value to be assigned to the covariate associated with `object`.

Value

a `varFunc` object similar to `object`, but with its `covariate` attribute replaced by `value`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`getCovariate.varFunc`

Examples

```
vf1 <- varPower(1.1, form = ~age)
covariate(vf1) <- Orthodont[["age"]]
```

`Dialyzer` *High-Flux Hemodialyzer*

Description

The `Dialyzer` data frame has 140 rows and 5 columns.

Format

This data frame contains the following columns:

Subject an ordered factor with levels `10` < `8` < `2` < `6` < `3` < `5` < `9` < `7` < `1` < `4` < `17` < `20` < `11` < `12` < `16` < `13` < `14` < `18` < `15` < `19` giving the unique identifier for each subject

QB a factor with levels `200` and `300` giving the bovine blood flow rate (dL/min).

pressure a numeric vector giving the transmembrane pressure (dmHg).

rate the hemodialyzer ultrafiltration rate (mL/hr).

index index of observation within subject—1 through 7.

Details

Vonesh and Carter (1992) describe data measured on high-flux hemodialyzers to assess their *in vivo* ultrafiltration characteristics. The ultrafiltration rates (in mL/hr) of 20 high-flux dialyzers were measured at seven different transmembrane pressures (in dmHg). The *in vitro* evaluation of the dialyzers used bovine blood at flow rates of either 200~dl/min or 300~dl/min. The data, are also analyzed in Littell, Milliken, Stroup, and Wolfinger (1996).

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.6)

Vonesh, E. F. and Carter, R. L. (1992), Mixed-effects nonlinear regression for unbalanced repeated measures, *Biometrics*, **48**, 1-18.

Littell, R. C., Milliken, G. A., Stroup, W. W. and Wolfinger, R. D. (1996), *SAS System for Mixed Models*, SAS Institute, Cary, NC.

Dim *Extract Dimensions from an Object*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: "corSpatial", "corStruct", "pdCompSymm", "pdDiag", "pdIdent", "pdMat", and "pdSymm".

Usage

```
Dim(object, ...)
```

Arguments

object	any object for which dimensions can be extracted.
...	some methods for this generic function require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

Note

If dim allowed more than one argument, there would be no need for this generic function.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Dim.corSpatial, Dim.pdMat, Dim.corStruct

Dim.corSpatial *Dimensions of a corSpatial Object*

Description

if groups is missing, it returns the Dim attribute of object; otherwise, calculates the dimensions associated with the grouping factor.

Usage

```
## S3 method for class 'corSpatial'
Dim(object, groups, ...)
```

Arguments

object	an object inheriting from class "corSpatial", representing a spatial correlation structure.
groups	an optional factor defining the grouping of the observations; observations within a group are correlated and observations in different groups are uncorrelated.
...	further arguments to be passed to or from methods.

Value

a list with components:

N	length of groups
M	number of groups
spClass	an integer representing the spatial correlation class; 0 = user defined class, 1 = corSpher, 2 = corExp, 3 = corGaus, 4 = corLin
sumLenSq	sum of the squares of the number of observations per group
len	an integer vector with the number of observations per group
start	an integer vector with the starting position for the distance vectors in each group, beginning from zero

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Dim, Dim.corStruct

Examples

```
Dim(corGaus(), getGroups(Orthodont))

cs1ARMA <- corARMA(0.4, form = ~ 1 | Subject, q = 1)
cs1ARMA <- Initialize(cs1ARMA, data = Orthodont)
Dim(cs1ARMA)
```

Dim.corStruct *Dimensions of a corStruct Object*

Description

if groups is missing, it returns the Dim attribute of object; otherwise, calculates the dimensions associated with the grouping factor.

Usage

```
## S3 method for class 'corStruct'
Dim(object, groups, ...)
```

Arguments

object	an object inheriting from class "corStruct", representing a correlation structure.
groups	an optional factor defining the grouping of the observations; observations within a group are correlated and observations in different groups are uncorrelated.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a list with components:

N	length of groups
M	number of groups
maxLen	maximum number of observations in a group
sumLenSq	sum of the squares of the number of observations per group
len	an integer vector with the number of observations per group
start	an integer vector with the starting position for the observations in each group, beginning from zero

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Dim, Dim.corSpatial

Examples

```
Dim(corAR1(), getGroups(Orthodont))
```

Dim.pdMat *Dimensions of a pdMat Object*

Description

This method function returns the dimensions of the matrix represented by `object`.

Usage

```
## S3 method for class 'pdMat'
Dim(object, ...)
```

Arguments

`object`	an object inheriting from class `"pdMat"`, representing a positive-definite matrix.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

an integer vector with the number of rows and columns of the matrix represented by `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`Dim`

Examples

```
Dim(pdSymm(diag(3)))
```

Earthquake *Earthquake Intensity*

Description

The `Earthquake` data frame has 182 rows and 5 columns.

Format

This data frame contains the following columns:

Quake an ordered factor with levels `20` < `16` < `14` < `10` < `3` < `8` < `23` < `22` < `6` < `13` < `7` < `21` < `18` < `15` < `4` < `12` < `19` < `5` < `9` < `1` < `2` < `17` < `11` indicating the earthquake on which the measurements were made.

Richter a numeric vector giving the intensity of the earthquake on the Richter scale.

distance the distance from the seismological measuring station to the epicenter of the earthquake (km).

soil a factor with levels `0` and `1` giving the soil condition at the measuring station, either soil or rock.

accel maximum horizontal acceleration observed (g).

Details

Measurements recorded at available seismometer locations for 23 large earthquakes in western North America between 1940 and 1980. They were originally given in Joyner and Boore (1981); are mentioned in Brillinger (1987); and are analyzed in Davidian and Giltinan (1995).

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.8)

Davidian, M. and Giltinan, D. M. (1995), *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, London.

Joyner and Boore (1981), Peak horizontal acceleration and velocity from strong-motion records including records from the 1979 Imperial Valley, California, earthquake, *Bulletin of the Seismological Society of America*, **71**, 2011-2038.

Brillinger, D. (1987), Comment on a paper by C. R. Rao, *Statistical Science*, **2**, 448-450.

`ergoStool` *Ergometrics experiment with stool types*

Description

The `ergoStool` data frame has 36 rows and 3 columns.

Format

This data frame contains the following columns:

effort a numeric vector giving the effort (Borg scale) required to arise from a stool.

Type a factor with levels `T1`, `T2`, `T3`, and `T4` giving the stool type.

Subject an ordered factor giving a unique identifier for the subject in the experiment.

Details

Devore (2000) cites data from an article in *Ergometrics* (1993, pp. 519-535) on "The Effects of a Pneumatic Stool and a One-Legged Stool on Lower Limb Joint Load and Muscular Activity."

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.9)

Devore, J. L. (2000), *Probability and Statistics for Engineering and the Sciences (5th ed)*, Duxbury, Boston, MA.

Examples

```
fm1 <-
   lme(effort ~ Type, data = ergoStool, random = ~ 1 | Subject)
anova( fm1 )
```

Fatigue *Cracks caused by metal fatigue*

Description

The Fatigue data frame has 262 rows and 3 columns.

Format

This data frame contains the following columns:

Path an ordered factor with levels 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < 11 < 12 < 13 < 14 < 15 < 16 < 17 < 18 < 19 < 20 < 21 giving the test path (or test unit) number. The order is in terms of increasing failure time or decreasing terminal crack length.

cycles number of test cycles at which the measurement is made (millions of cycles).

relLength relative crack length (dimensionless).

Details

These data are given in Lu and Meeker (1993) where they state "We obtained the data in Table 1 visually from figure 4.5.2 on page 242 of Bogdanoff and Kozin (1985)." The data represent the growth of cracks in metal for 21 test units. An initial notch of length 0.90 inches was made on each unit which then was subjected to several thousand test cycles. After every 10,000 test cycles the crack length was measured. Testing was stopped if the crack length exceeded 1.60 inches, defined as a failure, or at 120,000 cycles.

Source

Lu, C. Joséph , and Meeker, William Q. (1993), Using degradation measures to estimate a time-to-failure distribution, *Technometrics*, **35**, 161-174

fdHess *Finite difference Hessian*

Description

Evaluate an approximate Hessian and gradient of a scalar function using finite differences.

Usage

```
fdHess(pars, fun, ...,
       .relStep = .Machine$double.eps^(1/3), minAbsPar = 0)
```

Arguments

pars	the numeric values of the parameters at which to evaluate the function fun and its derivatives.
fun	a function depending on the parameters pars that returns a numeric scalar.
...	Optional additional arguments to fun
.relStep	The relative step size to use in the finite differences. It defaults to the cube root of .Machine$double.eps
minAbsPar	The minimum magnitude of a parameter value that is considered non-zero. It defaults to zero meaning that any non-zero value will be considered different from zero.

Details

This function uses a second-order response surface design known as a "Koschal design" to determine the parameter values at which the function is evaluated.

Value

A list with components

mean	the value of function fun evaluated at the parameter values pars
gradient	an approximate gradient (of length length(pars)).
Hessian	a matrix whose upper triangle contains an approximate Hessian.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

Examples

```
(fdH <- fdHess(c(12.3, 2.34), function(x) x[1]*(1-exp(-0.4*x[2]))))
stopifnot(length(fdH$ mean) == 1,
          length(fdH$ gradient) == 2,
          identical(dim(fdH$ Hessian), c(2L, 2L)))
```

`fitted.glsStruct` *Calculate glsStruct Fitted Values*

Description

The fitted values for the linear model represented by `object` are extracted.

Usage

```
## S3 method for class 'glsStruct'
fitted(object, glsFit, ...)
```

Arguments

`object`	an object inheriting from class `"glsStruct"`, representing a list of linear model components, such as `corStruct` and `"varFunc"` objects.
`glsFit`	an optional list with components `logLik` (log-likelihood), `beta` (coefficients), `sigma` (standard deviation for error term), `varBeta` (coefficients' covariance matrix), `fitted` (fitted values), and `residuals` (residuals). Defaults to `attr(object, "glsFit")`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the fitted values for the linear model represented by `object`.

Note

This method function is generally only used inside `gls` and `fitted.gls`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`gls`, `residuals.glsStruct`

`fitted.gnlsStruct` *Calculate gnlsStruct Fitted Values*

Description

The fitted values for the nonlinear model represented by `object` are extracted.

Usage

```
## S3 method for class 'gnlsStruct'
fitted(object, ...)
```

Arguments

`object`	an object inheriting from class "`gnlsStruct`", representing a list of model components, such as `corStruct` and `varFunc` objects, and attributes specifying the underlying nonlinear model and the response variable.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the fitted values for the nonlinear model represented by `object`.

Note

This method function is generally only used inside `gnls` and `fitted.gnls`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`gnls`, `residuals.gnlsStruct`

`fitted.lme` *Extract lme Fitted Values*

Description

The fitted values at level i are obtained by adding together the population fitted values (based only on the fixed effects estimates) and the estimated contributions of the random effects to the fitted values at grouping levels less or equal to i. The resulting values estimate the best linear unbiased predictions (BLUPs) at level i.

Usage

```
## S3 method for class 'lme'
fitted(object, level, asList, ...)
```

Arguments

`object`	an object inheriting from class `"lme"`, representing a fitted linear mixed-effects model.
`level`	an optional integer vector giving the level(s) of grouping to be used in extracting the fitted values from `object`. Level values increase from outermost to innermost grouping, with level zero corresponding to the population fitted values. Defaults to the highest or innermost level of grouping.
`asList`	an optional logical value. If `TRUE` and a single value is given in `level`, the returned object is a list with the fitted values split by groups; else the returned value is either a vector or a data frame, according to the length of `level`. Defaults to `FALSE`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

If a single level of grouping is specified in `level`, the returned value is either a list with the fitted values split by groups (`asList = TRUE`) or a vector with the fitted values (`asList = FALSE`); else, when multiple grouping levels are specified in `level`, the returned object is a data frame with columns given by the fitted values at different levels and the grouping factors. For a vector or data frame result the `napredict` method is applied.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Bates, D.M. and Pinheiro, J.C. (1998) "Computational methods for multilevel models" available in PostScript or PDF formats at http://nlme.stat.wisc.edu/pub/NLME/

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 235, 397.

See Also

lme, residuals.lme

Examples

```
fm1 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
fitted(fm1, level = 0:1)
```

---

fitted.lmeStruct *Calculate lmeStruct Fitted Values*

Description

The fitted values at level i are obtained by adding together the population fitted values (based only on the fixed effects estimates) and the estimated contributions of the random effects to the fitted values at grouping levels less or equal to i. The resulting values estimate the best linear unbiased predictions (BLUPs) at level i.

Usage

```
## S3 method for class 'lmeStruct'
fitted(object, level, conLin, lmeFit, ...)
```

Arguments

object	an object inheriting from class "lmeStruct", representing a list of linear mixed-effects model components, such as reStruct, corStruct, and varFunc objects.
level	an optional integer vector giving the level(s) of grouping to be used in extracting the fitted values from object. Level values increase from outermost to innermost grouping, with level zero corresponding to the population fitted values. Defaults to the highest or innermost level of grouping.
conLin	an optional condensed linear model object, consisting of a list with components "Xy", corresponding to a regression matrix (X) combined with a response vector (y), and "logLik", corresponding to the log-likelihood of the underlying lme model. Defaults to attr(object, "conLin").
lmeFit	an optional list with components beta and b containing respectively the fixed effects estimates and the random effects estimates to be used to calculate the fitted values. Defaults to attr(object, "lmeFit").
...	some methods for this generic accept other optional arguments.

Value

if a single level of grouping is specified in level, the returned value is a vector with the fitted values at the desired level; else, when multiple grouping levels are specified in level, the returned object is a matrix with columns given by the fitted values at different levels.

Note

This method function is generally only used inside `lme` and `fitted.lme`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lme`, `fitted.lme`, `residuals.lmeStruct`

`fitted.lmList` *Extract lmList Fitted Values*

Description

The fitted values are extracted from each `lm` component of `object` and arranged into a list with as many components as `object`, or combined into a single vector.

Usage

```
## S3 method for class 'lmList'
fitted(object, subset, asList, ...)
```

Arguments

`object`	an object inheriting from class `"lmList"`, representing a list of `lm` objects with a common model.
`subset`	an optional character or integer vector naming the `lm` components of `object` from which the fitted values are to be extracted. Default is `NULL`, in which case all components are used.
`asList`	an optional logical value. If `TRUE`, the returned object is a list with the fitted values split by groups; else the returned value is a vector. Defaults to `FALSE`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a list with components given by the fitted values of each `lm` component of `object`, or a vector with the fitted values for all `lm` components of `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lmList`, `residuals.lmList`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
fitted(fm1)
```

`fitted.nlmeStruct` *Calculate nlmeStruct Fitted Values*

Description

The fitted values at level i are obtained by adding together the contributions from the estimated fixed effects and the estimated random effects at levels less or equal to i and evaluating the model function at the resulting estimated parameters. The resulting values estimate the predictions at level i.

Usage

```
## S3 method for class 'nlmeStruct'
fitted(object, level, conLin, ...)
```

Arguments

`object`	an object inheriting from class `"nlmeStruct"`, representing a list of mixed-effects model components, such as `reStruct`, `corStruct`, and `varFunc` objects, plus attributes specifying the underlying nonlinear model and the response variable.
`level`	an optional integer vector giving the level(s) of grouping to be used in extracting the fitted values from `object`. Level values increase from outermost to innermost grouping, with level zero corresponding to the population fitted values. Defaults to the highest or innermost level of grouping.
`conLin`	an optional condensed linear model object, consisting of a list with components `"Xy"`, corresponding to a regression matrix (`X`) combined with a response vector (`y`), and `"logLik"`, corresponding to the log-likelihood of the underlying nlme model. Defaults to `attr(object, "conLin")`.
`...`	additional arguments that could be given to this method. None are used.

Value

if a single level of grouping is specified in `level`, the returned value is a vector with the fitted values at the desired level; else, when multiple grouping levels are specified in `level`, the returned object is a matrix with columns given by the fitted values at different levels.

Note

This method function is generally only used inside `nlme` and `fitted.nlme`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Bates, D.M. and Pinheiro, J.C. (1998) "Computational methods for multilevel models" available in PostScript or PDF formats at http://nlme.stat.wisc.edu/pub/NLME/

See Also

nlme, residuals.nlmeStruct

---

fixed.effects *Extract Fixed Effects*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include lmList and lme.

fixed.effects is an alias for fixef; methods are implemented for the latter.

Usage

```
fixed.effects(object, ...)
fixef(object, ...)
```

Arguments

object	any fitted model object from which fixed effects estimates can be extracted.
...	some methods for this generic function require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

fixef.lmList

`fixef.lmList` *Extract lmList Fixed Effects*

Description

The average of the coefficients corresponding to the `lm` components of `object` is calculated.

Usage

```
## S3 method for class 'lmList'
fixef(object, ...)
```

Arguments

`object`	an object inheriting from class `"lmList"`, representing a list of `lm` objects with a common model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the average of the individual `lm` coefficients in `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lmList`, `random.effects.lmList`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
fixef(fm1)
fixed.effects(fm1)  # the same, using the longer alias
```

`formula.pdBlocked` *Extract pdBlocked Formula*

Description

The `formula` attributes of the `pdMat` elements of `x` are extracted and returned as a list, in case `asList=TRUE`, or converted to a single one-sided formula when `asList=FALSE`. If the `pdMat` elements do not have a `formula` attribute, a `NULL` value is returned.

Usage

```
## S3 method for class 'pdBlocked'
formula(x, asList, ...)
```

Arguments

`x`	an object inheriting from class `"pdBlocked"`, representing a positive definite block diagonal matrix.
`asList`	an optional logical value. If `TRUE`, a list with the formulas for the individual block diagonal elements of `x` is returned; else, if `FALSE`, a one-sided formula combining all individual formulas is returned. Defaults to `FALSE`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a list of one-sided formulas, or a single one-sided formula, or `NULL`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`pdBlocked`, `pdMat`

Examples

```
pd1 <- pdBlocked(list(~ age, ~ Sex - 1))
formula(pd1)
formula(pd1, asList = TRUE)
```

`formula.pdMat` *Extract pdMat Formula*

Description

This method function extracts the formula associated with a `pdMat` object, in which the column and row names are specified.

Usage

```
## S3 method for class 'pdMat'
formula(x, asList, ...)
```

Arguments

`x`	an object inheriting from class "`pdMat`", representing a positive definite matrix.
`asList`	logical. Should the `asList` argument be applied to each of the components? Never used.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

if `x` has a `formula` attribute, its value is returned, else `NULL` is returned.

Note

Because factors may be present in `formula(x)`, the `pdMat` object needs to have access to a data frame where the variables named in the formula can be evaluated, before it can resolve its row and column names from the formula.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`pdMat`

Examples

```
pd1 <- pdSymm(~Sex*age)
formula(pd1)
```

formula.reStruct *Extract reStruct Object Formula*

Description

This method function extracts a formula from each of the components of x, returning a list of formulas.

Usage

```
## S3 method for class 'reStruct'
formula(x, asList, ...)
```

Arguments

x	an object inheriting from class "reStruct", representing a random effects structure and consisting of a list of pdMat objects.
asList	logical. Should the asList argument be applied to each of the components?
...	some methods for this generic require additional arguments. None are used in this method.

Value

a list with the formulas of each component of x.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

formula

Examples

```
rs1 <- reStruct(list(A = pdDiag(diag(2), ~age), B = ~1))
formula(rs1)
```

gapply *Apply a Function by Groups*

Description

Applies the function to the distinct sets of rows of the data frame defined by `groups`.

Usage

```
gapply(object, which, FUN, form, level, groups, ...)
```

Arguments

`object`	an object to which the function will be applied - usually a `groupedData` object or a `data.frame`. Must inherit from class `"data.frame"`.
`which`	an optional character or positive integer vector specifying which columns of `object` should be used with `FUN`. Defaults to all columns in `object`.
`FUN`	function to apply to the distinct sets of rows of the data frame `object` defined by the values of `groups`.
`form`	an optional one-sided formula that defines the groups. When this formula is given the right-hand side is evaluated in `object`, converted to a factor if necessary, and the unique levels are used to define the groups. Defaults to `formula(object)`.
`level`	an optional positive integer giving the level of grouping to be used in an object with multiple nested grouping levels. Defaults to the highest or innermost level of grouping.
`groups`	an optional factor that will be used to split the rows into groups. Defaults to `getGroups(object, form, level)`.
`...`	optional additional arguments to the summary function `FUN`. Often it is helpful to specify `na.rm = TRUE`.

Value

Returns a data frame with as many rows as there are levels in the `groups` argument.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. sec. 3.4.

See Also

gsummary

Examples

```
## Find number of non-missing "conc" observations for each Subject
gapply( Phenobarb, FUN = function(x) sum(!is.na(x$conc)) )

# Pinheiro and Bates, p. 127
table( gapply(Quinidine, "conc", function(x) sum(!is.na(x))) )
changeRecords <- gapply( Quinidine, FUN = function(frm)
    any(is.na(frm[["conc"]]) & is.na(frm[["dose"]])) )
```

Gasoline *Refinery yield of gasoline*

Description

The Gasoline data frame has 32 rows and 6 columns.

Format

This data frame contains the following columns:

yield a numeric vector giving the percentage of crude oil converted to gasoline after distillation and fractionation

endpoint a numeric vector giving the temperature (degrees F) at which all the gasoline is vaporized

Sample an ordered factor giving the inferred crude oil sample number

API a numeric vector giving the crude oil gravity (degrees API)

vapor a numeric vector giving the vapor pressure of the crude oil $(\mathrm{lbf}/\mathrm{in}^2)$

ASTM a numeric vector giving the crude oil 10% point ASTM—the temperature at which 10% of the crude oil has become vapor.

Details

Prater (1955) provides data on crude oil properties and gasoline yields. Atkinson (1985) uses these data to illustrate the use of diagnostics in multiple regression analysis. Three of the covariates—API, vapor, and ASTM—measure characteristics of the crude oil used to produce the gasoline. The other covariate — endpoint—is a characteristic of the refining process. Daniel and Wood (1980) notice that the covariates characterizing the crude oil occur in only ten distinct groups and conclude that the data represent responses measured on ten different crude oil samples.

Source

Prater, N. H. (1955), Estimate gasoline yields from crudes, *Petroleum Refiner*, **35** (5).

Atkinson, A. C. (1985), *Plots, Transformations, and Regression*, Oxford Press, New York.

Daniel, C. and Wood, F. S. (1980), *Fitting Equations to Data*, Wiley, New York

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S (4th ed)*, Springer, New York.

getCovariate *Extract Covariate from an Object*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include `corStruct`, `corSpatial`, `data.frame`, and `varFunc`.

Usage

```
getCovariate(object, form, data)
```

Arguments

`object`	any object with a `covariate` component
`form`	an optional one-sided formula specifying the covariate(s) to be extracted. Defaults to `formula(object)`.
`data`	a data frame in which to evaluate the variables defined in `form`.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. p. 100.

See Also

`getCovariate.corStruct`, `getCovariate.data.frame`, `getCovariate.varFunc`, `getCovariateFormula`

getCovariate.corStruct *Extract corStruct Object Covariate*

Description

This method function extracts the covariate(s) associated with `object`.

Usage

```
## S3 method for class 'corStruct'
getCovariate(object, form, data)
```

Arguments

`object`	an object inheriting from class `corStruct` representing a correlation structure.
`form`	this argument is included to make the method function compatible with the generic. It will be assigned the value of `formula(object)` and should not be modified.
`data`	an optional data frame in which to evaluate the variables defined in `form`, in case `object` is not initialized and the covariate needs to be evaluated.

Value

when the correlation structure does not include a grouping factor, the returned value will be a vector or a matrix with the covariate(s) associated with `object`. If a grouping factor is present, the returned value will be a list of vectors or matrices with the covariate(s) corresponding to each grouping level. For spatial correlation structures, this extracts the *distances* implied by the covariates, and excludes 1-observation groups.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`getCovariate`

Examples

```
cs1 <- corAR1(form = ~ 1 | Subject)
getCovariate(cs1, data = Orthodont)
```

`getCovariate.data.frame` *Extract Data Frame Covariate*

Description

The right hand side of `form`, stripped of any conditioning expression (i.e. an expression following a `|` operator), is evaluated in `object`.

Usage

```
## S3 method for class 'data.frame'
getCovariate(object, form, data)
```

Arguments

`object`	an object inheriting from class `data.frame`.
`form`	an optional formula specifying the covariate to be evaluated in `object`. Defaults to `formula(object)`.
`data`	some methods for this generic require a separate data frame. Not used in this method.

Value

the value of the right hand side of `form`, stripped of any conditional expression, evaluated in `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`getCovariateFormula`

Examples

```
getCovariate(Orthodont)
```

`getCovariate.varFunc` *Extract varFunc Covariate*

Description

This method function extracts the covariate(s) associated with the variance function represented by `object`, if any is present.

Usage

```
## S3 method for class 'varFunc'
getCovariate(object, form, data)
```

Arguments

`object`	an object inheriting from class `varFunc`, representing a variance function structure.
`form`	an optional formula specifying the covariate to be evaluated in `object`. Defaults to `formula(object)`.
`data`	some methods for this generic require a `data` object. Not used in this method.

Value

if `object` has a `covariate` attribute, its value is returned; else `NULL` is returned.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`covariate<-.varFunc`

Examples

```
vf1 <- varPower(1.1, form = ~age)
covariate(vf1) <- Orthodont[["age"]]
getCovariate(vf1)
```

getCovariateFormula *Extract Covariates Formula*

Description

The right hand side of `formula(object)`, without any conditioning expressions (i.e. any expressions after a `|` operator) is returned as a one-sided formula.

Usage

```
getCovariateFormula(object)
```

Arguments

`object`	any object from which a formula can be extracted.

Value

a one-sided formula describing the covariates associated with `formula(object)`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`getCovariate`

Examples

```
getCovariateFormula(y ~ x | g)
getCovariateFormula(y ~ x)
```

getData *Extract Data from an Object*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include `gls`, `lme`, and `lmList`.

Usage

```
getData(object)
```

Arguments

object	an object from which a data frame can be extracted, generally a fitted model object.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

getData.gls, getData.lme, getData.lmList

---

getData.gls *Extract gls Object Data*

Description

If present in the calling sequence used to produce object, the data frame used to fit the model is obtained.

Usage

```
## S3 method for class 'gls'
getData(object)
```

Arguments

object	an object inheriting from class gls, representing a generalized least squares fitted linear model.

Value

if a data argument is present in the calling sequence that produced object, the corresponding data frame (with na.action and subset applied to it, if also present in the call that produced object) is returned; else, NULL is returned.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gls, getData

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), data = Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
getData(fm1)
```

`getData.lme` *Extract lme Object Data*

Description

If present in the calling sequence used to produce `object`, the data frame used to fit the model is obtained.

Usage

```
## S3 method for class 'lme'
getData(object)
```

Arguments

`object`	an object inheriting from class `lme`, representing a linear mixed-effects fitted model.

Value

if a `data` argument is present in the calling sequence that produced `object`, the corresponding data frame (with `na.action` and `subset` applied to it, if also present in the call that produced `object`) is returned; else, `NULL` is returned.

Note that as from version 3.1-102, this only omits rows omitted in the fit if `na.action` = `na.omit`, and does not omit at all if `na.action` = `na.exclude`. That is generally what is wanted for plotting, the main use of this function.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lme`, `getData`

Examples

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), data = Ovary,
           random = ~ sin(2*pi*Time))
getData(fm1)
```

getData.lmList *Extract lmList Object Data*

Description

If present in the calling sequence used to produce `object`, the data frame used to fit the model is obtained.

Usage

```
## S3 method for class 'lmList'
getData(object)
```

Arguments

`object`	an object inheriting from class `lmList`, representing a list of `lm` objects with a common model.

Value

if a `data` argument is present in the calling sequence that produced `object`, the corresponding data frame (with `na.action` and `subset` applied to it, if also present in the call that produced `object`) is returned; else, `NULL` is returned.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lmList`, `getData`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
getData(fm1)
```

getGroups *Extract Grouping Factors from an Object*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include `corStruct`, `data.frame`, `gls`, `lme`, `lmList`, and `varFunc`.

Usage

```
getGroups(object, form, level, data, sep)
```

Arguments

`object`	any object
`form`	an optional formula with a conditioning expression on its right hand side (i.e. an expression involving the `|` operator). Defaults to `formula(object)`.
`level`	a positive integer vector with the level(s) of grouping to be used when multiple nested levels of grouping are present. This argument is optional for most methods of this generic function and defaults to all levels of nesting.
`data`	a data frame in which to interpret the variables named in `form`. Optional for most methods.
`sep`	character, the separator to use between group levels when multiple levels are collapsed. The default is `'/'`.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 100, 461.

See Also

`getGroupsFormula`

`getGroups.corStruct`, `getGroups.data.frame`, `getGroups.gls`, `getGroups.lmList`, `getGroups.lme`, `getGroups.varFunc`

getGroups.corStruct *Extract corStruct Groups*

Description

This method function extracts the grouping factor associated with `object`, if any is present.

Usage

```
## S3 method for class 'corStruct'
getGroups(object, form, level, data, sep)
```

Arguments

`object`	an object inheriting from class `corStruct` representing a correlation structure.
`form`	this argument is included to make the method function compatible with the generic. It will be assigned the value of `formula(object)` and should not be modified.
`level`	this argument is included to make the method function compatible with the generic and is not used.
`data`	an optional data frame in which to evaluate the variables defined in `form`, in case `object` is not initialized and the grouping factor needs to be evaluated.
`sep`	character, the separator to use between group levels when multiple levels are collapsed. The default is `'/'`.

Value

if a grouping factor is present in the correlation structure represented by `object`, the function returns the corresponding factor vector; else the function returns `NULL`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`getGroups`

Examples

```
cs1 <- corAR1(form = ~ 1 | Subject)
getGroups(cs1, data = Orthodont)
```

getGroups.data.frame *Extract Groups from a Data Frame*

Description

Each variable named in the expression after the | operator on the right hand side of form is evaluated in object. If more than one variable is indicated in level they are combined into a data frame; else the selected variable is returned as a vector. When multiple grouping levels are defined in form and level > 1, the levels of the returned factor are obtained by pasting together the levels of the grouping factors of level greater or equal to level, to ensure their uniqueness.

Usage

```
## S3 method for class 'data.frame'
getGroups(object, form, level, data, sep)
```

Arguments

object	an object inheriting from class data.frame.
form	an optional formula with a conditioning expression on its right hand side (i.e. an expression involving the | operator). Defaults to formula(object).
level	a positive integer vector with the level(s) of grouping to be used when multiple nested levels of grouping are present. Defaults to all levels of nesting.
data	unused
sep	character, the separator to use between group levels when multiple levels are collapsed. The default is '/'.

Value

either a data frame with columns given by the grouping factors indicated in level, from outer to inner, or, when a single level is requested, a factor representing the selected grouping factor.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 100, 461.

See Also

getGroupsFormula

Examples

```
getGroups(Pixel)
getGroups(Pixel, level = 2)
```

`getGroups.gls` *Extract gls Object Groups*

Description

If present, the grouping factor associated to the correlation structure for the linear model represented by `object` is extracted.

Usage

```
## S3 method for class 'gls'
getGroups(object, form, level, data, sep)
```

Arguments

`object`	an object inheriting from class `gls`, representing a generalized least squares fitted linear model.
`form`	an optional formula with a conditioning expression on its right hand side (i.e. an expression involving the `|` operator). Defaults to `formula(object)`. Not used.
`level`	a positive integer vector with the level(s) of grouping to be used when multiple nested levels of grouping are present. This argument is optional for most methods of this generic function and defaults to all levels of nesting. Not used.
`data`	a data frame in which to interpret the variables named in `form`. Optional for most methods. Not used.
`sep`	character, the separator to use between group levels when multiple levels are collapsed. The default is `'/'`. Not used.

Value

if the linear model represented by `object` incorporates a correlation structure and the corresponding `corStruct` object has a grouping factor, a vector with the group values is returned; else, `NULL` is returned.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`gls`, `corClasses`

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
getGroups(fm1)
```

`getGroups.lme` *Extract lme Object Groups*

Description

The grouping factors corresponding to the linear mixed-effects model represented by `object` are extracted. If more than one level is indicated in `level`, the corresponding grouping factors are combined into a data frame; else the selected grouping factor is returned as a vector.

Usage

```
## S3 method for class 'lme'
getGroups(object, form, level, data, sep)
```

Arguments

`object`	an object inheriting from class `lme`, representing a fitted linear mixed-effects model.
`form`	this argument is included to make the method function compatible with the generic and is ignored in this method.
`level`	an optional integer vector giving the level(s) of grouping to be extracted from `object`. Defaults to the highest or innermost level of grouping.
`data`	unused
`sep`	character, the separator to use between group levels when multiple levels are collapsed. The default is `'/'`.

Value

either a data frame with columns given by the grouping factors indicated in `level`, or, when a single level is requested, a factor representing the selected grouping factor.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lme`

Examples

```
fm1 <- lme(pixel ~ day + day^2, Pixel,
  random = list(Dog = ~day, Side = ~1))
getGroups(fm1, level = 1:2)
```

---

getGroups.lmList *Extract lmList Object Groups*

Description

The grouping factor determining the partitioning of the observations used to produce the `lm` components of `object` is extracted.

Usage

```
## S3 method for class 'lmList'
getGroups(object, form, level, data, sep)
```

Arguments

`object`	an object inheriting from class `lmList`, representing a list of `lm` objects with a common model.
`form`	an optional formula with a conditioning expression on its right hand side (i.e. an expression involving the `|` operator). Defaults to `formula(object)`. Not used.
`level`	a positive integer vector with the level(s) of grouping to be used when multiple nested levels of grouping are present. This argument is optional for most methods of this generic function and defaults to all levels of nesting. Not used.
`data`	a data frame in which to interpret the variables named in `form`. Optional for most methods. Not used.
`sep`	character, the separator to use between group levels when multiple levels are collapsed. The default is `'/'`. Not used.

Value

a vector with the grouping factor corresponding to the `lm` components of `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lmList`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
getGroups(fm1)
```

`getGroups.varFunc` *Extract varFunc Groups*

Description

This method function extracts the grouping factor associated with the variance function represented by `object`, if any is present.

Usage

```
## S3 method for class 'varFunc'
getGroups(object, form, level, data, sep)
```

Arguments

`object`	an object inheriting from class `varFunc`, representing a variance function structure.
`form`	an optional formula with a conditioning expression on its right hand side (i.e. an expression involving the `|` operator). Defaults to `formula(object)`. Not used.
`level`	a positive integer vector with the level(s) of grouping to be used when multiple nested levels of grouping are present. This argument is optional for most methods of this generic function and defaults to all levels of nesting. Not used.
`data`	a data frame in which to interpret the variables named in `form`. Optional for most methods. Not used.
`sep`	character, the separator to use between group levels when multiple levels are collapsed. The default is `'/'`. Not used.

Value

if `object` has a `groups` attribute, its value is returned; else `NULL` is returned.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

Examples

```
vf1 <- varPower(form = ~ age | Sex)
vf1 <- Initialize(vf1, Orthodont)
getGroups(vf1)
```

`getGroupsFormula` *Extract Grouping Formula*

Description

The conditioning expression associated with `formula(object)` (i.e. the expression after the `|` operator) is returned either as a named list of one-sided formulas, or a single one-sided formula, depending on the value of `asList`. The components of the returned list are ordered from outermost to innermost level and are named after the grouping factor expression.

Usage

```
getGroupsFormula(object, asList, sep)
```

Arguments

`object`	any object from which a formula can be extracted.
`asList`	an optional logical value. If `TRUE` the returned value with be a list of formulas; else, if `FALSE` the returned value will be a one-sided formula. Defaults to `FALSE`.
`sep`	character, the separator to use between group levels when multiple levels are collapsed. The default is `'/'`.

Value

a one-sided formula, or a list of one-sided formulas, with the grouping structure associated with `formula(object)`. If no conditioning expression is present in `formula(object)` a `NULL` value is returned.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`getGroupsFormula.gls`, `getGroupsFormula.lmList`, `getGroupsFormula.lme`, `getGroupsFormula.reStruct`, `getGroups`

Examples

```
getGroupsFormula(y ~ x | g1/g2)
```

getResponse *Extract Response Variable from an Object*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include `data.frame`, `gls`, `lme`, and `lmList`.

Usage

```
getResponse(object, form)
```

Arguments

`object`	any object.
`form`	an optional two-sided formula. Defaults to `formula(object)`.

Value

For the `data.frame` method, the result of evaluating the left-hand side of `form` in `object`.

For `gls`, `lme`, and `lmList`, the sum of the `fitted` values and the `residuals`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`getResponseFormula`

Examples

```
getResponse(Orthodont)
```

getResponseFormula *Extract Formula Specifying Response Variable*

Description

The left hand side of `formula{object}` is returned as a one-sided formula.

Usage

```
getResponseFormula(object)
```

Arguments

object	any object from which a formula can be extracted.

Value

a one-sided formula with the response variable associated with formula{object}.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

getResponse

Examples

```
getResponseFormula(y ~ x | g)
```

---

getVarCov *Extract variance-covariance matrix*

Description

Extract the variance-covariance matrix from a fitted model, such as a mixed-effects model.

Usage

```
getVarCov(obj, ...)
## S3 method for class 'lme'
getVarCov(obj, individuals,
    type = c("random.effects", "conditional", "marginal"), ...)
## S3 method for class 'gls'
getVarCov(obj, individual = 1, ...)
```

Arguments

obj	A fitted model. Methods are available for models fit by lme and by gls
individuals	For models fit by lme a vector of levels of the grouping factor can be specified for the conditional or marginal variance-covariance matrices.
individual	For models fit by gls the only type of variance-covariance matrix provided is the marginal variance-covariance of the responses by group. The optional argument individual specifies the group of responses.
type	For models fit by lme the type argument specifies the type of variance-covariance matrix, either "random.effects" for the random-effects variance-covariance (the default), or "conditional" for the conditional. variance-covariance of the responses or "marginal" for the the marginal variance-covariance of the responses.
...	Optional arguments for some methods, as described above

Value

A variance-covariance matrix or a list of variance-covariance matrices.

Author(s)

Mary Lindstrom <lindstro@biostat.wisc.edu>

See Also

lme, gls

Examples

```
fm1 <- lme(distance ~ age, data = Orthodont, subset = Sex == "Female")
getVarCov(fm1)
getVarCov(fm1, individuals = "F01", type = "marginal")
getVarCov(fm1, type = "conditional")
fm2 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
getVarCov(fm2)
```

---

gls *Fit Linear Model Using Generalized Least Squares*

Description

This function fits a linear model using generalized least squares. The errors are allowed to be correlated and/or have unequal variances.

Usage

```
gls(model, data, correlation, weights, subset, method, na.action,
    control, verbose)
## S3 method for class 'gls'
update(object, model., ..., evaluate = TRUE)
```

Arguments

object	an object inheriting from class "gls", representing a generalized least squares fitted linear model.
model	a two-sided linear formula object describing the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
model.	Changes to the model – see update.formula for details.
data	an optional data frame containing the variables named in model, correlation, weights, and subset. By default the variables are taken from the environment from which gls is called.

correlation	an optional corStruct object describing the within-group correlation structure. See the documentation of corClasses for a description of the available corStruct classes. If a grouping variable is to be used, it must be specified in the form argument to the corStruct constructor. Defaults to NULL, corresponding to uncorrelated errors.
weights	an optional varFunc object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to varFixed, corresponding to fixed variance weights. See the documentation on varClasses for a description of the available varFunc classes. Defaults to NULL, corresponding to homoscedastic errors.
subset	an optional expression indicating which subset of the rows of data should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
method	a character string. If "REML" the model is fit by maximizing the restricted log-likelihood. If "ML" the log-likelihood is maximized. Defaults to "REML".
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes gls to print an error message and terminate if there are any incomplete observations.
control	a list of control values for the estimation algorithm to replace the default values returned by the function glsControl. Defaults to an empty list.
verbose	an optional logical value. If TRUE information on the evolution of the iterative algorithm is printed. Default is FALSE.
...	some methods for this generic require additional arguments. None are used in this method.
evaluate	If TRUE evaluate the new call else return the call.

Details

offset terms in model are an error since 3.1-157 (2022-03): previously they were silently ignored.

Value

an object of class "gls" representing the linear model fit. Generic functions such as print, plot, and summary have methods to show the results of the fit. See glsObject for the components of the fit. The functions resid, coef and fitted, can be used to extract some of its components.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

The different correlation structures available for the correlation argument are described in Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994), Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996), and Venables, W.N. and Ripley, B.D. (2002). The use of variance functions

for linear and nonlinear models is presented in detail in Carroll, R.J. and Ruppert, D. (1988) and Davidian, M. and Giltinan, D.M. (1995).

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Carroll, R.J. and Ruppert, D. (1988) "Transformation and Weighting in Regression", Chapman and Hall.

Davidian, M. and Giltinan, D.M. (1995) "Nonlinear Mixed Effects Models for Repeated Measurement Data", Chapman and Hall.

Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 100, 461.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

See Also

corClasses, glsControl, glsObject, glsStruct, plot.gls, predict.gls, qqnorm.gls, residuals.gls, summary.gls, varClasses, varFunc

Examples

```
# AR(1) errors within each Mare
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
# variance increases as a power of the absolute fitted values
fm2 <- update(fm1, weights = varPower())
```

---

glsControl *Control Values for gls Fit*

Description

The values supplied in the function call replace the defaults and a list with all possible arguments is returned. The returned list is used as the `control` argument to the `gls` function.

Usage

```
glsControl(maxIter, msMaxIter, tolerance, msTol, msVerbose,
           singular.ok, returnObject = FALSE, apVar, .relStep,
           opt = c("nlminb", "optim"), optimMethod,
           minAbsParApVar, natural, sigma = NULL)
```

Arguments

maxIter	maximum number of iterations for the gls optimization algorithm. Default is 50.
msMaxIter	maximum number of iterations for the optimization step inside the gls optimization. Default is 50.
tolerance	tolerance for the convergence criterion in the gls algorithm. Default is 1e-6.
msTol	tolerance for the convergence criterion of the first outer iteration when optim is used. Default is 1e-7.
msVerbose	a logical value passed as the trace control value to the chosen optimizer (see documentation on that function). Default is FALSE.
singular.ok	a logical value indicating whether non-estimable coefficients (resulting from linear dependencies among the columns of the regression matrix) should be allowed. Default is FALSE.
returnObject	a logical value indicating whether the fitted object should be returned when the maximum number of iterations is reached without convergence of the algorithm. Default is FALSE.
apVar	a logical value indicating whether the approximate covariance matrix of the variance-covariance parameters should be calculated. Default is TRUE.
.relStep	relative step for numerical derivatives calculations. Default is .Machine$double.eps^(1/3).
opt	the optimizer to be used, either "nlminb" (the current default) or "optim" (the previous default).
optimMethod	character - the optimization method to be used with the optim optimizer. The default is "BFGS". An alternative is "L-BFGS-B".
minAbsParApVar	numeric value - minimum absolute parameter value in the approximate variance calculation. The default is 0.05.
natural	logical. Should the natural parameterization be used for the approximate variance calculations? Default is TRUE.
sigma	optionally a positive number to fix the residual error at. If NULL, as by default, or 0, sigma is estimated.

Value

a list with components for each of the possible arguments.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>; the sigma option: Siem Heisterkamp and Bert van Willigen.

See Also

gls

Examples

```
# decrease the maximum number of iterations and request tracing
glsControl(msMaxIter = 20, msVerbose = TRUE)
```

`glsObject` *Fitted gls Object*

Description

An object returned by the `gls` function, inheriting from class `"gls"` and representing a generalized least squares fitted linear model. Objects of this class have methods for the generic functions `anova`, `coef`, `fitted`, `formula`, `getGroups`, `getResponse`, `intervals`, `logLik`, `plot`, `predict`, `print`, `residuals`, `summary`, and `update`.

Value

The following components must be included in a legitimate `"gls"` object.

`apVar`	an approximate covariance matrix for the variance-covariance coefficients. If `apVar = FALSE` in the list of control values used in the call to `gls`, this component is equal to `NULL`.
`call`	a list containing an image of the `gls` call that produced the object.
`coefficients`	a vector with the estimated linear model coefficients.
`contrasts`	a list of the contrast matrices used to represent factors in the model formula. This information is important for making predictions from a new data frame in which not all levels of the original factors are observed. If no factors are used in the model, this component will be an empty list.
`dims`	a list with basic dimensions used in the model fit, including the components `N` - the number of observations in the data and `p` - the number of coefficients in the linear model.
`fitted`	a vector with the fitted values.
`modelStruct`	an object inheriting from class `glsStruct`, representing a list of linear model components, such as `corStruct` and `varFunc` objects.
`groups`	the correlation structure grouping factor, if any is present.
`logLik`	the log-likelihood at convergence.
`method`	the estimation method: either `"ML"` for maximum likelihood, or `"REML"` for restricted maximum likelihood.
`numIter`	the number of iterations used in the iterative algorithm.
`residuals`	a vector with the residuals.
`sigma`	the estimated residual standard error.
`varBeta`	an approximate covariance matrix of the coefficients estimates.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gls, glsStruct

---

glsStruct *Generalized Least Squares Structure*

Description

A generalized least squares structure is a list of model components representing different sets of parameters in the linear model. A glsStruct may contain corStruct and varFunc objects. NULL arguments are not included in the glsStruct list.

Usage

```
glsStruct(corStruct, varStruct)
```

Arguments

corStruct	an optional corStruct object, representing a correlation structure. Default is NULL.
varStruct	an optional varFunc object, representing a variance function structure. Default is NULL.

Value

a list of model variance-covariance components determining the parameters to be estimated for the associated linear model.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

corClasses, gls, residuals.glsStruct, varFunc

Examples

```
gls1 <- glsStruct(corAR1(), varPower())
```

Glucose *Glucose levels over time*

Description

The `Glucose` data frame has 378 rows and 4 columns.

Format

This data frame contains the following columns:

Subject an ordered factor with levels `6` < `2` < `3` < `5` < `1` < `4`

Time a numeric vector

conc a numeric vector of glucose levels

Meal an ordered factor with levels `2am` < `6am` < `10am` < `2pm` < `6pm` < `10pm`

Source

Hand, D. and Crowder, M. (1996), *Practical Longitudinal Data Analysis*, Chapman and Hall, London.

Glucose2 *Glucose Levels Following Alcohol Ingestion*

Description

The `Glucose2` data frame has 196 rows and 4 columns.

Format

This data frame contains the following columns:

Subject a factor with levels `1` to `7` identifying the subject whose glucose level is measured.

Date a factor with levels `1` `2` indicating the occasion in which the experiment was conducted.

Time a numeric vector giving the time since alcohol ingestion (in min/10).

glucose a numeric vector giving the blood glucose level (in mg/dl).

Details

Hand and Crowder (Table A.14, pp. 180-181, 1996) describe data on the blood glucose levels measured at 14 time points over 5 hours for 7 volunteers who took alcohol at time 0. The same experiment was repeated on a second date with the same subjects but with a dietary additive used for all subjects.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.10)

Hand, D. and Crowder, M. (1996), *Practical Longitudinal Data Analysis*, Chapman and Hall, London.

`gnls` *Fit Nonlinear Model Using Generalized Least Squares*

Description

This function fits a nonlinear model using generalized least squares. The errors are allowed to be correlated and/or have unequal variances.

Usage

```
gnls(model, data, params, start, correlation, weights, subset,
     na.action, naPattern, control, verbose)
```

Arguments

`model`	a two-sided formula object describing the model, with the response on the left of a `~` operator and a nonlinear expression involving parameters and covariates on the right. If `data` is given, all names used in the formula should be defined as parameters or variables in the data frame.
`data`	an optional data frame containing the variables named in `model`, `correlation`, `weights`, `subset`, and `naPattern`. By default the variables are taken from the environment from which `gnls` is called.
`params`	an optional two-sided linear formula of the form `p1+...+pn~x1+...+xm`, or list of two-sided formulas of the form `p1~x1+...+xm`, with possibly different models for each parameter. The `p1,...,pn` represent parameters included on the right hand side of `model` and `x1+...+xm` define a linear model for the parameters (when the left hand side of the formula contains several parameters, they are all assumed to follow the same linear model described by the right hand side expression). A `1` on the right hand side of the formula(s) indicates a single fixed effects for the corresponding parameter(s). By default, the parameters are obtained from the names of `start`.
`start`	an optional named list, or numeric vector, with the initial values for the parameters in `model`. It can be omitted when a `"selfStart"` model is used, in which case the starting estimates will be obtained from a single call to the `nls` function.
`correlation`	an optional `corStruct` object describing the within-group correlation structure. See the documentation of `corClasses` for a description of the available `corStruct` classes. If a grouping variable is to be used, it must be specified in the `form` argument to the `corStruct` constructor. Defaults to `NULL`, corresponding to uncorrelated errors.

weights	an optional varFunc object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to varFixed, corresponding to fixed variance weights. See the documentation on varClasses for a description of the available varFunc classes. Defaults to NULL, corresponding to homoscedastic errors.
subset	an optional expression indicating which subset of the rows of data should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes gnls to print an error message and terminate if there are any incomplete observations.
naPattern	an expression or formula object, specifying which returned values are to be regarded as missing.
control	a list of control values for the estimation algorithm to replace the default values returned by the function gnlsControl. Defaults to an empty list.
verbose	an optional logical value. If TRUE information on the evolution of the iterative algorithm is printed. Default is FALSE.

Value

an object of class gnls, also inheriting from class gls, representing the nonlinear model fit. Generic functions such as print, plot and summary have methods to show the results of the fit. See gnlsObject for the components of the fit. The functions resid, coef, and fitted can be used to extract some of its components.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

The different correlation structures available for the correlation argument are described in Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994), Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996), and Venables, W.N. and Ripley, B.D. (2002). The use of variance functions for linear and nonlinear models is presented in detail in Carrol, R.J. and Rupert, D. (1988) and Davidian, M. and Giltinan, D.M. (1995).

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Carrol, R.J. and Rupert, D. (1988) "Transformation and Weighting in Regression", Chapman and Hall.

Davidian, M. and Giltinan, D.M. (1995) "Nonlinear Mixed Effects Models for Repeated Measurement Data", Chapman and Hall.

Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996) "SAS Systems for Mixed Models", SAS Institute.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

corClasses, gnlsControl, gnlsObject, gnlsStruct, predict.gnls, varClasses, varFunc

Examples

```
# variance increases with a power of the absolute fitted values
fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            weights = varPower())
summary(fm1)
```

---

gnlsControl *Control Values for gnls Fit*

Description

The values supplied in the function call replace the defaults and a list with all possible arguments is returned. The returned list is used as the `control` argument to the `gnls` function.

Usage

```
gnlsControl(maxIter = 50, nlsMaxIter = 7, msMaxIter = 50, minScale = 0.001,
            tolerance = 1e-6, nlsTol = 0.001, msTol = 1e-7,
            returnObject = FALSE, msVerbose = FALSE,
            apVar = TRUE, .relStep =,
            opt = c("nlminb", "optim"), optimMethod = "BFGS",
            minAbsParApVar = 0.05, sigma = NULL)
```

Arguments

`maxIter`	maximum number of iterations for the `gnls` optimization algorithm. Default is 50.
`nlsMaxIter`	maximum number of iterations for the `nls` optimization step *inside* the `gnls` optimization. Default is 7.
`msMaxIter`	maximum number of iterations for the `optimization` step inside the `gnls` optimization. Default is 50.
`minScale`	minimum factor by which to shrink the default step size in an attempt to decrease the sum of squares in the `nls` step. Default 0.001.
`tolerance`	tolerance for the convergence criterion in the `gnls` algorithm. Default is 1e-6.
`nlsTol`	tolerance for the convergence criterion in `nls` step. Default is 1e-3.
`msTol`	tolerance for the convergence criterion of the first outer iteration when `optim` is used. Default is 1e-7.

returnObject	a logical value indicating whether the fitted object should be returned with a warning (instead of an error via stop()) when the maximum number of iterations is reached without convergence of the algorithm.
msVerbose	a logical value passed as the trace argument to the optimizer chosen by opt; see documentation on that. Default is FALSE.
apVar	a logical value indicating whether the approximate covariance matrix of the variance-covariance parameters should be calculated. Default is TRUE.
.relStep	relative step for numerical derivatives calculations. Default is .Machine$double.eps^(1/3) (about 6e-6).
opt	the optimizer to be used, either "nlminb" (the current default) or "optim" (the previous default).
optimMethod	character - the optimization method to be used with the optim optimizer. The default is "BFGS". An alternative is "L-BFGS-B".
minAbsParApVar	numeric value - minimum absolute parameter value in the approximate variance calculation. The default is 0.05.
sigma	optionally a positive number to fix the residual error at. If NULL, as by default, or 0, sigma is estimated.

Value

a list with components for each of the possible arguments.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>; the sigma option: Siem Heisterkamp and Bert van Willigen.

See Also

gnls

Examples

```
# decrease the maximum number of iterations and request tracing
gnlsControl(msMaxIter = 20, msVerbose = TRUE)
```

---

gnlsObject *Fitted gnls Object*

Description

An object returned by the gnls function, inheriting from class "gnls" and also from class "gls", and representing a generalized nonlinear least squares fitted model. Objects of this class have methods for the generic functions anova, coef, fitted, formula, getGroups, getResponse, intervals, logLik, plot, predict, print, residuals, summary, and update.

Value

The following components must be included in a legitimate "gnls" object.

apVar	an approximate covariance matrix for the variance-covariance coefficients. If apVar = FALSE in the control values used in the call to gnls, this component is equal to NULL.
call	a list containing an image of the gnls call that produced the object.
coefficients	a vector with the estimated nonlinear model coefficients.
contrasts	a list of the contrast matrices used to represent factors in the model formula. This information is important for making predictions from a new data frame in which not all levels of the original factors are observed. If no factors are used in the model, this component will be an empty list.
dims	a list with basic dimensions used in the model fit, including the components N - the number of observations used in the fit and p - the number of coefficients in the nonlinear model.
fitted	a vector with the fitted values.
modelStruct	an object inheriting from class gnlsStruct, representing a list of model components, such as corStruct and varFunc objects.
groups	a vector with the correlation structure grouping factor, if any is present.
logLik	the log-likelihood at convergence.
numIter	the number of iterations used in the iterative algorithm.
plist	
pmap	
residuals	a vector with the residuals.
sigma	the estimated residual standard error.
varBeta	an approximate covariance matrix of the coefficients estimates.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gnls, gnlsStruct

gnlsStruct *Generalized Nonlinear Least Squares Structure*

Description

A generalized nonlinear least squares structure is a list of model components representing different sets of parameters in the nonlinear model. A gnlsStruct may contain corStruct and varFunc objects. NULL arguments are not included in the gnlsStruct list.

Usage

```
gnlsStruct(corStruct, varStruct)
```

Arguments

corStruct	an optional corStruct object, representing a correlation structure. Default is NULL.
varStruct	an optional varFunc object, representing a variance function structure. Default is NULL.

Value

a list of model variance-covariance components determining the parameters to be estimated for the associated nonlinear model.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gnls, corClasses, residuals.gnlsStruct varFunc

Examples

```
gnls1 <- gnlsStruct(corAR1(), varPower())
```

groupedData *Construct a groupedData Object*

Description

An object of the groupedData class is constructed from the formula and data by attaching the formula as an attribute of the data, along with any of outer, inner, labels, and units that are given. If order.groups is TRUE the grouping factor is converted to an ordered factor with the ordering determined by FUN. Depending on the number of grouping levels and the type of primary covariate, the returned object will be of one of three classes: nfnGroupedData - numeric covariate, single level of nesting; nffGroupedData - factor covariate, single level of nesting; and nmGroupedData - multiple levels of nesting. Several modeling and plotting functions can use the formula stored with a groupedData object to construct default plots and models.

Usage

```
groupedData(formula, data, order.groups, FUN, outer, inner,
            labels, units)

## S3 method for class 'groupedData'
update(object, formula, data, order.groups, FUN,
       outer, inner, labels, units, ...)
```

Arguments

object	an object inheriting from class groupedData.
formula	a formula of the form resp ~ cov | group where resp is the response, cov is the primary covariate, and group is the grouping factor. The expression 1 can be used for the primary covariate when there is no other suitable candidate. Multiple nested grouping factors can be listed separated by the / symbol as in fact1/fact2. In an expression like this the fact2 factor is nested within the fact1 factor.
data	a data frame in which the expressions in formula can be evaluated. The resulting groupedData object will consist of the same data values in the same order but with additional attributes.
order.groups	an optional logical value, or list of logical values, indicating if the grouping factors should be converted to ordered factors according to the function FUN applied to the response from each group. If multiple levels of grouping are present, this argument can be either a single logical value (which will be repeated for all grouping levels) or a list of logical values. If no names are assigned to the list elements, they are assumed in the same order as the group levels (outermost to innermost grouping). Ordering within a level of grouping is done within the levels of the grouping factors which are outer to it. Changing the grouping factor to an ordered factor does not affect the ordering of the rows in the data frame but it does affect the order of the panels in a trellis display of the data or models fitted to the data. Defaults to TRUE.

FUN — an optional summary function that will be applied to the values of the response for each level of the grouping factor, when order.groups = TRUE, to determine the ordering. Defaults to the max function.

outer — an optional one-sided formula, or list of one-sided formulas, indicating covariates that are outer to the grouping factor(s). If multiple levels of grouping are present, this argument can be either a single one-sided formula, or a list of one-sided formulas. If no names are assigned to the list elements, they are assumed in the same order as the group levels (outermost to innermost grouping). An outer covariate is invariant within the sets of rows defined by the grouping factor. Ordering of the groups is done in such a way as to preserve adjacency of groups with the same value of the outer variables. When plotting a groupedData object, the argument outer = TRUE causes the panels to be determined by the outer formula. The points within the panels are associated by level of the grouping factor. Defaults to NULL, meaning that no outer covariates are present.

inner — an optional one-sided formula, or list of one-sided formulas, indicating covariates that are inner to the grouping factor(s). If multiple levels of grouping are present, this argument can be either a single one-sided formula, or a list of one-sided formulas. If no names are assigned to the list elements, they are assumed in the same order as the group levels (outermost to innermost grouping). An inner covariate can change within the sets of rows defined by the grouping factor. An inner formula can be used to associate points in a plot of a groupedData object. Defaults to NULL, meaning that no inner covariates are present.

labels — an optional list of character strings giving labels for the response and the primary covariate. The label for the primary covariate is named x and that for the response is named y. Either label can be omitted.

units — an optional list of character strings giving the units for the response and the primary covariate. The units string for the primary covariate is named x and that for the response is named y. Either units string can be omitted.

... — some methods for this generic require additional arguments. None are used in this method.

Value

an object of one of the classes nfnGroupedData, nffGroupedData, or nmGroupedData, and also inheriting from classes groupedData and data.frame.

Author(s)

Douglas Bates and José Pinheiro

References

Bates, D.M. and Pinheiro, J.C. (1997), "Software Design for Longitudinal Data Analysis", in "Modelling Longitudinal and Spatially Correlated Data: Methods, Applications and Future Directions", T.G. Gregoire (ed.), Springer-Verlag, New York. doi:10.1007/9781461206996_4

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

formula, gapply, gsummary, lme, plot.nffGroupedData, plot.nfnGroupedData, plot.nmGroupedData, reStruct

Examples

```
Orth.new <-  # create a new copy of the groupedData object
  groupedData( distance ~ age | Subject,
               data = as.data.frame( Orthodont ),
               FUN = mean,
               outer = ~ Sex,
               labels = list( x = "Age",
                 y = "Distance from pituitary to pterygomaxillary fissure" ),
               units = list( x = "(yr)", y = "(mm)") )
plot( Orth.new )         # trellis plot by Subject
formula( Orth.new )      # extractor for the formula
gsummary( Orth.new )     # apply summary by Subject
fm1 <- lme( Orth.new )   # fixed and groups formulae extracted from object
Orthodont2 <- update(Orthodont, FUN = mean)
```

---

gsummary *Summarize by Groups*

Description

Provide a summary of the variables in a data frame by groups of rows. This is most useful with a groupedData object to examine the variables by group.

Usage

```
gsummary(object, FUN, omitGroupingFactor, form, level,
   groups, invariantsOnly, ...)
```

Arguments

object	an object to be summarized - usually a groupedData object or a data.frame.
FUN	an optional summary function or a list of summary functions to be applied to each variable in the frame. The function or functions are applied only to variables in object that vary within the groups defined by groups. Invariant variables are always summarized by group using the unique value that they assume within that group. If FUN is a single function it will be applied to each non-invariant variable by group to produce the summary for that variable. If FUN is a list of functions, the names in the list should designate classes of variables in the frame such as ordered, factor, or numeric. The indicated function will be applied to any non-invariant variables of that class. The default functions to be used are mean for numeric factors, and Mode for both factor and ordered. The Mode function, defined internally in gsummary, returns the modal or most popular value of the variable. It is different from the mode function that returns the S-language mode of the variable.

omitGroupingFactor
: an optional logical value. When TRUE the grouping factor itself will be omitted from the group-wise summary but the levels of the grouping factor will continue to be used as the row names for the data frame that is produced by the summary. Defaults to FALSE.

form
: an optional one-sided formula that defines the groups. When this formula is given, the right-hand side is evaluated in object, converted to a factor if necessary, and the unique levels are used to define the groups. Defaults to formula(object).

level
: an optional positive integer giving the level of grouping to be used in an object with multiple nested grouping levels. Defaults to the highest or innermost level of grouping.

groups
: an optional factor that will be used to split the rows into groups. Defaults to getGroups(object, form, level).

invariantsOnly
: an optional logical value. When TRUE only those covariates that are invariant within each group will be summarized. The summary value for the group is always the unique value taken on by that covariate within the group. The columns in the summary are of the same class as the corresponding columns in object. By definition, the grouping factor itself must be an invariant. When combined with omitGroupingFactor = TRUE, this option can be used to discover is there are invariant covariates in the data frame. Defaults to FALSE.

...
: optional additional arguments to the summary functions that are invoked on the variables by group. Often it is helpful to specify na.rm = TRUE.

Value

A data.frame with one row for each level of the grouping factor. The number of columns is at most the number of columns in object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

summary, groupedData, getGroups

Examples

```
gsummary(Orthodont)  # default summary by Subject
## gsummary with invariantsOnly = TRUE and omitGroupingFactor = TRUE
## determines whether there are covariates like Sex that are invariant
## within the repeated observations on the same Subject.
gsummary(Orthodont, invariantsOnly = TRUE, omitGroupingFactor = TRUE)
```

Gun — *Methods for firing naval guns*

Description

The Gun data frame has 36 rows and 4 columns.

Format

This data frame contains the following columns:

rounds a numeric vector

Method a factor with levels M1 M2

Team an ordered factor with levels T1S < T3S < T2S < T1A < T2A < T3A < T1H < T3H < T2H

Physique an ordered factor with levels Slight < Average < Heavy

Details

Hicks (p.180, 1993) reports data from an experiment on methods for firing naval guns. Gunners of three different physiques (slight, average, and heavy) tested two firing methods. Both methods were tested twice by each of nine teams of three gunners with identical physique. The response was the number of rounds fired per minute.

Source

Hicks, C. R. (1993), *Fundamental Concepts in the Design of Experiments (4th ed)*, Harcourt Brace, New York.

IGF — *Radioimmunoassay of IGF-I Protein*

Description

The IGF data frame has 237 rows and 3 columns.

Format

This data frame contains the following columns:

Lot an ordered factor giving the radioactive tracer lot.

age a numeric vector giving the age (in days) of the radioactive tracer.

conc a numeric vector giving the estimated concentration of IGF-I protein (ng/ml)

Details

Davidian and Giltinan (1995) describe data obtained during quality control radioimmunoassays for ten different lots of radioactive tracer used to calibrate the Insulin-like Growth Factor (IGF-I) protein concentration measurements.

Source

Davidian, M. and Giltinan, D. M. (1995), *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, London.

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.11)

Initialize *Initialize Object*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: `corStruct`, `glsStruct`, `lmeStruct`, `reStruct`, and `varFunc`.

Usage

```
Initialize(object, data, ...)
```

Arguments

`object`	any object requiring initialization, e.g. "plug-in" structures such as `corStruct` and `varFunc` objects.
`data`	a data frame to be used in the initialization procedure.
`...`	some methods for this generic function require additional arguments.

Value

an initialized object with the same class as `object`. Changes introduced by the initialization procedure will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

Initialize.corStruct, Initialize.glsStruct, Initialize.lmeStruct, Initialize.reStruct, Initialize.varFunc, isInitialized

---

Initialize.corStruct *Initialize corStruct Object*

Description

This method initializes object by evaluating its associated covariate(s) and grouping factor, if any is present, in data, calculating various dimensions and constants used by optimization algorithms involving corStruct objects (see the appropriate Dim method documentation), and assigning initial values for the coefficients in object, if none were present.

Usage

```
## S3 method for class 'corStruct'
Initialize(object, data, ...)
```

Arguments

object	an object inheriting from class "corStruct" representing a correlation structure.
data	a data frame in which to evaluate the variables defined in formula(object).
...	this argument is included to make this method compatible with the generic.

Value

an initialized object with the same class as object representing a correlation structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

Dim.corStruct

Examples

```
cs1 <- corAR1(form = ~ 1 | Subject)
cs1 <- Initialize(cs1, data = Orthodont)
```

Initialize.glsStruct *Initialize a glsStruct Object*

Description

The individual linear model components of the glsStruct list are initialized.

Usage

```
## S3 method for class 'glsStruct'
Initialize(object, data, control, ...)
```

Arguments

object	an object inheriting from class "glsStruct", representing a list of linear model components, such as corStruct and varFunc objects.
data	a data frame in which to evaluate the variables defined in formula(object).
control	an optional list with control parameters for the initialization and optimization algorithms used in gls. Defaults to list(singular.ok = FALSE), implying that linear dependencies are not allowed in the model.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a glsStruct object similar to object, but with initialized model components.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gls, Initialize.corStruct, Initialize.varFunc, Initialize

Initialize.lmeStruct *Initialize an lmeStruct Object*

Description

The individual linear mixed-effects model components of the lmeStruct list are initialized.

Usage

```
## S3 method for class 'lmeStruct'
Initialize(object, data, groups, conLin, control, ...)
```

Arguments

object	an object inheriting from class "lmeStruct", representing a list of linear mixed-effects model components, such as reStruct, corStruct, and varFunc objects.
data	a data frame in which to evaluate the variables defined in formula(object).
groups	a data frame with the grouping factors corresponding to the lme model associated with object as columns, sorted from innermost to outermost grouping level.
conLin	an optional condensed linear model object, consisting of a list with components "Xy", corresponding to a regression matrix (X) combined with a response vector (y), and "logLik", corresponding to the log-likelihood of the underlying lme model. Defaults to attr(object, "conLin").
control	an optional list with control parameters for the initialization and optimization algorithms used in lme. Defaults to list(niterEM=20, gradHess=TRUE), implying that 20 EM iterations are to be used in the derivation of initial estimates for the coefficients of the reStruct component of object and, if possible, numerical gradient vectors and Hessian matrices for the log-likelihood function are to be used in the optimization algorithm.
...	some methods for this generic require additional arguments. None are used in this method.

Value

an lmeStruct object similar to object, but with initialized model components.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lme, Initialize.reStruct, Initialize.corStruct, Initialize.varFunc, Initialize

---

Initialize.reStruct *Initialize reStruct Object*

Description

Initial estimates for the parameters in the pdMat objects forming object, which have not yet been initialized, are obtained using the methodology described in Bates and Pinheiro (1998). These estimates may be refined using a series of EM iterations, as described in Bates and Pinheiro (1998). The number of EM iterations to be used is defined in control.

Usage

```
## S3 method for class 'reStruct'
Initialize(object, data, conLin, control, ...)
```

Arguments

object	an object inheriting from class "reStruct", representing a random effects structure and consisting of a list of pdMat objects.
data	a data frame in which to evaluate the variables defined in formula(object).
conLin	a condensed linear model object, consisting of a list with components "Xy", corresponding to a regression matrix (X) combined with a response vector (y), and "logLik", corresponding to the log-likelihood of the underlying model.
control	an optional list with a single component niterEM controlling the number of iterations for the EM algorithm used to refine initial parameter estimates. It is given as a list for compatibility with other Initialize methods. Defaults to list(niterEM = 20).
...	some methods for this generic require additional arguments. None are used in this method.

Value

an reStruct object similar to object, but with all pdMat components initialized.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

reStruct, pdMat, Initialize

Initialize.varFunc *Initialize varFunc Object*

Description

This method initializes object by evaluating its associated covariate(s) and grouping factor, if any is present, in data; determining if the covariate(s) need to be updated when the values of the coefficients associated with object change; initializing the log-likelihood and the weights associated with object; and assigning initial values for the coefficients in object, if none were present. The covariate(s) will only be initialized if no update is needed when coef(object) changes.

Usage

```
## S3 method for class 'varFunc'
Initialize(object, data, ...)
```

Arguments

object	an object inheriting from class "varFunc", representing a variance function structure.
data	a data frame in which to evaluate the variables named in formula(object).
...	this argument is included to make this method compatible with the generic.

Value

an initialized object with the same class as object representing a variance function structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Initialize

Examples

```
vf1 <- varPower( form = ~ age | Sex )
vf1 <- Initialize( vf1, Orthodont )
```

---

intervals *Confidence Intervals on Coefficients*

Description

Confidence intervals on the parameters associated with the model represented by object are obtained. This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: gls, lme, and lmList.

Usage

```
intervals(object, level, ...)
```

Arguments

object	a fitted model object from which parameter estimates can be extracted.
level	an optional numeric value for the interval confidence level. Defaults to 0.95.
...	some methods for the generic may require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

intervals.lme, intervals.lmList, intervals.gls

---

intervals.gls *Confidence Intervals on gls Parameters*

Description

Approximate confidence intervals for the parameters in the linear model represented by `object` are obtained, using a normal approximation to the distribution of the (restricted) maximum likelihood estimators (the estimators are assumed to have a normal distribution centered at the true parameter values and with covariance matrix equal to the negative inverse Hessian matrix of the (restricted) log-likelihood evaluated at the estimated parameters). Confidence intervals are obtained in an unconstrained scale first, using the normal approximation, and, if necessary, transformed to the constrained scale.

Usage

```
## S3 method for class 'gls'
intervals(object, level, which, ...)
```

Arguments

`object`	an object inheriting from class "gls", representing a generalized least squares fitted linear model.
`level`	an optional numeric value for the interval confidence level. Defaults to 0.95.
`which`	an optional character string specifying the subset of parameters for which to construct the confidence intervals. Possible values are "all" for all parameters, "var-cov" for the variance-covariance parameters only, and "coef" for the linear model coefficients only. Defaults to "all".
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a list with components given by data frames with rows corresponding to parameters and columns `lower`, `est.`, and `upper` representing respectively lower confidence limits, the estimated values, and upper confidence limits for the parameters. Possible components are:

`coef`	linear model coefficients, only present when `which` is not equal to `"var-cov"`.
`corStruct`	correlation parameters, only present when `which` is not equal to `"coef"` and a correlation structure is used in `object`.
`varFunc`	variance function parameters, only present when `which` is not equal to `"coef"` and a variance function structure is used in `object`.
`sigma`	residual standard error.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`gls`, `intervals`, `print.intervals.gls`

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
intervals(fm1)
```

---

intervals.lme *Confidence Intervals on lme Parameters*

Description

Approximate confidence intervals for the parameters in the linear mixed-effects model represented by `object` are obtained, using a normal approximation to the distribution of the (restricted) maximum likelihood estimators (the estimators are assumed to have a normal distribution centered at the true parameter values and with covariance matrix equal to the negative inverse Hessian matrix of the (restricted) log-likelihood evaluated at the estimated parameters). Confidence intervals are obtained in an unconstrained scale first, using the normal approximation, and, if necessary, transformed to the constrained scale. The `pdNatural` parametrization is used for general positive-definite matrices.

Usage

```
## S3 method for class 'lme'
intervals(object, level = 0.95,
          which = c("all", "var-cov", "fixed"), ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model.
level	an optional numeric value with the confidence level for the intervals. Defaults to 0.95.
which	an optional character string specifying the subset of parameters for which to construct the confidence intervals. Possible values are "all" for all parameters, "var-cov" for the variance-covariance parameters only, and "fixed" for the fixed effects only. Defaults to "all".
...	some methods for this generic require additional arguments. None are used in this method.

Value

a list with components given by data frames with rows corresponding to parameters and columns lower, est., and upper representing respectively lower confidence limits, the estimated values, and upper confidence limits for the parameters. Possible components are:

fixed	fixed effects, only present when which is not equal to "var-cov".
reStruct	random effects variance-covariance parameters, only present when which is not equal to "fixed".
corStruct	within-group correlation parameters, only present when which is not equal to "fixed" and a correlation structure is used in object.
varFunc	within-group variance function parameters, only present when which is not equal to "fixed" and a variance function structure is used in object.
sigma	within-group standard deviation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

lme, intervals, print.intervals.lme, pdNatural

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
intervals(fm1)
```

`intervals.lmList` *Confidence Intervals on lmList Coefficients*

Description

Confidence intervals on the linear model coefficients are obtained for each `lm` component of `object` and organized into a three dimensional array. The first dimension corresponding to the names of the `object` components. The second dimension is given by `lower`, `est.`, and `upper` corresponding, respectively, to the lower confidence limit, estimated coefficient, and upper confidence limit. The third dimension is given by the coefficients names.

Usage

```
## S3 method for class 'lmList'
intervals(object, level = 0.95, pool = attr(object, "pool"), ...)
```

Arguments

`object`	an object inheriting from class `"lmList"`, representing a list of `lm` objects with a common model.
`level`	an optional numeric value with the confidence level for the intervals. Defaults to 0.95.
`pool`	an optional logical value indicating whether a pooled estimate of the residual standard error should be used. Default is `attr(object, "pool")`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a three dimensional array with the confidence intervals and estimates for the coefficients of each `lm` component of `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`lmList`, `intervals`, `plot.intervals.lmList`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
intervals(fm1)
```

isBalanced *Check a Design for Balance*

Description

Check the design of the experiment or study for balance.

Usage

```
isBalanced(object, countOnly, level)
```

Arguments

object	A groupedData object containing a data frame and a formula that describes the roles of variables in the data frame. The object will have one or more nested grouping factors and a primary covariate.
countOnly	A logical value indicating if the check for balance should only consider the number of observations at each level of the grouping factor(s). Defaults to FALSE.
level	an optional integer vector specifying the desired prediction levels. Levels increase from outermost to innermost grouping, with level 0 representing the population (fixed effects) predictions. Defaults to the innermost level.

Details

A design is balanced with respect to the grouping factor(s) if there are the same number of observations at each distinct value of the grouping factor or each combination of distinct levels of the nested grouping factors. If countOnly is FALSE the design is also checked for balance with respect to the primary covariate, which is often the time of the observation. A design is balanced with respect to the grouping factor and the covariate if the number of observations at each distinct level (or combination of levels for nested factors) is constant and the times at which the observations are taken (in general, the values of the primary covariates) also are constant.

Value

TRUE or FALSE according to whether the data are balanced or not

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

table, groupedData

Examples

```
isBalanced(Orthodont)                    # should return TRUE
isBalanced(Orthodont, countOnly = TRUE)  # should return TRUE
isBalanced(Pixel)                        # should return FALSE
isBalanced(Pixel, level = 1)             # should return FALSE
```

---

isInitialized *Check if Object is Initialized*

Description

Checks if `object` has been initialized (generally through a call to `Initialize`), by searching for components and attributes which are modified during initialization.

Usage

```
isInitialized(object)
```

Arguments

`object`	any object requiring initialization.

Value

a logical value indicating whether `object` has been initialized.

Author(s)

José Pinheiro and Douglas Bates

See Also

`Initialize`

Examples

```
pd1 <- pdDiag(~age)
isInitialized(pd1)
```

LDEsysMat *Generate system matrix for LDEs*

Description

Generate the system matrix for the linear differential equations determined by a compartment model.

Usage

```
LDEsysMat(pars, incidence)
```

Arguments

pars	a numeric vector of parameter values.
incidence	an integer matrix with columns named From, To, and Par. Values in the Par column must be in the range 1 to length(pars). Values in the From column must be between 1 and the number of compartments. Values in the To column must be between 0 and the number of compartments.

Details

A compartment model describes material transfer between k in a system of k compartments to a linear system of differential equations. Given a description of the system and a vector of parameter values this function returns the system matrix.

This function is intended for use in a general system for solving compartment models, as described in Bates and Watts (1988).

Value

A k by k numeric matrix.

Author(s)

Douglas Bates <bates@stat.wisc.edu>

References

Bates, D. M. and Watts, D. G. (1988), *Nonlinear Regression Analysis and Its Applications*, Wiley, New York.

Examples

```
# incidence matrix for a two compartment open system
incidence <-
  matrix(c(1,1,2,2,2,1,3,2,0), ncol = 3, byrow = TRUE,
   dimnames = list(NULL, c("Par", "From", "To")))
incidence
LDEsysMat(c(1.2, 0.3, 0.4), incidence)
```

lme *Linear Mixed-Effects Models*

Description

This generic function fits a linear mixed-effects model in the formulation described in Laird and Ware (1982) but allowing for nested random effects. The within-group errors are allowed to be correlated and/or have unequal variances.

This page describes the formula method; the methods `lme.lmList` and `lme.groupedData` are documented separately.

Usage

```
lme(fixed, data, random, correlation, weights, subset, method,
    na.action, control, contrasts = NULL, keep.data = TRUE)

## S3 method for class 'formula'
lme(fixed, data, random, correlation, weights, subset, method,
    na.action, control, contrasts = NULL, keep.data = TRUE)

## S3 method for class 'lme'
update(object, fixed., ..., evaluate = TRUE)
```

Arguments

`object`	an object inheriting from class `lme`, representing a fitted linear mixed-effects model.
`fixed`	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a `~` operator and the terms, separated by `+` operators, on the right, an "`lmList`" object, or a "`groupedData`" object. There is limited support for formulae such as `resp ~ 1` and `resp ~ 0`, and less prior to version '`3.1-112`'.
`fixed.`	Changes to the fixed-effects formula – see `update.formula` for details.
`data`	an optional data frame containing the variables named in `fixed`, `random`, `correlation`, `weights`, and `subset`. By default the variables are taken from the environment from which `lme` is called.
`random`	optionally, any of the following: (i) a one-sided formula of the form `~ x1 + ... + xn | g1/.../gm`, with `x1 + ... + xn` specifying the model for the random effects and `g1/.../gm` the grouping structure (`m` may be equal to 1, in which case no `/` is required). The random effects formula will be repeated for all levels of grouping, in the case of multiple levels of grouping; (ii) a list of one-sided formulas of the form `~ x1 + ... + xn | g`, with possibly different random effects models for each grouping level. The order of nesting will be assumed the same as the order of the elements in the list; (iii) a one-sided formula of the form `~ x1 + ... + xn`, or a `pdMat` object with a formula (i.e. a non-`NULL` value for

formula(object)), or a list of such formulas or pdMat objects. In this case, the grouping structure formula will be derived from the data used to fit the linear mixed-effects model, which should inherit from class "groupedData"; (iv) a named list of formulas or pdMat objects as in (iii), with the grouping factors as names. The order of nesting will be assumed the same as the order of the order of the elements in the list; (v) an reStruct object. See the documentation on pdClasses for a description of the available pdMat classes. Defaults to a formula consisting of the right hand side of fixed.

correlation — an optional corStruct object describing the within-group correlation structure. See the documentation of corClasses for a description of the available corStruct classes. Defaults to NULL, corresponding to no within-group correlations.

weights — an optional varFunc object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to varFixed, corresponding to fixed variance weights. See the documentation on varClasses for a description of the available varFunc classes. Defaults to NULL, corresponding to homoscedastic within-group errors.

subset — an optional expression indicating the subset of the rows of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.

method — a character string. If "REML" the model is fit by maximizing the restricted log-likelihood. If "ML" the log-likelihood is maximized. Defaults to "REML".

na.action — a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes lme to print an error message and terminate if there are any incomplete observations.

control — a list of control values for the estimation algorithm to replace the default values returned by the function lmeControl. Defaults to an empty list.

contrasts — an optional list. See the contrasts.arg of model.matrix.default.

keep.data — logical: should the data argument (if supplied and a data frame) be saved as part of the model object?

... — some methods for this generic require additional arguments. None are used in this method.

evaluate — If TRUE evaluate the new call else return the call.

Details

offset terms in fixed are an error since 3.1-157 (2022-03): previously they were silently ignored.

Value

An object of class "lme" representing the linear mixed-effects model fit. Generic functions such as print, plot and summary have methods to show the results of the fit. See lmeObject for the components of the fit. The functions resid, coef, fitted, fixed.effects, and random.effects can be used to extract some of its components.

Note

The function does not do any scaling internally: the optimization will work best when the response is scaled so its variance is of the order of one.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

The computational methods follow the general framework of Lindstrom and Bates (1988). The model formulation is described in Laird and Ware (1982). The variance-covariance parametrizations are described in Pinheiro and Bates (1996). The different correlation structures available for the `correlation` argument are described in Box, Jenkins and Reinsel (1994), Littell *et al* (1996), and Venables and Ripley (2002). The use of variance functions for linear and nonlinear mixed effects models is presented in detail in Davidian and Giltinan (1995).

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994). *Time Series Analysis: Forecasting and Control*, 3rd Edition, Holden–Day.

Davidian, M. and Giltinan, D.M. (1995). *Nonlinear Mixed Effects Models for Repeated Measurement Data*, Chapman and Hall. doi:10.1201/9780203745502.

Laird, N.M. and Ware, J.H. (1982). Random-Effects Models for Longitudinal Data. *Biometrics* **38**, 963–974. doi:10.2307/2529876.

Lindstrom, M.J. and Bates, D.M. (1988). Newton-Raphson and EM Algorithms for Linear Mixed-Effects Models for Repeated-Measures Data. *Journal of the American Statistical Association* **83**, 1014–1022. doi:10.2307/2290128.

Littell, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996). *SAS Systems for Mixed Models*, SAS Institute.

Pinheiro, J.C. and Bates., D.M. (1996). Unconstrained Parametrizations for Variance-Covariance Matrices. *Statistics and Computing* **6**, 289–296. doi:10.1007/BF00140873.

Pinheiro, J.C., and Bates, D.M. (2000). *Mixed-Effects Models in S and S-PLUS*, Springer. doi:10.1007/b98882.

Venables, W.N. and Ripley, B.D. (2002). *Modern Applied Statistics with S*, 4th Edition, Springer-Verlag. doi:10.1007/9780387217062.

See Also

`corClasses`, `lme.lmList`, `lme.groupedData`, `lmeControl`, `lmeObject`, `lmeStruct`, `lmList`, `pdClasses`, `plot.lme`, `predict.lme`, `qqnorm.lme`, `residuals.lme`, `reStruct`, `simulate.lme`, `summary.lme`, `varClasses`, `varFunc`

Examples

```
fm1 <- lme(distance ~ age, data = Orthodont) # random is ~ age
fm2 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
summary(fm1)
summary(fm2)
```

`lme.groupedData` *LME fit from groupedData Object*

Description

The response variable and primary covariate in `formula(fixed)` are used to construct the fixed effects model formula. This formula and the `groupedData` object are passed as the `fixed` and `data` arguments to `lme.formula`, together with any other additional arguments in the function call. See the documentation on `lme.formula` for a description of that function.

Usage

```
## S3 method for class 'groupedData'
lme(fixed, data, random, correlation, weights,
    subset, method, na.action, control, contrasts, keep.data = TRUE)
```

Arguments

`fixed`	a data frame inheriting from class "`groupedData`".
`data`	this argument is included for consistency with the generic function. It is ignored in this method function.
`random`	optionally, any of the following: (i) a one-sided formula of the form `~x1+...+xn | g1/.../gm`, with `x1+...+xn` specifying the model for the random effects and `g1/.../gm` the grouping structure (`m` may be equal to 1, in which case no `/` is required). The random effects formula will be repeated for all levels of grouping, in the case of multiple levels of grouping; (ii) a list of one-sided formulas of the form `~x1+...+xn | g`, with possibly different random effects models for each grouping level. The order of nesting will be assumed the same as the order of the elements in the list; (iii) a one-sided formula of the form `~x1+...+xn`, or a `pdMat` object with a formula (i.e. a non-`NULL` value for `formula(object)`), or a list of such formulas or `pdMat` objects. In this case, the grouping structure formula will be derived from the data used to fit the linear mixed-effects model, which should inherit from class `groupedData`; (iv) a named list of formulas or `pdMat` objects as in (iii), with the grouping factors as names. The order of nesting will be assumed the same as the order of the order of the elements in the list; (v) an `reStruct` object. See the documentation on `pdClasses` for a description of the available `pdMat` classes. Defaults to a formula consisting of the right hand side of `fixed`.
`correlation`	an optional `corStruct` object describing the within-group correlation structure. See the documentation of `corClasses` for a description of the available `corStruct` classes. Defaults to `NULL`, corresponding to no within-group correlations.
`weights`	an optional `varFunc` object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to `varFixed`, corresponding to fixed variance weights. See the documentation on `varClasses` for a description of the available `varFunc` classes. Defaults to `NULL`, corresponding to homoscedastic within-group errors.

subset	an optional expression indicating the subset of the rows of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
method	a character string. If "REML" the model is fit by maximizing the restricted log-likelihood. If "ML" the log-likelihood is maximized. Defaults to "REML".
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes lme to print an error message and terminate if there are any incomplete observations.
control	a list of control values for the estimation algorithm to replace the default values returned by the function lmeControl. Defaults to an empty list.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
keep.data	logical: should the data argument (if supplied and a data frame) be saved as part of the model object?

Value

an object of class lme representing the linear mixed-effects model fit. Generic functions such as print, plot and summary have methods to show the results of the fit. See lmeObject for the components of the fit. The functions resid, coef, fitted, fixed.effects, and random.effects can be used to extract some of its components.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

The computational methods follow on the general framework of Lindstrom, M.J. and Bates, D.M. (1988). The model formulation is described in Laird, N.M. and Ware, J.H. (1982). The variance-covariance parametrizations are described in Pinheiro, J.C. and Bates., D.M. (1996). The different correlation structures available for the correlation argument are described in Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994), Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996), and Venables, W.N. and Ripley, B.D. (2002). The use of variance functions for linear and nonlinear mixed effects models is presented in detail in Davidian, M. and Giltinan, D.M. (1995).

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Davidian, M. and Giltinan, D.M. (1995) "Nonlinear Mixed Effects Models for Repeated Measurement Data", Chapman and Hall.

Laird, N.M. and Ware, J.H. (1982) "Random-Effects Models for Longitudinal Data", Biometrics, 38, 963-974.

Lindstrom, M.J. and Bates, D.M. (1988) "Newton-Raphson and EM Algorithms for Linear Mixed-Effects Models for Repeated-Measures Data", Journal of the American Statistical Association, 83, 1014-1022.

Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289-296.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

See Also

lme, groupedData, lmeObject

Examples

```
fm1 <- lme(Orthodont)
summary(fm1)
```

---

lme.lmList *LME fit from lmList Object*

Description

If the random effects names defined in random are a subset of the lmList object coefficient names, initial estimates for the covariance matrix of the random effects are obtained (overwriting any values given in random). formula(fixed) and the data argument in the calling sequence used to obtain fixed are passed as the fixed and data arguments to lme.formula, together with any other additional arguments in the function call. See the documentation on lme.formula for a description of that function.

Usage

```
## S3 method for class 'lmList'
lme(fixed, data, random, correlation, weights, subset, method,
    na.action, control, contrasts, keep.data)
```

Arguments

fixed	an object inheriting from class "lmList.", representing a list of lm fits with a common model.
data	this argument is included for consistency with the generic function. It is ignored in this method function.
random	an optional one-sided linear formula with no conditioning expression, or a pdMat object with a formula attribute. Multiple levels of grouping are not allowed with this method function. Defaults to a formula consisting of the right hand side of formula(fixed).
correlation	an optional corStruct object describing the within-group correlation structure. See the documentation of corClasses for a description of the available corStruct classes. Defaults to NULL, corresponding to no within-group correlations.

weights	an optional varFunc object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to varFixed, corresponding to fixed variance weights. See the documentation on varClasses for a description of the available varFunc classes. Defaults to NULL, corresponding to homoscedastic within-group errors.
subset	an optional expression indicating the subset of the rows of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
method	a character string. If "REML" the model is fit by maximizing the restricted log-likelihood. If "ML" the log-likelihood is maximized. Defaults to "REML".
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes lme to print an error message and terminate if there are any incomplete observations.
control	a list of control values for the estimation algorithm to replace the default values returned by the function lmeControl. Defaults to an empty list.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
keep.data	logical: should the data argument (if supplied and a data frame) be saved as part of the model object?

Value

an object of class lme representing the linear mixed-effects model fit. Generic functions such as print, plot and summary have methods to show the results of the fit. See lmeObject for the components of the fit. The functions resid, coef, fitted, fixed.effects, and random.effects can be used to extract some of its components.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

The computational methods follow the general framework of Lindstrom and Bates (1988). The model formulation is described in Laird and Ware (1982). The variance-covariance parametrizations are described in Pinheiro and Bates (1996). The different correlation structures available for the correlation argument are described in Box, Jenkins and Reinse (1994), Littel *et al* (1996), and Venables and Ripley, (2002). The use of variance functions for linear and nonlinear mixed effects models is presented in detail in Davidian and Giltinan (1995).

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden–Day.

Davidian, M. and Giltinan, D.M. (1995) "Nonlinear Mixed Effects Models for Repeated Measurement Data", Chapman and Hall.

Laird, N.M. and Ware, J.H. (1982) "Random-Effects Models for Longitudinal Data", Biometrics, 38, 963–974.

Lindstrom, M.J. and Bates, D.M. (1988) "Newton-Raphson and EM Algorithms for Linear Mixed-Effects Models for Repeated-Measures Data", Journal of the American Statistical Association, 83, 1014–1022.

Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289–296.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

See Also

lme, lmList, lmeObject

Examples

```
fm1 <- lmList(Orthodont)
fm2 <- lme(fm1)
summary(fm1)
summary(fm2)
```

---

lmeControl *Specifying Control Values for lme Fit*

Description

The values supplied in the `lmeControl()` call replace the defaults, and a `list` with all settings (i.e., values for all possible arguments) is returned. The returned list is used as the `control` argument to the `lme` function.

Usage

```
lmeControl(maxIter = 50, msMaxIter = 50, tolerance = 1e-6, niterEM = 25,
           msMaxEval = 200,
    msTol = 1e-7, msVerbose = FALSE,
           returnObject = FALSE, gradHess = TRUE, apVar = TRUE,
    .relStep = .Machine$double.eps^(1/3), minAbsParApVar = 0.05,
           opt = c("nlminb", "optim"),
    optimMethod = "BFGS", natural = TRUE,
    sigma = NULL,
           allow.n.lt.q = FALSE,
           ...)
```

Arguments

maxIter	maximum number of iterations for the lme optimization algorithm. Default is 50.
msMaxIter	maximum number of iterations for the optimization step inside the lme optimization. Default is 50.
tolerance	tolerance for the convergence criterion in the lme algorithm. Default is 1e-6.
niterEM	number of iterations for the EM algorithm used to refine the initial estimates of the random effects variance-covariance coefficients. Default is 25.
msMaxEval	maximum number of evaluations of the objective function permitted for nlminb. Default is 200.
msTol	tolerance for the convergence criterion on the first iteration when optim is used. Default is 1e-7.
msVerbose	a logical value passed as the trace argument to nlminb or optim. Default is FALSE.
returnObject	a logical value indicating whether the fitted object should be returned with a warning (instead of an error via stop()) when the maximum number of iterations is reached without convergence of the algorithm. Default is FALSE.
gradHess	a logical value indicating whether numerical gradient vectors and Hessian matrices of the log-likelihood function should be used in the internal optimization. This option is only available when the correlation structure (corStruct) and the variance function structure (varFunc) have no "varying" parameters and the pdMat classes used in the random effects structure are pdSymm (general positive-definite), pdDiag (diagonal), pdIdent (multiple of the identity), or pdCompSymm (compound symmetry). Default is TRUE.
apVar	a logical value indicating whether the approximate covariance matrix of the variance-covariance parameters should be calculated. Default is TRUE.
.relStep	relative step for numerical derivatives calculations. Default is .Machine$double.eps^(1/3).
opt	the optimizer to be used, either "nlminb" (the default) or "optim".
optimMethod	character - the optimization method to be used with the optim optimizer. The default is "BFGS". An alternative is "L-BFGS-B".
minAbsParApVar	numeric value - minimum absolute parameter value in the approximate variance calculation. The default is 0.05.
natural	a logical value indicating whether the pdNatural parametrization should be used for general positive-definite matrices (pdSymm) in reStruct, when the approximate covariance matrix of the estimators is calculated. Default is TRUE.
sigma	optionally a positive number to fix the residual error at. If NULL, as by default, or 0, sigma is estimated.
allow.n.lt.q	logical indicating if it is ok to have less observations than random effects for each group. The default, FALSE signals an error; if NA, such a situation only gives a warning, as in **nlme** versions prior to 2019; if true, no message is given at all.
...	further named control arguments to be passed, depending on opt, to nlminb (those from abs.tol down) or optim (those except trace and maxit; reltol is used only from the second iteration).

Value

a list with components for each of the possible arguments.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>; the sigma option: Siem Heisterkamp and Bert van Willigen.

See Also

lme, nlminb, optim

Examples

```
# decrease the maximum number iterations in the ms call and
# request that information on the evolution of the ms iterations be printed
str(lCtr <- lmeControl(msMaxIter = 20, msVerbose = TRUE))
## This should always work:
do.call(lmeControl, lCtr)
```

---

lmeObject *Fitted lme Object*

Description

An object returned by the lme function, inheriting from class "lme" and representing a fitted linear mixed-effects model. Objects of this class have methods for the generic functions anova, coef, fitted, fixed.effects, formula, getGroups, getResponse, intervals, logLik, pairs, plot, predict, print, random.effects, residuals, sigma, summary, update, and vcov.

Value

The following components must be included in a legitimate "lme" object.

apVar	an approximate covariance matrix for the variance-covariance coefficients. If apVar = FALSE in the control values used in the call to lme, this component is NULL.
call	a list containing an image of the lme call that produced the object.
coefficients	a list with two components, fixed and random, where the first is a vector containing the estimated fixed effects and the second is a list of matrices with the estimated random effects for each level of grouping. For each matrix in the random list, the columns refer to the random effects and the rows to the groups.
contrasts	a list of the contrast matrices used to represent factors in the fixed effects formula and/or random effects formula. This information is important for making predictions from a new data frame in which not all levels of the original factors are observed. If no factors are used in the lme model, this component will be an empty list.

`dims` a list with basic dimensions used in the lme fit, including the components `N` - the number of observations in the data, `Q` - the number of grouping levels, `qvec` - the number of random effects at each level from innermost to outermost (last two values are equal to zero and correspond to the fixed effects and the response), `ngrps` - the number of groups at each level from innermost to outermost (last two values are one and correspond to the fixed effects and the response), and `ncol` - the number of columns in the model matrix for each level of grouping from innermost to outermost (last two values are equal to the number of fixed effects and one).

`fitted` a data frame with the fitted values as columns. The leftmost column corresponds to the population fixed effects (corresponding to the fixed effects only) and successive columns from left to right correspond to increasing levels of grouping.

`fixDF` a list with components `X` and `terms` specifying the denominator degrees of freedom for, respectively, t-tests for the individual fixed effects and F-tests for the fixed-effects terms in the models.

`groups` a data frame with the grouping factors as columns. The grouping level increases from left to right.

`logLik` the (restricted) log-likelihood at convergence.

`method` the estimation method: either `"ML"` for maximum likelihood, or `"REML"` for restricted maximum likelihood.

`modelStruct` an object inheriting from class `lmeStruct`, representing a list of mixed-effects model components, such as `reStruct`, `corStruct`, and `varFunc` objects.

`numIter` the number of iterations used in the iterative algorithm.

`residuals` a data frame with the residuals as columns. The leftmost column corresponds to the population residuals and successive columns from left to right correspond to increasing levels of grouping.

`terms` the `terms`, including `formula`, see also terms.object.

`sigma` the estimated within-group error standard deviation.

`varFix` an approximate covariance matrix of the fixed effects estimates.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lme`, `lmeStruct`

lmeStruct *Linear Mixed-Effects Structure*

Description

A linear mixed-effects structure is a list of model components representing different sets of parameters in the linear mixed-effects model. An lmeStruct list must contain at least a reStruct object, but may also contain corStruct and varFunc objects. NULL arguments are not included in the lmeStruct list.

Usage

```
lmeStruct(reStruct, corStruct, varStruct)
```

Arguments

reStruct	a reStruct representing a random effects structure.
corStruct	an optional corStruct object, representing a correlation structure. Default is NULL.
varStruct	an optional varFunc object, representing a variance function structure. Default is NULL.

Value

a list of model components determining the parameters to be estimated for the associated linear mixed-effects model.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

corClasses, lme, residuals.lmeStruct, reStruct, varFunc

Examples

```
lms1 <- lmeStruct(reStruct(~age), corAR1(), varPower())
```

lmList *List of lm Objects with a Common Model*

Description

Data is partitioned according to the levels of the grouping factor g and individual lm fits are obtained for each data partition, using the model defined in object.

Usage

```
lmList(object, data, level, subset, na.action = na.fail,
       pool = TRUE, warn.lm = TRUE)

## S3 method for class 'formula'
lmList(object, data, level, subset, na.action = na.fail,
       pool = TRUE, warn.lm = TRUE)

## S3 method for class 'lmList'
update(object, formula., ..., evaluate = TRUE)
## S3 method for class 'lmList'
print(x, pool, ...)
```

Arguments

object	For lmList, either a linear formula object of the form y ~ x1+...+xn | g or a groupedData object. In the formula object, y represents the response, x1, ..., xn the covariates, and g the grouping factor specifying the partitioning of the data according to which different lm fits should be performed. The grouping factor g may be omitted from the formula, in which case the grouping structure will be obtained from data, which must inherit from class groupedData. The method function lmList.groupedData is documented separately. For the method update.lmList, object is an object inheriting from class lmList.
formula	(used in update.lmList only) a two-sided linear formula with the common model for the individuals lm fits.
formula.	Changes to the formula – see update.formula for details.
data	a data frame in which to interpret the variables named in object.
level	an optional integer specifying the level of grouping to be used when multiple nested levels of grouping are present.
subset	an optional expression indicating which subset of the rows of data should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes lmList to print an error message and terminate if there are any incomplete observations.

pool	an optional logical value indicating whether a pooled estimate of the residual standard error should be used in calculations of standard deviations or standard errors for summaries.
warn.lm	logical indicating if lm() errors (all of which are caught by tryCatch) should be signalled as a "summarizing" warning.
x	an object inheriting from class lmList to be printed.
...	some methods for this generic require additional arguments. None are used in this method.
evaluate	If TRUE evaluate the new call else return the call.

Value

a list of lm objects with as many components as the number of groups defined by the grouping factor. Generic functions such as coef, fixed.effects, lme, pairs, plot, predict, random.effects, summary, and update have methods that can be applied to an lmList object.

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

lm, lme.lmList, plot.lmList, pooledSD, predict.lmList, residuals.lmList, summary.lmList

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
summary(fm1)
```

---

lmList.groupedData *lmList Fit from a groupedData Object*

Description

The response variable and primary covariate in formula(object) are used to construct the linear model formula. This formula and the groupedData object are passed as the object and data arguments to lmList.formula, together with any other additional arguments in the function call. See the documentation on lmList.formula for a description of that function.

Usage

```
## S3 method for class 'groupedData'
lmList(object, data, level, subset, na.action = na.fail,
       pool = TRUE, warn.lm = TRUE)
```

Arguments

object	a data frame inheriting from class "groupedData".
data	this argument is included for consistency with the generic function. It is ignored in this method function.
level	an optional integer specifying the level of grouping to be used when multiple nested levels of grouping are present.
subset	an optional expression indicating which subset of the rows of data should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes lmList to print an error message and terminate if there are any incomplete observations.
pool, warn.lm	optional logicals, see lmList.

Value

a list of lm objects with as many components as the number of groups defined by the grouping factor. Generic functions such as coef, fixed.effects, lme, pairs, plot, predict, random.effects, summary, and update have methods that can be applied to an lmList object.

See Also

groupedData, lm, lme.lmList, lmList, lmList.formula

Examples

```
fm1 <- lmList(Orthodont)
summary(fm1)
```

---

logDet *Extract the Logarithm of the Determinant*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: corStruct, several pdMat classes, and reStruct.

Usage

```
logDet(object, ...)
```

Arguments

object	any object from which a matrix, or list of matrices, can be extracted
...	some methods for this generic function require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

logLik, logDet.corStruct, logDet.pdMat, logDet.reStruct

---

logDet.corStruct *Extract corStruct Log-Determinant*

Description

This method function extracts the logarithm of the determinant of a square-root factor of the correlation matrix associated with `object`, or the sum of the log-determinants of square-root factors of the list of correlation matrices associated with `object`.

Usage

```
## S3 method for class 'corStruct'
logDet(object, covariate, ...)
```

Arguments

`object`	an object inheriting from class "corStruct", representing a correlation structure.
`covariate`	an optional covariate vector (matrix), or list of covariate vectors (matrices), at which values the correlation matrix, or list of correlation matrices, are to be evaluated. Defaults to `getCovariate(object)`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the log-determinant of a square-root factor of the correlation matrix associated with `object`, or the sum of the log-determinants of square-root factors of the list of correlation matrices associated with `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

logLik.corStruct, corMatrix.corStruct, logDet

Examples

```
cs1 <- corAR1(0.3)
logDet(cs1, covariate = 1:4)
```

`logDet.pdMat` *Extract Log-Determinant from a pdMat Object*

Description

This method function extracts the logarithm of the determinant of a square-root factor of the positive-definite matrix represented by `object`.

Usage

```
## S3 method for class 'pdMat'
logDet(object, ...)
```

Arguments

`object`	an object inheriting from class "pdMat", representing a positive definite matrix.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the log-determinant of a square-root factor of the positive-definite matrix represented by `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

pdMat, logDet

Examples

```
pd1 <- pdSymm(diag(1:3))
logDet(pd1)
```

logDet.reStruct *Extract reStruct Log-Determinants*

Description

Calculates, for each of the pdMat components of object, the logarithm of the determinant of a square-root factor.

Usage

```
## S3 method for class 'reStruct'
logDet(object, ...)
```

Arguments

object	an object inheriting from class "reStruct", representing a random effects structure and consisting of a list of pdMat objects.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the log-determinants of square-root factors of the pdMat components of object.

Author(s)

José Pinheiro

See Also

reStruct, pdMat, logDet

Examples

```
rs1 <- reStruct(list(A = pdSymm(diag(1:3), form = ~Score),
  B = pdDiag(2 * diag(4), form = ~Educ)))
logDet(rs1)
```

`logLik.corStruct` *Extract corStruct Log-Likelihood*

Description

This method function extracts the component of a Gaussian log-likelihood associated with the correlation structure, which is equal to the negative of the logarithm of the determinant (or sum of the logarithms of the determinants) of the matrix (or matrices) represented by `object`.

Usage

```
## S3 method for class 'corStruct'
logLik(object, data, ...)
```

Arguments

`object`	an object inheriting from class "`corStruct`", representing a correlation structure.
`data`	this argument is included to make this method function compatible with other `logLik` methods and will be ignored.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the negative of the logarithm of the determinant (or sum of the logarithms of the determinants) of the correlation matrix (or matrices) represented by `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`logDet.corStruct`, `logLik.lme`,

Examples

```
cs1 <- corAR1(0.2)
cs1 <- Initialize(cs1, data = Orthodont)
logLik(cs1)
```

logLik.glsStruct *Log-Likelihood of a glsStruct Object*

Description

Pars is used to update the coefficients of the model components of object and the individual (restricted) log-likelihood contributions of each component are added together. The type of log-likelihood (restricted or not) is determined by the settings attribute of object.

Usage

```
## S3 method for class 'glsStruct'
logLik(object, Pars, conLin, ...)
```

Arguments

object	an object inheriting from class "glsStruct", representing a list of linear model components, such as corStruct and "varFunc" objects.
Pars	the parameter values at which the (restricted) log-likelihood is to be evaluated.
conLin	an optional condensed linear model object, consisting of a list with components "Xy", corresponding to a regression matrix (X) combined with a response vector (y), and "logLik", corresponding to the log-likelihood of the underlying linear model. Defaults to attr(object, "conLin").
...	some methods for this generic require additional arguments. None are used in this method.

Value

the (restricted) log-likelihood for the linear model described by object, evaluated at Pars.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gls, glsStruct, logLik.lme

logLik.gnls *Log-Likelihood of a gnls Object*

Description

Returns the log-likelihood value of the nonlinear model represented by `object` evaluated at the estimated coefficients.

Usage

```
## S3 method for class 'gnls'
logLik(object, REML, ...)
```

Arguments

`object`	an object inheriting from class "gnls", representing a generalized nonlinear least squares fitted model.
`REML`	an logical value for consistency with `logLik,gls`, but only `FALSE` is accepted..
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the log-likelihood of the linear model represented by `object` evaluated at the estimated coefficients.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gnls, logLik.lme

Examples

```
fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            weights = varPower())
logLik(fm1)
```

`logLik.gnlsStruct` *Log-Likelihood of a gnlsStruct Object*

Description

`Pars` is used to update the coefficients of the model components of `object` and the individual log-likelihood contributions of each component are added together.

Usage

```
## S3 method for class 'gnlsStruct'
logLik(object, Pars, conLin, ...)
```

Arguments

`object`	an object inheriting from class `gnlsStruct`, representing a list of model components, such as `corStruct` and `varFunc` objects, and attributes specifying the underlying nonlinear model and the response variable.
`Pars`	the parameter values at which the log-likelihood is to be evaluated.
`conLin`	an optional condensed linear model object, consisting of a list with components `"Xy"`, corresponding to a regression matrix (`X`) combined with a response vector (`y`), and `"logLik"`, corresponding to the log-likelihood of the underlying nonlinear model. Defaults to `attr(object, "conLin")`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the log-likelihood for the linear model described by `object`, evaluated at `Pars`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`gnls`, `gnlsStruct`, `logLik.gnls`

logLik.lme *Log-Likelihood of an lme Object*

Description

If REML=FALSE, returns the log-likelihood value of the linear mixed-effects model represented by object evaluated at the estimated coefficients; else, the restricted log-likelihood evaluated at the estimated coefficients is returned.

Usage

```
## S3 method for class 'lme'
logLik(object, REML, ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model.
REML	an optional logical value. If TRUE the restricted log-likelihood is returned, else, if FALSE, the log-likelihood is returned. Defaults to the method of estimation used, that is TRUE if and only object was fitted with method = "REML" (the default for these fitting functions) .
...	some methods for this generic require additional arguments. None are used in this method.

Value

the (restricted) log-likelihood of the model represented by object evaluated at the estimated coefficients.

Author(s)

José Pinheiro and Douglas Bates

References

Harville, D.A. (1974) “Bayesian Inference for Variance Components Using Only Error Contrasts”, *Biometrika*, **61**, 383–385.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

lme,gls, logLik.corStruct, logLik.glsStruct, logLik.lmeStruct, logLik.lmList, logLik.reStruct, logLik.varFunc,

Examples

```
fm1 <- lme(distance ~ Sex * age, Orthodont, random = ~ age, method = "ML")
logLik(fm1)
logLik(fm1, REML = TRUE)
```

`logLik.lmeStruct` *Log-Likelihood of an lmeStruct Object*

Description

`Pars` is used to update the coefficients of the model components of `object` and the individual (restricted) log-likelihood contributions of each component are added together. The type of log-likelihood (restricted or not) is determined by the `settings` attribute of `object`.

Usage

```
## S3 method for class 'lmeStruct'
logLik(object, Pars, conLin, ...)
```

Arguments

`object`	an object inheriting from class `"lmeStruct"`, representing a list of linear mixed-effects model components, such as `reStruct`, `corStruct`, and `varFunc` objects.
`Pars`	the parameter values at which the (restricted) log-likelihood is to be evaluated.
`conLin`	an optional condensed linear model object, consisting of a list with components `"Xy"`, corresponding to a regression matrix (`X`) combined with a response vector (`y`), and `"logLik"`, corresponding to the log-likelihood of the underlying lme model. Defaults to `attr(object, "conLin")`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the (restricted) log-likelihood for the linear mixed-effects model described by `object`, evaluated at `Pars`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lme`, `lmeStruct`, `logLik.lme`

logLik.lmList *Log-Likelihood of an lmList Object*

Description

If pool=FALSE, the (restricted) log-likelihoods of the lm components of object are summed together. Else, the (restricted) log-likelihood of the lm fit with different coefficients for each level of the grouping factor associated with the partitioning of the object components is obtained.

Usage

```
## S3 method for class 'lmList'
logLik(object, REML, pool, ...)
```

Arguments

object	an object inheriting from class "lmList", representing a list of lm objects with a common model.
REML	an optional logical value. If TRUE the restricted log-likelihood is returned, else, if FALSE, the log-likelihood is returned. Defaults to FALSE.
pool	an optional logical value indicating whether all lm components of object may be assumed to have the same error variance. Default is attr(object, "pool").
...	some methods for this generic require additional arguments. None are used in this method.

Value

either the sum of the (restricted) log-likelihoods of each lm component in object, or the (restricted) log-likelihood for the lm fit with separate coefficients for each component of object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lmList, logLik.lme,

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
logLik(fm1)   # returns NA when it should not
```

`logLik.reStruct` *Calculate reStruct Log-Likelihood*

Description

Calculates the log-likelihood, or restricted log-likelihood, of the Gaussian linear mixed-effects model represented by `object` and `conLin` (assuming spherical within-group covariance structure), evaluated at `coef(object)`. The `settings` attribute of `object` determines whether the log-likelihood, or the restricted log-likelihood, is to be calculated. The computational methods are described in Bates and Pinheiro (1998).

Usage

```
## S3 method for class 'reStruct'
logLik(object, conLin, ...)
```

Arguments

`object`	an object inheriting from class `"reStruct"`, representing a random effects structure and consisting of a list of `pdMat` objects.
`conLin`	a condensed linear model object, consisting of a list with components `"Xy"`, corresponding to a regression matrix (`X`) combined with a response vector (`y`), and `"logLik"`, corresponding to the log-likelihood of the underlying model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the log-likelihood, or restricted log-likelihood, of linear mixed-effects model represented by `object` and `conLin`, evaluated at `coef{object}`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`reStruct`, `pdMat`, `logLik.lme`

logLik.varFunc *Extract varFunc logLik*

Description

This method function extracts the component of a Gaussian log-likelihood associated with the variance function structure represented by `object`, which is equal to the sum of the logarithms of the corresponding weights.

Usage

```
## S3 method for class 'varFunc'
logLik(object, data, ...)
```

Arguments

`object`	an object inheriting from class "varFunc", representing a variance function structure.
`data`	this argument is included to make this method function compatible with other `logLik` methods and will be ignored.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the sum of the logarithms of the weights corresponding to the variance function structure represented by `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

logLik.lme

Examples

```
vf1 <- varPower(form = ~age)
vf1 <- Initialize(vf1, Orthodont)
coef(vf1) <- 0.1
logLik(vf1)
```

Machines *Productivity Scores for Machines and Workers*

Description

The Machines data frame has 54 rows and 3 columns.

Format

This data frame contains the following columns:

Worker an ordered factor giving the unique identifier for the worker.

Machine a factor with levels A, B, and C identifying the machine brand.

score a productivity score.

Details

Data on an experiment to compare three brands of machines used in an industrial process are presented in Milliken and Johnson (p. 285, 1992). Six workers were chosen randomly among the employees of a factory to operate each machine three times. The response is an overall productivity score taking into account the number and quality of components produced.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.14)

Milliken, G. A. and Johnson, D. E. (1992), *Analysis of Messy Data, Volume I: Designed Experiments*, Chapman and Hall, London.

MathAchieve *Mathematics achievement scores*

Description

The MathAchieve data frame has 7185 rows and 6 columns.

Format

This data frame contains the following columns:

School an ordered factor identifying the school that the student attends

Minority a factor with levels No Yes indicating if the student is a member of a minority racial group.

Sex a factor with levels Male Female

SES a numeric vector of socio-economic status.

MathAch a numeric vector of mathematics achievement scores.

MEANSES a numeric vector of the mean SES for the school.

Details

Each row in this data frame contains the data for one student.

Examples

```
summary(MathAchieve)
```

MathAchSchool *School demographic data for MathAchieve*

Description

The MathAchSchool data frame has 160 rows and 7 columns.

Format

This data frame contains the following columns:

School a factor giving the school on which the measurement is made.

Size a numeric vector giving the number of students in the school

Sector a factor with levels Public Catholic

PRACAD a numeric vector giving the percentage of students on the academic track

DISCLIM a numeric vector measuring the discrimination climate

HIMINTY a factor with levels 0 1

MEANSES a numeric vector giving the mean SES score.

Details

These variables give the school-level demographic data to accompany the MathAchieve data.

Matrix *Assign Matrix Values*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include pdMat, pdBlocked, and reStruct.

Usage

```
matrix(object) <- value
```

Arguments

object	any object to which as.matrix can be applied.
value	a matrix, or list of matrices, with the same dimensions as as.matrix(object) with the new values to be assigned to the matrix associated with object.

Value

will depend on the method function; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

as.matrix, also for examples, matrix<-.pdMat, matrix<-.reStruct.

Matrix.pdMat *Assign Matrix to a pdMat or pdBlocked Object*

Description

The positive-definite matrix represented by object is replaced by value. If the original matrix had row and/or column names, the corresponding names for value can either be NULL, or a permutation of the original names.

Usage

```
## S3 replacement method for class 'pdMat'
matrix(object) <- value
## S3 replacement method for class 'pdBlocked'
matrix(object) <- value
```

Arguments

object	an object inheriting from class "pdMat", representing a positive definite matrix.
value	a matrix with the new values to be assigned to the positive-definite matrix represented by object. Must have the same dimensions as as.matrix(object).

Value

a pdMat or pdBlocked object similar to object, but with its coefficients modified to produce the matrix in value.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

pdMat, "matrix<-"

Examples

```
class(pd1 <- pdSymm(diag(3))) # "pdSymm" "pdMat"
matrix(pd1) <- diag(1:3)
pd1
```

Matrix.reStruct *Assign reStruct Matrices*

Description

The individual matrices in value are assigned to each pdMat component of object, in the order they are listed. The new matrices must have the same dimensions as the matrices they are meant to replace.

Usage

```
## S3 replacement method for class 'reStruct'
matrix(object) <-  value
```

Arguments

object	an object inheriting from class "reStruct", representing a random effects structure and consisting of a list of pdMat objects.
value	a matrix, or list of matrices, with the new values to be assigned to the matrices associated with the pdMat components of object.

Value

an reStruct object similar to object, but with the coefficients of the individual pdMat components modified to produce the matrices listed in value.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

reStruct, pdMat, "matrix<-"

Examples

```
rs1 <- reStruct(list(Dog = ~day, Side = ~1), data = Pixel)
matrix(rs1) <- list(diag(2), 3)
```

Meat *Tenderness of meat*

Description

The Meat data frame has 30 rows and 4 columns.

Format

This data frame contains the following columns:

Storage an ordered factor specifying the storage treatment - 1 (0 days), 2 (1 day), 3 (2 days), 4 (4 days), 5 (9 days), and 6 (18 days)

score a numeric vector giving the tenderness score of beef roast.

Block an ordered factor identifying the muscle from which the roast was extracted with levels II < V < I < III < IV

Pair an ordered factor giving the unique identifier for each pair of beef roasts with levels II-1 < ... < IV-1

Details

Cochran and Cox (section 11.51, 1957) describe data from an experiment conducted at Iowa State College (Paul, 1943) to compare the effects of length of cold storage on the tenderness of beef roasts. Six storage periods ranging from 0 to 18 days were used. Thirty roasts were scored by four judges on a scale from 0 to 10, with the score increasing with tenderness. The response was the sum of all four scores. Left and right roasts from the same animal were grouped into pairs, which were further grouped into five blocks, according to the muscle from which they were extracted. Different storage periods were applied to each roast within a pair according to a balanced incomplete block design.

Source

Cochran, W. G. and Cox, G. M. (1957), *Experimental Designs*, Wiley, New York.

Milk *Protein content of cows' milk*

Description

The Milk data frame has 1337 rows and 4 columns.

Format

This data frame contains the following columns:

protein a numeric vector giving the protein content of the milk.

Time a numeric vector giving the time since calving (weeks).

Cow an ordered factor giving a unique identifier for each cow.

Diet a factor with levels `barley`, `barley+lupins`, and `lupins` identifying the diet for each cow.

Details

Diggle, Liang, and Zeger (1994) describe data on the protein content of cows' milk in the weeks following calving. The cattle are grouped according to whether they are fed a diet with barley alone, with barley and lupins, or with lupins alone.

Source

Diggle, Peter J., Liang, Kung-Yee and Zeger, Scott L. (1994), *Analysis of longitudinal data*, Oxford University Press, Oxford.

`model.matrix.reStruct` *reStruct Model Matrix*

Description

The model matrices for each element of `formula(object)`, calculated using `data`, are bound together column-wise. When multiple grouping levels are present (i.e. when `length(object) > 1`), the individual model matrices are combined from innermost (at the leftmost position) to outermost (at the rightmost position).

Usage

```
## S3 method for class 'reStruct'
model.matrix(object, data, contrast, ...)
```

Arguments

`object`	an object inheriting from class `"reStruct"`, representing a random effects structure and consisting of a list of `pdMat` objects.
`data`	a data frame in which to evaluate the variables defined in `formula(object)`.
`contrast`	an optional named list specifying the contrasts to be used for representing the `factor` variables in `data`. The components names should match the names of the variables in `data` for which the contrasts are to be specified. The components of this list will be used as the `contrasts` attribute of the corresponding factor. If missing, the default contrast specification is used.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a matrix obtained by binding together, column-wise, the model matrices for each element of `formula(object)`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`model.matrix`, `contrasts`, `reStruct`, `formula.reStruct`

Examples

```
rs1 <- reStruct(list(Dog = ~day, Side = ~1), data = Pixel)
model.matrix(rs1, Pixel)
```

Muscle *Contraction of heart muscle sections*

Description

The `Muscle` data frame has 60 rows and 3 columns.

Format

This data frame contains the following columns:

Strip an ordered factor indicating the strip of muscle being measured.

conc a numeric vector giving the concentration of CaCl2

length a numeric vector giving the shortening of the heart muscle strip.

Details

Baumann and Waldvogel (1963) describe data on the shortening of heart muscle strips dipped in a $CaCl_2$ solution. The muscle strips are taken from the left auricle of a rat's heart.

Source

Baumann, F. and Waldvogel, F. (1963), La restitution pastsystolique de la contraction de l'oreillette gauche du rat. Effets de divers ions et de l'acetylcholine, *Helvetica Physiologica Acta*, **21**.

Names *Names Associated with an Object*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: `formula`, `pdBlocked`, `pdMat`, and `reStruct`.

Usage

```
Names(object, ...)
Names(object, ...) <- value
```

Arguments

`object`	any object for which names can be extracted and/or assigned.
`...`	some methods for this generic function require additional arguments.
`value`	names to be assigned to `object`.

Value

will depend on the method function used; see the appropriate documentation.

SIDE EFFECTS

On the left side of an assignment, sets the names associated with `object` to `value`, which must have an appropriate length.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`Names.formula`, `Names.pdBlocked`, `Names.pdMat`, `Names.reStruct`

`Names.formula` *Extract Names from a formula*

Description

This method function returns the names of the terms corresponding to the right hand side of `object` (treated as a linear formula), obtained as the column names of the corresponding `model.matrix`.

Usage

```
## S3 method for class 'formula'
Names(object, data, exclude, ...)
```

Arguments

`object`	an object inheriting from class "`formula`".
`data`	an optional data frame containing the variables specified in `object`. By default the variables are taken from the environment from which `Names.formula` is called.
`exclude`	an optional character vector with names to be excluded from the returned value. Default is `c("pi",".")`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a character vector with the column names of the `model.matrix` corresponding to the right hand side of `object` which are not listed in `excluded`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`model.matrix`, `terms`, `Names`

Examples

```
Names(distance ~ Sex * age, data = Orthodont)
```

Names.pdBlocked *Names of a pdBlocked Object*

Description

This method function extracts the first element of the Dimnames attribute, which contains the column names, for each block diagonal element in the matrix represented by object.

Usage

```
## S3 method for class 'pdBlocked'
Names(object, asList, ...)
```

Arguments

object	an object inheriting from class "pdBlocked" representing a positive-definite matrix with block diagonal structure
asList	a logical value. If TRUE a list with the names for each block diagonal element is returned. If FALSE a character vector with all column names is returned. Defaults to FALSE.
...	some methods for this generic require additional arguments. None are used in this method.

Value

if asList is FALSE, a character vector with column names of the matrix represented by object; otherwise, if asList is TRUE, a list with components given by the column names of the individual block diagonal elements in the matrix represented by object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Names, Names.pdMat

Examples

```
pd1 <- pdBlocked(list(~Sex - 1, ~age - 1), data = Orthodont)
Names(pd1)
```

Names.pdMat *Names of a pdMat Object*

Description

This method function returns the fist element of the Dimnames attribute of object, which contains the column names of the matrix represented by object.

Usage

```
## S3 method for class 'pdMat'
Names(object, ...)
## S3 replacement method for class 'pdMat'
Names(object, ...) <- value
```

Arguments

object	an object inheriting from class "pdMat", representing a positive-definite matrix.
value	a character vector with the replacement values for the column and row names of the matrix represented by object. It must have length equal to the dimension of the matrix represented by object and, if names have been previously assigned to object, it must correspond to a permutation of the original names.
...	some methods for this generic require additional arguments. None are used in this method.

Value

if object has a Dimnames attribute then the first element of this attribute is returned; otherwise NULL.

SIDE EFFECTS

On the left side of an assignment, sets the Dimnames attribute of object to list(value, value).

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Names, Names.pdBlocked

Examples

```
pd1 <- pdSymm(~age, data = Orthodont)
Names(pd1)
```

Names.reStruct *Names of an reStruct Object*

Description

This method function extracts the column names of each of the positive-definite matrices represented the pdMat elements of object.

Usage

```
## S3 method for class 'reStruct'
Names(object, ...)
## S3 replacement method for class 'reStruct'
Names(object, ...) <- value
```

Arguments

object	an object inheriting from class "reStruct", representing a random effects structure and consisting of a list of pdMat objects.
value	a list of character vectors with the replacement values for the names of the individual pdMat objects that form object. It must have the same length as object.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a list containing the column names of each of the positive-definite matrices represented by the pdMat elements of object.

SIDE EFFECTS

On the left side of an assignment, sets the Names of the pdMat elements of object to the corresponding element of value.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

reStruct, pdMat, Names.pdMat

Examples

```
rs1 <- reStruct(list(Dog = ~day, Side = ~1), data = Pixel)
Names(rs1)
```

needUpdate *Check if Update is Needed*

Description

This function is generic; method functions can be written to handle specific classes of objects. By default, it tries to extract a needUpdate attribute of object. If this is NULL or FALSE it returns FALSE; else it returns TRUE. Updating of objects usually takes place in iterative algorithms in which auxiliary quantities associated with the object, and not being optimized over, may change.

Usage

```
needUpdate(object)
```

Arguments

object	any object

Value

a logical value indicating whether object needs to be updated.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

needUpdate.modelStruct

Examples

```
vf1 <- varExp()
vf1 <- Initialize(vf1, data = Orthodont)
needUpdate(vf1)
```

needUpdate.modelStruct *Check if a modelStruct Object Needs Updating*

Description

This method function checks if any of the elements of object needs to be updated. Updating of objects usually takes place in iterative algorithms in which auxiliary quantities associated with the object, and not being optimized over, may change.

Usage

```
## S3 method for class 'modelStruct'
needUpdate(object)
```

Arguments

`object`	an object inheriting from class `"modelStruct"`, representing a list of model components, such as `corStruct` and `varFunc` objects.

Value

a logical value indicating whether any element of `object` needs to be updated.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`needUpdate`

Examples

```
lms1 <- lmeStruct(reStruct = reStruct(pdDiag(diag(2), ~age)),
   varStruct = varPower(form = ~age))
needUpdate(lms1)
```

Nitrendipene *Assay of nitrendipene*

Description

The `Nitrendipene` data frame has 89 rows and 4 columns.

Format

This data frame contains the following columns:

activity a numeric vector

NIF a numeric vector

Tissue an ordered factor with levels 2 < 1 < 3 < 4

log.NIF a numeric vector

Source

Bates, D. M. and Watts, D. G. (1988), *Nonlinear Regression Analysis and Its Applications*, Wiley, New York.

nlme *Nonlinear Mixed-Effects Models*

Description

This generic function fits a nonlinear mixed-effects model in the formulation described in Lindstrom and Bates (1990) but allowing for nested random effects. The within-group errors are allowed to be correlated and/or have unequal variances.

Usage

```
nlme(model, data, fixed, random, groups, start, correlation, weights,
     subset, method, na.action, naPattern, control, verbose)

## S3 method for class 'formula'
nlme(model, data, fixed, random, groups, start, correlation, weights,
     subset, method, na.action, naPattern, control, verbose)
```

Arguments

model	a nonlinear model formula, with the response on the left of a ~ operator and an expression involving parameters and covariates on the right, or an nlsList object. If data is given, all names used in the formula should be defined as parameters or variables in the data frame. The method function nlme.nlsList is documented separately.
data	an optional data frame containing the variables named in model, fixed, random, correlation, weights, subset, and naPattern. By default the variables are taken from the environment from which nlme is called.
fixed	a two-sided linear formula of the form f1+...+fn~x1+...+xm, or a list of two-sided formulas of the form f1~x1+...+xm, with possibly different models for different parameters. The f1,...,fn are the names of parameters included on the right hand side of model and the x1+...+xm expressions define linear models for these parameters (when the left hand side of the formula contains several parameters, they all are assumed to follow the same linear model, described by the right hand side expression). A 1 on the right hand side of the formula(s) indicates a single fixed effects for the corresponding parameter(s).
random	optionally, any of the following: (i) a two-sided formula of the form r1+...+rn~x1+...+xm | g1/.../gQ, with r1,...,rn naming parameters included on the right hand side of model, x1+...+xm specifying the random-effects model for these parameters and g1/.../gQ the grouping structure (Q may be equal to 1, in which case no / is required). The random effects formula will be repeated for all levels of grouping, in the case of multiple levels of grouping; (ii) a two-sided formula of the form r1+...+rn~x1+..+xm, a list of two-sided formulas of the form r1~x1+...+xm, with possibly different random-effects models for different parameters, a pdMat object with a two-sided formula, or list of two-sided

formulas (i.e. a non-NULL value for formula(random)), or a list of pdMat objects with two-sided formulas, or lists of two-sided formulas. In this case, the grouping structure formula will be given in groups, or derived from the data used to fit the nonlinear mixed-effects model, which should inherit from class groupedData,; (iii) a named list of formulas, lists of formulas, or pdMat objects as in (ii), with the grouping factors as names. The order of nesting will be assumed the same as the order of the order of the elements in the list; (iv) an reStruct object. See the documentation on pdClasses for a description of the available pdMat classes. Defaults to fixed, resulting in all fixed effects having also random effects.

groups — an optional one-sided formula of the form ~g1 (single level of nesting) or ~g1/.../gQ (multiple levels of nesting), specifying the partitions of the data over which the random effects vary. g1,...,gQ must evaluate to factors in data. The order of nesting, when multiple levels are present, is taken from left to right (i.e. g1 is the first level, g2 the second, etc.).

start — an optional numeric vector, or list of initial estimates for the fixed effects and random effects. If declared as a numeric vector, it is converted internally to a list with a single component fixed, given by the vector. The fixed component is required, unless the model function inherits from class selfStart, in which case initial values will be derived from a call to nlsList. An optional random component is used to specify initial values for the random effects and should consist of a matrix, or a list of matrices with length equal to the number of grouping levels. Each matrix should have as many rows as the number of groups at the corresponding level and as many columns as the number of random effects in that level.

correlation — an optional corStruct object describing the within-group correlation structure. See the documentation of corClasses for a description of the available corStruct classes. Defaults to NULL, corresponding to no within-group correlations.

weights — an optional varFunc object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to varFixed, corresponding to fixed variance weights. See the documentation on varClasses for a description of the available varFunc classes. Defaults to NULL, corresponding to homoscedastic within-group errors.

subset — an optional expression indicating the subset of the rows of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.

method — a character string. If "REML" the model is fit by maximizing the restricted log-likelihood. If "ML" the log-likelihood is maximized. Defaults to "ML".

na.action — a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes nlme to print an error message and terminate if there are any incomplete observations.

naPattern — an expression or formula object, specifying which returned values are to be regarded as missing.

control — a list of control values for the estimation algorithm to replace the default values returned by the function nlmeControl. Defaults to an empty list.

verbose — an optional logical value. If TRUE information on the evolution of the iterative algorithm is printed. Default is FALSE.

Value

an object of class nlme representing the nonlinear mixed-effects model fit. Generic functions such as print, plot and summary have methods to show the results of the fit. See nlmeObject for the components of the fit. The functions resid, coef, fitted, fixed.effects, and random.effects can be used to extract some of its components.

Note

The function does not do any scaling internally: the optimization will work best when the response is scaled so its variance is of the order of one.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

The model formulation and computational methods are described in Lindstrom, M.J. and Bates, D.M. (1990). The variance-covariance parametrizations are described in Pinheiro and Bates (1996).

Lindstrom, M.J. and Bates, D.M. (1990) "Nonlinear Mixed Effects Models for Repeated Measures Data", Biometrics, 46, 673-687.

Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289-296.

For the different correlation structures, variance functions and links, see 'References' in lme.

See Also

nlmeControl, nlme.nlsList, nlmeObject, nlsList, nlmeStruct, pdClasses, reStruct, varFunc, corClasses, varClasses

Examples

```
fm1 <- nlme(height ~ SSasymp(age, Asym, R0, lrc),
            data = Loblolly,
            fixed = Asym + R0 + lrc ~ 1,
            random = Asym ~ 1,
            start = c(Asym = 103, R0 = -8.5, lrc = -3.3))
summary(fm1)
fm2 <- update(fm1, random = pdDiag(Asym + lrc ~ 1))
summary(fm2)
```

nlme.nlsList *NLME fit from nlsList Object*

Description

If the random effects names defined in random are a subset of the lmList object coefficient names, initial estimates for the covariance matrix of the random effects are obtained (overwriting any values given in random). formula(fixed) and the data argument in the calling sequence used to obtain fixed are passed as the fixed and data arguments to nlme.formula, together with any other additional arguments in the function call. See the documentation on nlme.formula for a description of that function.

Usage

```
## S3 method for class 'nlsList'
nlme(model, data, fixed, random, groups, start, correlation, weights,
     subset, method, na.action, naPattern, control, verbose)
```

Arguments

model	an object inheriting from class "nlsList", representing a list of nls fits with a common model.
data	this argument is included for consistency with the generic function. It is ignored in this method function.
fixed	this argument is included for consistency with the generic function. It is ignored in this method function.
random	an optional one-sided linear formula with no conditioning expression, or a pdMat object with a formula attribute. Multiple levels of grouping are not allowed with this method function. Defaults to a formula consisting of the right hand side of formula(fixed).
groups	an optional one-sided formula of the form ~g1 (single level of nesting) or ~g1/.../gQ (multiple levels of nesting), specifying the partitions of the data over which the random effects vary. g1,...,gQ must evaluate to factors in data. The order of nesting, when multiple levels are present, is taken from left to right (i.e. g1 is the first level, g2 the second, etc.).
start	an optional numeric vector, or list of initial estimates for the fixed effects and random effects. If declared as a numeric vector, it is converted internally to a list with a single component fixed, given by the vector. The fixed component is required, unless the model function inherits from class selfStart, in which case initial values will be derived from a call to nlsList. An optional random component is used to specify initial values for the random effects and should consist of a matrix, or a list of matrices with length equal to the number of grouping levels. Each matrix should have as many rows as the number of groups at the corresponding level and as many columns as the number of random effects in that level.

correlation	an optional corStruct object describing the within-group correlation structure. See the documentation of corClasses for a description of the available corStruct classes. Defaults to NULL, corresponding to no within-group correlations.
weights	an optional varFunc object or one-sided formula describing the within-group heteroscedasticity structure. If given as a formula, it is used as the argument to varFixed, corresponding to fixed variance weights. See the documentation on varClasses for a description of the available varFunc classes. Defaults to NULL, corresponding to homoscedastic within-group errors.
subset	an optional expression indicating the subset of the rows of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
method	a character string. If "REML" the model is fit by maximizing the restricted log-likelihood. If "ML" the log-likelihood is maximized. Defaults to "ML".
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes nlme to print an error message and terminate if there are any incomplete observations.
naPattern	an expression or formula object, specifying which returned values are to be regarded as missing.
control	a list of control values for the estimation algorithm to replace the default values returned by the function nlmeControl. Defaults to an empty list.
verbose	an optional logical value. If TRUE information on the evolution of the iterative algorithm is printed. Default is FALSE.

Value

an object of class nlme representing the linear mixed-effects model fit. Generic functions such as print, plot and summary have methods to show the results of the fit. See nlmeObject for the components of the fit. The functions resid, coef, fitted, fixed.effects, and random.effects can be used to extract some of its components.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

The computational methods follow on the general framework of Lindstrom, M.J. and Bates, D.M. (1988). The model formulation is described in Laird, N.M. and Ware, J.H. (1982). The variance-covariance parametrizations are described in <Pinheiro, J.C. and Bates., D.M. (1996). The different correlation structures available for the correlation argument are described in Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994), Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996), and Venables, W.N. and Ripley, B.D. (2002). The use of variance functions for linear and nonlinear mixed effects models is presented in detail in Davidian, M. and Giltinan, D.M. (1995).

Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day.

Davidian, M. and Giltinan, D.M. (1995) "Nonlinear Mixed Effects Models for Repeated Measurement Data", Chapman and Hall.

Laird, N.M. and Ware, J.H. (1982) "Random-Effects Models for Longitudinal Data", Biometrics, 38, 963-974.

Lindstrom, M.J. and Bates, D.M. (1988) "Newton-Raphson and EM Algorithms for Linear Mixed-Effects Models for Repeated-Measures Data", Journal of the American Statistical Association, 83, 1014-1022.

Littel, R.C., Milliken, G.A., Stroup, W.W., and Wolfinger, R.D. (1996) "SAS Systems for Mixed Models", SAS Institute.

Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289-296.

Venables, W.N. and Ripley, B.D. (2002) "Modern Applied Statistics with S", 4th Edition, Springer-Verlag.

See Also

nlme, lmList, nlmeObject

Examples

```
fm1 <- nlsList(SSasymp, data = Loblolly)
fm2 <- nlme(fm1, random = Asym ~ 1)
summary(fm1)
summary(fm2)
```

---

nlmeControl *Control Values for nlme Fit*

Description

The values supplied in the function call replace the defaults and a list with all possible arguments is returned. The returned list is used as the `control` argument to the `nlme` function.

Usage

```
nlmeControl(maxIter, pnlsMaxIter, msMaxIter, minScale,
            tolerance, niterEM, pnlsTol, msTol,
            returnObject, msVerbose, msWarnNoConv,
            gradHess, apVar, .relStep, minAbsParApVar = 0.05,
            opt = c("nlminb", "nlm"), natural = TRUE, sigma = NULL, ...)
```

Arguments

maxIter	maximum number of iterations for the nlme optimization algorithm. Default is 50.
pnlsMaxIter	maximum number of iterations for the PNLS optimization step inside the nlme optimization. Default is 7.
msMaxIter	maximum number of iterations for nlminb (iter.max) or the nlm (iterlim, from the 10-th step) optimization step inside the nlme optimization. Default is 50 (which may be too small for e.g. for overparametrized cases).
minScale	minimum factor by which to shrink the default step size in an attempt to decrease the sum of squares in the PNLS step. Default 0.001.
tolerance	tolerance for the convergence criterion in the nlme algorithm. Default is 1e-6.
niterEM	number of iterations for the EM algorithm used to refine the initial estimates of the random effects variance-covariance coefficients. Default is 25.
pnlsTol	tolerance for the convergence criterion in PNLS step. Default is 1e-3.
msTol	tolerance for the convergence criterion in nlm, passed as the gradtol argument to the function (see documentation on nlm). Default is 1e-7.
returnObject	a logical value indicating whether the fitted object should be returned when the maximum number of iterations is reached without convergence of the algorithm. Default is FALSE.
msVerbose	a logical value passed as the trace to nlminb(.., control= list(trace = *, ..)) or as argument print.level to nlm(). Default is FALSE.
msWarnNoConv	logical indicating if a warning should be signalled whenever the minimization (by opt) in the LME step does not converge; defaults to TRUE.
gradHess	a logical value indicating whether numerical gradient vectors and Hessian matrices of the log-likelihood function should be used in the nlm optimization. This option is only available when the correlation structure (corStruct) and the variance function structure (varFunc) have no "varying" parameters and the pdMat classes used in the random effects structure are pdSymm (general positive-definite), pdDiag (diagonal), pdIdent (multiple of the identity), or pdCompSymm (compound symmetry). Default is TRUE.
apVar	a logical value indicating whether the approximate covariance matrix of the variance-covariance parameters should be calculated. Default is TRUE.
.relStep	relative step for numerical derivatives calculations. Default is .Machine$double.eps^(1/3).
minAbsParApVar	numeric value - minimum absolute parameter value in the approximate variance calculation. The default is 0.05.
opt	the optimizer to be used, either "nlminb" (the default) or "nlm".
natural	a logical value indicating whether the pdNatural parametrization should be used for general positive-definite matrices (pdSymm) in reStruct, when the approximate covariance matrix of the estimators is calculated. Default is TRUE.
sigma	optionally a positive number to fix the residual error at. If NULL, as by default, or 0, sigma is estimated.
...	Further, named control arguments to be passed to nlminb (apart from trace and iter.max mentioned above), where used (eval.max and those from abs.tol down).

Value

a list with components for each of the possible arguments.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>; the sigma option: Siem Heisterkamp and Bert van Willigen.

See Also

nlme, nlm, optim, nlmeStruct

Examples

```
# decrease the maximum number of iterations and request tracing
nlmeControl(msMaxIter = 20, msVerbose = TRUE)
```

---

nlmeObject *Fitted nlme Object*

Description

An object returned by the nlme function, inheriting from class "nlme", also inheriting from class "lme", and representing a fitted nonlinear mixed-effects model. Objects of this class have methods for the generic functions anova, coef, fitted, fixed.effects, formula, getGroups, getResponse, intervals, logLik, pairs, plot, predict, print, random.effects, residuals, summary, and update.

Value

The following components must be included in a legitimate "nlme" object.

apVar	an approximate covariance matrix for the variance-covariance coefficients. If apVar = FALSE in the control values used in the call to nlme, this component is NULL.
call	a list containing an image of the nlme call that produced the object.
coefficients	a list with two components, fixed and random, where the first is a vector containing the estimated fixed effects and the second is a list of matrices with the estimated random effects for each level of grouping. For each matrix in the random list, the columns refer to the random effects and the rows to the groups.
contrasts	a list of the contrast matrices used to represent factors in the fixed effects formula and/or random effects formula. This information is important for making predictions from a new data frame in which not all levels of the original factors are observed. If no factors are used in the nlme model, this component will be an empty list.

dims	a list with basic dimensions used in the nlme fit, including the components N - the number of observations in the data, Q - the number of grouping levels, qvec - the number of random effects at each level from innermost to outermost (last two values are equal to zero and correspond to the fixed effects and the response), ngrps - the number of groups at each level from innermost to outermost (last two values are one and correspond to the fixed effects and the response), and ncol - the number of columns in the model matrix for each level of grouping from innermost to outermost (last two values are equal to the number of fixed effects and one).
fitted	a data frame with the fitted values as columns. The leftmost column corresponds to the population fixed effects (corresponding to the fixed effects only) and successive columns from left to right correspond to increasing levels of grouping.
fixDF	a list with components X and terms specifying the denominator degrees of freedom for, respectively, t-tests for the individual fixed effects and F-tests for the fixed-effects terms in the models.
groups	a data frame with the grouping factors as columns. The grouping level increases from left to right.
logLik	the (restricted) log-likelihood at convergence.
map	a list with components fmap, rmap, rmapRel, and bmap, specifying various mappings for the fixed and random effects, used to generate predictions from the fitted object.
method	the estimation method: either "ML" for maximum likelihood, or "REML" for restricted maximum likelihood.
modelStruct	an object inheriting from class nlmeStruct, representing a list of mixed-effects model components, such as reStruct, corStruct, and varFunc objects.
numIter	the number of iterations used in the iterative algorithm.
residuals	a data frame with the residuals as columns. The leftmost column corresponds to the population residuals and successive columns from left to right correspond to increasing levels of grouping.
sigma	the estimated within-group error standard deviation.
varFix	an approximate covariance matrix of the fixed effects estimates.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

nlme, nlmeStruct

nlmeStruct *Nonlinear Mixed-Effects Structure*

Description

A nonlinear mixed-effects structure is a list of model components representing different sets of parameters in the nonlinear mixed-effects model. An `nlmeStruct` list must contain at least a `reStruct` object, but may also contain `corStruct` and `varFunc` objects. `NULL` arguments are not included in the `nlmeStruct` list.

Usage

```
nlmeStruct(reStruct, corStruct, varStruct)
```

Arguments

`reStruct`	a `reStruct` representing a random effects structure.
`corStruct`	an optional `corStruct` object, representing a correlation structure. Default is `NULL`.
`varStruct`	an optional `varFunc` object, representing a variance function structure. Default is `NULL`.

Value

a list of model components determining the parameters to be estimated for the associated nonlinear mixed-effects model.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`corClasses`, `nlme`, `residuals.nlmeStruct`, `reStruct`, `varFunc`

Examples

```
nlms1 <- nlmeStruct(reStruct(~age), corAR1(), varPower())
```

nlsList *List of nls Objects with a Common Model*

Description

Data is partitioned according to the levels of the grouping factor defined in model and individual nls fits are obtained for each data partition, using the model defined in model.

Usage

```
nlsList(model, data, start, control, level, subset,
        na.action = na.fail, pool = TRUE, warn.nls = NA)

## S3 method for class 'formula'
nlsList(model, data, start, control, level, subset,
        na.action = na.fail, pool = TRUE, warn.nls = NA)

## S3 method for class 'nlsList'
update(object, model., ..., evaluate = TRUE)
```

Arguments

object	an object inheriting from class nlsList, representing a list of fitted nls objects.
model	either a nonlinear model formula, with the response on the left of a ~ operator and an expression involving parameters, covariates, and a grouping factor separated by the | operator on the right, or a selfStart function. The method function nlsList.selfStart is documented separately.
model.	changes to the model – see update.formula for details.
data	a data frame in which to interpret the variables named in model.
start	an optional named list with initial values for the parameters to be estimated in model. It is passed as the start argument to each nls call and is required when the nonlinear function in model does not inherit from class selfStart.
control	a list of control values passed as the control argument to nls. Defaults to an empty list.
level	an optional integer specifying the level of grouping to be used when multiple nested levels of grouping are present.
subset	an optional expression indicating the subset of the rows of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes nlsList to print an error message and terminate if there are any incomplete observations.

pool	an optional logical value that is preserved as an attribute of the returned value. This will be used as the default for pool in calculations of standard deviations or standard errors for summaries.
warn.nls	logical indicating if nls() errors (all of which are caught by tryCatch) should be signalled as a "summarizing" warning.
...	some methods for this generic require additional arguments. None are used in this method.
evaluate	If TRUE evaluate the new call else return the call.

Details

As nls(.) is called on each sub group, and convergence of these may be problematic, these calls happen with error catching.

Since **nlme** version 3.1-127 (2016-04), all the errors are caught (via tryCatch) and if present, a "summarizing" warning is stored as attribute of the resulting "nlsList" object and signalled unless suppressed by warn.nls = FALSE or currently also when warn.nls = NA (default) *and* getOption("show.error.messages") is false.

nlsList() originally had used try(*) (with its default silent=FALSE) and hence all errors were printed to the console *unless* the global option show.error.messages was set to true. This still works, but has been *deprecated*.

Value

a list of nls objects with as many components as the number of groups defined by the grouping factor. Generic functions such as coef, fixed.effects, lme, pairs, plot, predict, random.effects, summary, and update have methods that can be applied to an nlsList object.

References

Pinheiro, J.C., and Bates, D.M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer.

See Also

nls, nlme.nlsList, nlsList.selfStart, summary.nlsList

Examples

```
fm1 <- nlsList(uptake ~ SSasympOff(conc, Asym, lrc, c0),
   data = CO2, start = c(Asym = 30, lrc = -4.5, c0 = 52))
summary(fm1)
cfm1 <- confint(fm1) # via profiling each % FIXME: only *one* message instead of one *each*
mat.class <- class(matrix(1)) # ("matrix", "array") for R >= 4.0.0;  ("matrix" in older R)
i.ok <- which(vapply(cfm1,
                function(r) identical(class(r), mat.class), NA))
stopifnot(length(i.ok) > 0, !anyNA(match(c(2:4, 6:9, 12), i.ok)))
## where as (some of) the others gave errors during profile re-fitting :
str(cfm1[- i.ok])
```

nlsList.selfStart *nlsList Fit from a selfStart Function*

Description

The response variable and primary covariate in `formula(data)` are used together with `model` to construct the nonlinear model formula. This is used in the `nls` calls and, because a self-starting model function can calculate initial estimates for its parameters from the data, no starting estimates need to be provided.

Usage

```
## S3 method for class 'selfStart'
nlsList(model, data, start, control, level, subset,
        na.action = na.fail, pool = TRUE, warn.nls = NA)
```

Arguments

`model`	a "`selfStart`" model function, which calculates initial estimates for the model parameters from `data`.
`data`	a data frame in which to interpret the variables in `model`. Because no grouping factor can be specified in `model`, `data` must inherit from class "`groupedData`".
`start`	an optional named list with initial values for the parameters to be estimated in `model`. It is passed as the `start` argument to each `nls` call and is required when the nonlinear function in `model` does not inherit from class `selfStart`.
`control`	a list of control values passed as the `control` argument to `nls`. Defaults to an empty list.
`level`	an optional integer specifying the level of grouping to be used when multiple nested levels of grouping are present.
`subset`	an optional expression indicating the subset of the rows of `data` that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default action (`na.fail`) causes `nlsList` to print an error message and terminate if there are any incomplete observations.
`pool`, `warn.nls`	optional `logical`s, see `nlsList`.

Value

a list of `nls` objects with as many components as the number of groups defined by the grouping factor. A `NULL` value is assigned to the components corresponding to clusters for which the `nls` algorithm failed to converge. Generic functions such as `coef`, `fixed.effects`, `lme`, `pairs`, `plot`, `predict`, `random.effects`, `summary`, and `update` have methods that can be applied to an `nlsList` object.

See Also

selfStart, groupedData, nls, nlsList, nlme.nlsList, nlsList.formula

Examples

```
fm1 <- nlsList(SSasympOff, CO2)
summary(fm1)
```

Oats *Split-plot Experiment on Varieties of Oats*

Description

The Oats data frame has 72 rows and 4 columns.

Format

This data frame contains the following columns:

Block an ordered factor with levels VI < V < III < IV < II < I

Variety a factor with levels Golden Rain Marvellous Victory

nitro a numeric vector

yield a numeric vector

Details

These data have been introduced by Yates (1935) as an example of a split-plot design. The treatment structure used in the experiment was a 3×4 full factorial, with three varieties of oats and four concentrations of nitrogen. The experimental units were arranged into six blocks, each with three whole-plots subdivided into four subplots. The varieties of oats were assigned randomly to the whole-plots and the concentrations of nitrogen to the subplots. All four concentrations of nitrogen were used on each whole-plot.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.15)

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S. (4th ed)*, Springer, New York.

`Orthodont` *Growth curve data on an orthdontic measurement*

Description

The `Orthodont` data frame has 108 rows and 4 columns of the change in an orthdontic measurement over time for several young subjects.

Format

This data frame contains the following columns:

distance a numeric vector of distances from the pituitary to the pterygomaxillary fissure (mm). These distances are measured on x-ray images of the skull.

age a numeric vector of ages of the subject (yr).

Subject an ordered factor indicating the subject on which the measurement was made. The levels are labelled `M01` to `M16` for the males and `F01` to `F13` for the females. The ordering is by increasing average distance within sex.

Sex a factor with levels `Male` and `Female`

Details

Investigators at the University of North Carolina Dental School followed the growth of 27 children (16 males, 11 females) from age 8 until age 14. Every two years they measured the distance between the pituitary and the pterygomaxillary fissure, two points that are easily identified on x-ray exposures of the side of the head.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.17)

Potthoff, R. F. and Roy, S. N. (1964), "A generalized multivariate analysis of variance model useful especially for growth curve problems", *Biometrika*, **51**, 313–326.

Examples

```
formula(Orthodont)
plot(Orthodont)
```

Ovary *Counts of Ovarian Follicles*

Description

The Ovary data frame has 308 rows and 3 columns.

Format

This data frame contains the following columns:

Mare an ordered factor indicating the mare on which the measurement is made.

Time time in the estrus cycle. The data were recorded daily from 3 days before ovulation until 3 days after the next ovulation. The measurement times for each mare are scaled so that the ovulations for each mare occur at times 0 and 1.

follicles the number of ovarian follicles greater than 10 mm in diameter.

Details

Pierson and Ginther (1987) report on a study of the number of large ovarian follicles detected in different mares at several times in their estrus cycles.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.18)

Pierson, R. A. and Ginther, O. J. (1987), Follicular population dynamics during the estrus cycle of the mare, *Animal Reproduction Science*, **14**, 219-231.

Oxboys *Heights of Boys in Oxford*

Description

The Oxboys data frame has 234 rows and 4 columns.

Format

This data frame contains the following columns:

Subject an ordered factor giving a unique identifier for each boy in the experiment

age a numeric vector giving the standardized age (dimensionless)

height a numeric vector giving the height of the boy (cm)

Occasion an ordered factor - the result of converting age from a continuous variable to a count so these slightly unbalanced data can be analyzed as balanced.

Details

These data are described in Goldstein (1987) as data on the height of a selection of boys from Oxford, England versus a standardized age.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.19)

Oxide *Variability in Semiconductor Manufacturing*

Description

The `Oxide` data frame has 72 rows and 5 columns.

Format

This data frame contains the following columns:

Source a factor with levels `1` and `2`

Lot a factor giving a unique identifier for each lot.

Wafer a factor giving a unique identifier for each wafer within a lot.

Site a factor with levels `1`, `2`, and `3`

Thickness a numeric vector giving the thickness of the oxide layer.

Details

These data are described in Littell et al. (1996, p. 155) as coming "from a passive data collection study in the semiconductor industry where the objective is to estimate the variance components to determine the assignable causes of the observed variability." The observed response is the thickness of the oxide layer on silicon wafers, measured at three different sites of each of three wafers selected from each of eight lots sampled from the population of lots.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.20)

Littell, R. C., Milliken, G. A., Stroup, W. W. and Wolfinger, R. D. (1996), *SAS System for Mixed Models*, SAS Institute, Cary, NC.

pairs.compareFits *Pairs Plot of compareFits Object*

Description

Scatter plots of the values being compared are generated for each pair of coefficients in x. Different symbols (colors) are used for each object being compared and values corresponding to the same group are joined by a line, to facilitate comparison of fits. If only two coefficients are present, the trellis function xyplot is used; otherwise the trellis function splom is used.

Usage

```
## S3 method for class 'compareFits'
pairs(x, subset, key, ...)
```

Arguments

x	an object of class compareFits.
subset	an optional logical or integer vector specifying which rows of x should be used in the plots. If missing, all rows are used.
key	an optional logical value, or list. If TRUE, a legend is included at the top of the plot indicating which symbols (colors) correspond to which objects being compared. If FALSE, no legend is included. If given as a list, key is passed down as an argument to the trellis function generating the plots (splom or xyplot). Defaults to TRUE.
...	optional arguments passed down to the trellis function generating the plots.

Value

Pairwise scatter plots of the values being compared, with different symbols (colors) used for each object under comparison.

Author(s)

José Pinheiro and Douglas Bates

See Also

compareFits, plot.compareFits, pairs.lme, pairs.lmList, xyplot, splom

Examples

```
example(compareFits) # cF12 <- compareFits(coef(lmList(Orthodont)), .. lme(*))
pairs(cF12)
```

`pairs.lme` *Pairs Plot of an lme Object*

Description

Diagnostic plots for the linear mixed-effects fit are obtained. The `form` argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a `|` operator) always implies that different panels are used for each level of the conditioning factor, according to a Trellis display. The expression on the right hand side of the formula, before a `|` operator, must evaluate to a data frame with at least two columns. If the data frame has two columns, a scatter plot of the two variables is displayed (the Trellis function `xyplot` is used). Otherwise, if more than two columns are present, a scatter plot matrix with pairwise scatter plots of the columns in the data frame is displayed (the Trellis function `splom` is used).

Usage

```
## S3 method for class 'lme'
pairs(x, form, label, id, idLabels, grid, ...)
```

Arguments

`x`	an object inheriting from class `"lme"`, representing a fitted linear mixed-effects model.
`form`	an optional one-sided formula specifying the desired type of plot. Any variable present in the original data frame used to obtain `x` can be referenced. In addition, `x` itself can be referenced in the formula using the symbol `"."`. Conditional expressions on the right of a `|` operator can be used to define separate panels in a Trellis display. The expression on the right hand side of `form`, and to the left of the `|` operator, must evaluate to a data frame with at least two columns. Default is `~ coef(.)` , corresponding to a pairs plot of the coefficients evaluated at the innermost level of nesting.
`label`	an optional character vector of labels for the variables in the pairs plot.
`id`	an optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for an outlier test based on the Mahalanobis distances of the estimated random effects. Groups with random effects distances greater than the $1 - value$ percentile of the appropriate chi-square distribution are identified in the plot using `idLabels`. If given as a one-sided formula, its right hand side must evaluate to a logical, integer, or character vector which is used to identify points in the plot. If missing, no points are identified.
`idLabels`	an optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the points identified according to `id`. If given as a one-sided formula, its right hand side must evaluate to a vector which is converted to character and used to label the identified points. Default is the innermost grouping factor.
`grid`	an optional logical value indicating whether a grid should be added to plot. Default is `FALSE`.
`...`	optional arguments passed to the Trellis plot function.

Value

a diagnostic Trellis plot.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lme, pairs.compareFits, pairs.lmList, xyplot, splom

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)

# scatter plot of coefficients by gender, identifying unusual subjects
pairs(fm1, ~coef(., augFrame = TRUE) | Sex, id = 0.1, adj = -0.5)

# scatter plot of estimated random effects :
pairs(fm1, ~ranef(.))
```

---

pairs.lmList *Pairs Plot of an lmList Object*

Description

Diagnostic plots for the linear model fits corresponding to the x components are obtained. The form argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a | operator) always implies that different panels are used for each level of the conditioning factor, according to a Trellis display. The expression on the right hand side of the formula, before a | operator, must evaluate to a data frame with at least two columns. If the data frame has two columns, a scatter plot of the two variables is displayed (the Trellis function xyplot is used). Otherwise, if more than two columns are present, a scatter plot matrix with pairwise scatter plots of the columns in the data frame is displayed (the Trellis function splom is used).

Usage

```
## S3 method for class 'lmList'
pairs(x, form, label, id, idLabels, grid, ...)
```

Arguments

x	an object inheriting from class "lmList", representing a list of lm objects with a common model.

form	an optional one-sided formula specifying the desired type of plot. Any variable present in the original data frame used to obtain x can be referenced. In addition, x itself can be referenced in the formula using the symbol ".". Conditional expressions on the right of a | operator can be used to define separate panels in a Trellis display. The expression on the right hand side of form, and to the left of the | operator, must evaluate to a data frame with at least two columns. Default is ~ coef(.) , corresponding to a pairs plot of the coefficients of x.
label	an optional character vector of labels for the variables in the pairs plot.
id	an optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for an outlier test based on the Mahalanobis distances of the estimated random effects. Groups with random effects distances greater than the $1 - value$ percentile of the appropriate chi-square distribution are identified in the plot using idLabels. If given as a one-sided formula, its right hand side must evaluate to a logical, integer, or character vector which is used to identify points in the plot. If missing, no points are identified.
idLabels	an optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the points identified according to id. If given as a one-sided formula, its right hand side must evaluate to a vector which is converted to character and used to label the identified points. Default is the innermost grouping factor.
grid	an optional logical value indicating whether a grid should be added to plot. Default is FALSE.
...	optional arguments passed to the Trellis plot function.

Value

a diagnostic Trellis plot.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lmList, pairs.lme, pairs.compareFits, xyplot, splom

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)

# scatter plot of coefficients by gender, identifying unusual subjects
pairs(fm1, ~coef(.) | Sex, id = 0.1, adj = -0.5)

# scatter plot of estimated random effects -- "bivariate Gaussian (?)"
pairs(fm1, ~ranef(.))
```

PBG *Effect of Phenylbiguanide on Blood Pressure*

Description

The PBG data frame has 60 rows and 5 columns.

Format

This data frame contains the following columns:

deltaBP a numeric vector

dose a numeric vector

Run an ordered factor with levels T5 < T4 < T3 < T2 < T1 < P5 < P3 < P2 < P4 < P1

Treatment a factor with levels MDL 72222 Placebo

Rabbit an ordered factor with levels 5 < 3 < 2 < 4 < 1

Details

Data on an experiment to examine the effect of a antagonist MDL 72222 on the change in blood pressure experienced with increasing dosage of phenylbiguanide are described in Ludbrook (1994) and analyzed in Venables and Ripley (2002, section 10.3). Each of five rabbits was exposed to increasing doses of phenylbiguanide after having either a placebo or the HD5-antagonist MDL 72222 administered.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.21)

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S (4th ed)*, Springer, New York.

Ludbrook, J. (1994), Repeated measurements and multiple comparisons in cardiovascular research, *Cardiovascular Research*, **28**, 303-311.

pdBlocked *Positive-Definite Block Diagonal Matrix*

Description

This function is a constructor for the pdBlocked class, representing a positive-definite block-diagonal matrix. Each block-diagonal element of the underlying matrix is itself a positive-definite matrix and is represented internally as an individual pdMat object. When value is numeric(0), a list of uninitialized pdMat objects, a list of one-sided formulas, or a list of vectors of character strings, object is returned as an uninitialized pdBlocked object (with just some of its attributes and its class defined) and needs to have its coefficients assigned later, generally using the coef or matrix replacement functions. If value is a list of initialized pdMat objects, object will be constructed from the list obtained by applying as.matrix to each of the pdMat elements of value. Finally, if value is a list of numeric vectors, they are assumed to represent the unrestricted coefficients of the block-diagonal elements of the underlying positive-definite matrix.

Usage

```
pdBlocked(value, form, nam, data, pdClass)
```

Arguments

value	an optional list with elements to be used as the value argument to other pdMat constructors. These include: pdMat objects, positive-definite matrices, one-sided linear formulas, vectors of character strings, or numeric vectors. All elements in the list must be similar (e.g. all one-sided formulas, or all numeric vectors). Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional list of one-sided linear formulas specifying the row/column names for the block-diagonal elements of the matrix represented by object. Because factors may be present in form, the formulas needs to be evaluated on a data.frame to resolve the names they define. This argument is ignored when value is a list of one-sided formulas. Defaults to NULL.
nam	an optional list of vector of character strings specifying the row/column names for the block-diagonal elements of the matrix represented by object. Each of its components must have length equal to the dimension of the corresponding block-diagonal element and unreplicated elements. This argument is ignored when value is a list of vector of character strings. Defaults to NULL.
data	an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on any factors appearing in the formulas. Defaults to the parent frame from which the function was called.
pdClass	an optional vector of character strings naming the pdMat classes to be assigned to the individual blocks in the underlying matrix. If a single class is specified, it is used for all block-diagonal elements. This argument will only be used when value is missing, or its elements are not pdMat objects. Defaults to "pdSymm".

Value

a pdBlocked object representing a positive-definite block-diagonal matrix, also inheriting from class pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. p. 162.

See Also

as.matrix.pdMat, coef.pdMat, pdClasses, matrix<-.pdMat

Examples

```
pd1 <- pdBlocked(list(diag(1:2), diag(c(0.1, 0.2, 0.3))),
                 nam = list(c("A","B"), c("a1", "a2", "a3")))
pd1
```

---

pdClasses *Positive-Definite Matrix Classes*

Description

Standard classes of positive-definite matrices (pdMat) available in the **nlme** package.

Value

Available standard classes:

pdSymm	general positive-definite matrix, with no additional structure
pdLogChol	general positive-definite matrix, with no additional structure, using a log-Cholesky parameterization
pdDiag	diagonal
pdIdent	multiple of an identity
pdCompSymm	compound symmetry structure (constant diagonal and constant off-diagonal elements)
pdBlocked	block-diagonal matrix, with diagonal blocks of any "atomic" pdMat class
pdNatural	general positive-definite matrix in natural parametrization (i.e. parametrized in terms of standard deviations and correlations). The underlying coefficients are not unrestricted, so this class should NOT be used for optimization.

Note

Users may define their own pdMat classes by specifying a constructor function and, at a minimum, methods for the functions pdConstruct, pdMatrix, and coef. For examples of these functions, see the methods for classes pdSymm and pdDiag.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

pdBlocked, pdCompSymm, pdDiag, pdFactor, pdIdent, pdMat, pdMatrix, pdNatural, pdSymm, pdLogChol

---

pdCompSymm *Positive-Definite Matrix with Compound Symmetry Structure*

Description

This function is a constructor for the pdCompSymm class, representing a positive-definite matrix with compound symmetry structure (constant diagonal and constant off-diagonal elements). The underlying matrix is represented by 2 unrestricted parameters. When value is numeric(0), an uninitialized pdMat object, a one-sided formula, or a vector of character strings, object is returned as an uninitialized pdCompSymm object (with just some of its attributes and its class defined) and needs to have its coefficients assigned later, generally using the coef or matrix replacement functions. If value is an initialized pdMat object, object will be constructed from as.matrix(value). Finally, if value is a numeric vector of length 2, it is assumed to represent the unrestricted coefficients of the underlying positive-definite matrix.

Usage

```
pdCompSymm(value, form, nam, data)
```

Arguments

value	an optional initialization value, which can be any of the following: a pdMat object, a positive-definite matrix, a one-sided linear formula (with variables separated by +), a vector of character strings, or a numeric vector of length 2. Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional one-sided linear formula specifying the row/column names for the matrix represented by object. Because factors may be present in form, the formula needs to be evaluated on a data.frame to resolve the names it defines. This argument is ignored when value is a one-sided formula. Defaults to NULL.
nam	an optional vector of character strings specifying the row/column names for the matrix represented by object. It must have length equal to the dimension of the underlying positive-definite matrix and unreplicated elements. This argument is ignored when value is a vector of character strings. Defaults to NULL.

data — an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.

Value

a pdCompSymm object representing a positive-definite matrix with compound symmetry structure, also inheriting from class pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. p. 161.

See Also

as.matrix.pdMat, coef.pdMat, matrix<-.pdMat, pdClasses

Examples

```
pd1 <- pdCompSymm(diag(3) + 1, nam = c("A","B","C"))
pd1
```

---

pdConstruct *Construct pdMat Objects*

Description

This function is an alternative constructor for the pdMat class associated with object and is mostly used internally in other functions. See the documentation on the principal constructor function, generally with the same name as the pdMat class of object.

Usage

```
pdConstruct(object, value, form, nam, data, ...)
```

Arguments

object	an object inheriting from class pdMat, representing a positive definite matrix.
value	an optional initialization value, which can be any of the following: a pdMat object, a positive-definite matrix, a one-sided linear formula (with variables separated by +), a vector of character strings, or a numeric vector. Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional one-sided linear formula specifying the row/column names for the matrix represented by object. Because factors may be present in form, the formula needs to be evaluated on a data.frame to resolve the names it defines. This argument is ignored when value is a one-sided formula. Defaults to NULL.
nam	an optional vector of character strings specifying the row/column names for the matrix represented by object. It must have length equal to the dimension of the underlying positive-definite matrix and unreplicated elements. This argument is ignored when value is a vector of character strings. Defaults to NULL.
data	an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.
...	optional arguments for some methods.

Value

a pdMat object representing a positive-definite matrix, inheriting from the same classes as object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

pdCompSymm, pdDiag, pdIdent, pdNatural, pdSymm

Examples

```
pd1 <- pdSymm()
pdConstruct(pd1, diag(1:4))
```

---

pdConstruct.pdBlocked *Construct pdBlocked Objects*

Description

This function give an alternative constructor for the pdBlocked class, representing a positive-definite block-diagonal matrix. Each block-diagonal element of the underlying matrix is itself a positive-definite matrix and is represented internally as an individual pdMat object. When value is numeric(0), a list of uninitialized pdMat objects, a list of one-sided formulas, or a list of vectors of character strings, object is returned as an uninitialized pdBlocked object (with just some of its attributes and its class defined) and needs to have its coefficients assigned later, generally using the coef or matrix replacement functions. If value is a list of initialized pdMat objects, object will be constructed from the list obtained by applying as.matrix to each of the pdMat elements of value. Finally, if value is a list of numeric vectors, they are assumed to represent the unrestricted coefficients of the block-diagonal elements of the underlying positive-definite matrix.

Usage

```
## S3 method for class 'pdBlocked'
pdConstruct(object, value, form, nam, data, pdClass,
...)
```

Arguments

object	an object inheriting from class "pdBlocked", representing a positive definite block-diagonal matrix.
value	an optional list with elements to be used as the value argument to other pdMat constructors. These include: pdMat objects, positive-definite matrices, one-sided linear formulas, vectors of character strings, or numeric vectors. All elements in the list must be similar (e.g. all one-sided formulas, or all numeric vectors). Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional list of one-sided linear formula specifying the row/column names for the block-diagonal elements of the matrix represented by object. Because factors may be present in form, the formulas needs to be evaluated on a data.frame to resolve the names they defines. This argument is ignored when value is a list of one-sided formulas. Defaults to NULL.
nam	an optional list of vector of character strings specifying the row/column names for the block-diagonal elements of the matrix represented by object. Each of its components must have length equal to the dimension of the corresponding block-diagonal element and unreplicated elements. This argument is ignored when value is a list of vector of character strings. Defaults to NULL.
data	an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.
pdClass	an optional vector of character strings naming the pdMat classes to be assigned to the individual blocks in the underlying matrix. If a single class is specified, it is used for all block-diagonal elements. This argument will only be used when value is missing, or its elements are not pdMat objects. Defaults to "pdSymm".

... some methods for this generic require additional arguments. None are used in this method.

Value

a pdBlocked object representing a positive-definite block-diagonal matrix, also inheriting from class pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

as.matrix.pdMat, coef.pdMat, pdBlocked, pdClasses, pdConstruct, matrix<-.pdMat

Examples

```
pd1 <- pdBlocked(list(c("A","B"), c("a1", "a2", "a3")))
pdConstruct(pd1, list(diag(1:2), diag(c(0.1, 0.2, 0.3))))
```

---

pdDiag *Diagonal Positive-Definite Matrix*

Description

This function is a constructor for the pdDiag class, representing a diagonal positive-definite matrix. If the matrix associated with object is of dimension n, it is represented by n unrestricted parameters, given by the logarithm of the square-root of the diagonal values. When value is numeric(0), an uninitialized pdMat object, a one-sided formula, or a vector of character strings, object is returned as an uninitialized pdDiag object (with just some of its attributes and its class defined) and needs to have its coefficients assigned later, generally using the coef or matrix replacement functions. If value is an initialized pdMat object, object will be constructed from as.matrix(value). Finally, if value is a numeric vector, it is assumed to represent the unrestricted coefficients of the underlying positive-definite matrix.

Usage

```
pdDiag(value, form, nam, data)
```

Arguments

value	an optional initialization value, which can be any of the following: a pdMat object, a positive-definite matrix, a one-sided linear formula (with variables separated by +), a vector of character strings, or a numeric vector of length equal to the dimension of the underlying positive-definite matrix. Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional one-sided linear formula specifying the row/column names for the matrix represented by object. Because factors may be present in form, the formula needs to be evaluated on a data.frame to resolve the names it defines. This argument is ignored when value is a one-sided formula. Defaults to NULL.
nam	an optional vector of character strings specifying the row/column names for the matrix represented by object. It must have length equal to the dimension of the underlying positive-definite matrix and unreplicated elements. This argument is ignored when value is a vector of character strings. Defaults to NULL.
data	an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.

Value

a pdDiag object representing a diagonal positive-definite matrix, also inheriting from class pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

as.matrix.pdMat, coef.pdMat, pdClasses, matrix<-.pdMat

Examples

```
pd1 <- pdDiag(diag(1:3), nam = c("A","B","C"))
pd1
```

pdFactor *Square-Root Factor of a Positive-Definite Matrix*

Description

A square-root factor of the positive-definite matrix represented by `object` is obtained. Letting Σ denote a positive-definite matrix, a square-root factor of Σ is any square matrix L such that $\Sigma = L'L$. This function extracts L.

Usage

```
pdFactor(object)
```

Arguments

`object`	an object inheriting from class `pdMat`, representing a positive definite matrix, which must have been initialized (i.e. `length(coef(object)) > 0`).

Value

a vector with a square-root factor of the positive-definite matrix associated with `object` stacked column-wise.

Note

This function is used intensively in optimization algorithms and its value is returned as a vector for efficiency reasons. The `pdMatrix` function can be used to obtain square-root factors in matrix form.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`pdMatrix`

Examples

```
pd1 <- pdCompSymm(4 * diag(3) + 1)
pdFactor(pd1)
```

pdFactor.reStruct *Extract Square-Root Factor from Components of an reStruct Object*

Description

This method function extracts square-root factors of the positive-definite matrices corresponding to the pdMat elements of object.

Usage

```
## S3 method for class 'reStruct'
pdFactor(object)
```

Arguments

object	an object inheriting from class "reStruct", representing a random effects structure and consisting of a list of pdMat objects.

Value

a vector with square-root factors of the positive-definite matrices corresponding to the elements of object stacked column-wise.

Note

This function is used intensively in optimization algorithms and its value is returned as a vector for efficiency reasons. The pdMatrix function can be used to obtain square-root factors in matrix form.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

pdFactor, pdMatrix.reStruct, pdFactor.pdMat

Examples

```
rs1 <- reStruct(pdSymm(diag(3), ~age+Sex, data = Orthodont))
pdFactor(rs1)
```

pdIdent *Multiple of the Identity Positive-Definite Matrix*

Description

This function is a constructor for the pdIdent class, representing a multiple of the identity positive-definite matrix. The matrix associated with object is represented by 1 unrestricted parameter, given by the logarithm of the square-root of the diagonal value. When value is numeric(0), an uninitialized pdMat object, a one-sided formula, or a vector of character strings, object is returned as an uninitialized pdIdent object (with just some of its attributes and its class defined) and needs to have its coefficients assigned later, generally using the coef or matrix replacement functions. If value is an initialized pdMat object, object will be constructed from as.matrix(value). Finally, if value is a numeric value, it is assumed to represent the unrestricted coefficient of the underlying positive-definite matrix.

Usage

```
pdIdent(value, form, nam, data)
```

Arguments

value	an optional initialization value, which can be any of the following: a pdMat object, a positive-definite matrix, a one-sided linear formula (with variables separated by +), a vector of character strings, or a numeric value. Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional one-sided linear formula specifying the row/column names for the matrix represented by object. Because factors may be present in form, the formula needs to be evaluated on a data.frame to resolve the names it defines. This argument is ignored when value is a one-sided formula. Defaults to NULL.
nam	an optional vector of character strings specifying the row/column names for the matrix represented by object. It must have length equal to the dimension of the underlying positive-definite matrix and unreplicated elements. This argument is ignored when value is a vector of character strings. Defaults to NULL.
data	an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.

Value

a pdIdent object representing a multiple of the identity positive-definite matrix, also inheriting from class pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

as.matrix.pdMat, coef.pdMat, pdClasses, matrix<-.pdMat

Examples

```
pd1 <- pdIdent(4 * diag(3), nam = c("A","B","C"))
pd1
```

---

pdLogChol *General Positive-Definite Matrix*

Description

This function is a constructor for the pdLogChol class, representing a general positive-definite matrix. If the matrix associated with object is of dimension n, it is represented by $n(n+1)/2$ unrestricted parameters, using the log-Cholesky parametrization described in Pinheiro and Bates (1996).

- When value is numeric(0), an uninitialized pdMat object, a one-sided formula, or a character vector, object is returned as an *uninitialized* pdLogChol object (with just some of its attributes and its class defined) and needs to have its coefficients assigned later, generally using the coef or matrix replacement functions.
- If value is an *initialized* pdMat object, object will be constructed from as.matrix(value).
- Finally, if value is a numeric vector, it is assumed to represent the unrestricted coefficients of the matrix-logarithm parametrization of the underlying positive-definite matrix.

Usage

```
pdLogChol(value, form, nam, data)
```

Arguments

value	an optional initialization value, which can be any of the following: a pdMat object, a positive-definite matrix, a one-sided linear formula (with variables separated by +), a vector of character strings, or a numeric vector. Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional one-sided linear formula specifying the row/column names for the matrix represented by object. Because factors may be present in form, the formula needs to be evaluated on a data frame to resolve the names it defines. This argument is ignored when value is a one-sided formula. Defaults to NULL.

nam — an optional character vector specifying the row/column names for the matrix represented by object. It must have length equal to the dimension of the underlying positive-definite matrix and unreplicated elements. This argument is ignored when value is a character vector. Defaults to NULL.

data — an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.

Details

Internally, the pdLogChol representation of a symmetric positive definite matrix is a vector starting with the logarithms of the diagonal of the Choleski factorization of that matrix followed by its upper triangular portion.

Value

a pdLogChol object representing a general positive-definite matrix, also inheriting from class pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C. and Bates., D.M. (1996) Unconstrained Parametrizations for Variance-Covariance Matrices, *Statistics and Computing* **6**, 289–296.

Pinheiro, J.C., and Bates, D.M. (2000) *Mixed-Effects Models in S and S-PLUS*, Springer.

See Also

as.matrix.pdMat, coef.pdMat, pdClasses, matrix<-.pdMat

Examples

```
(pd1 <- pdLogChol(diag(1:3), nam = c("A","B","C")))

(pd4 <- pdLogChol(1:6))
(pd4c <- chol(pd4)) # -> upper-tri matrix with off-diagonals  4 5 6
pd4c[upper.tri(pd4c)]
log(diag(pd4c)) # 1 2 3
```

pdMat *Positive-Definite Matrix*

Description

This function gives an alternative way of constructing an object inheriting from the pdMat class named in pdClass, or from data.class(object) if object inherits from pdMat, and is mostly used internally in other functions. See the documentation on the principal constructor function, generally with the same name as the pdMat class of object.

Usage

```
pdMat(value, form, nam, data, pdClass)
```

Arguments

value	an optional initialization value, which can be any of the following: a pdMat object, a positive-definite matrix, a one-sided linear formula (with variables separated by +), a vector of character strings, or a numeric vector. Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional one-sided linear formula specifying the row/column names for the matrix represented by object. Because factors may be present in form, the formula needs to be evaluated on a data.frame to resolve the names it defines. This argument is ignored when value is a one-sided formula. Defaults to NULL.
nam	an optional vector of character strings specifying the row/column names for the matrix represented by object. It must have length equal to the dimension of the underlying positive-definite matrix and unreplicated elements. This argument is ignored when value is a vector of character strings. Defaults to NULL.
data	an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.
pdClass	an optional character string naming the pdMat class to be assigned to the returned object. This argument will only be used when value is not a pdMat object. Defaults to "pdSymm".

Value

a pdMat object representing a positive-definite matrix, inheriting from the class named in pdClass, or from class(object), if object inherits from pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

pdClasses, pdCompSymm, pdDiag, pdIdent, pdNatural, pdSymm, reStruct, solve.pdMat, summary.pdMat

Examples

```
pd1 <- pdMat(diag(1:4), pdClass = "pdDiag")
pd1
str(pd1)
```

---

pdMatrix *Extract Matrix or Square-Root Factor from a pdMat Object*

Description

The positive-definite matrix represented by `object`, or a square-root factor of it is obtained. Letting Σ denote a positive-definite matrix, a square-root factor of Σ is any square matrix L such that $\Sigma = L'L$. This function extracts Σ or L.

Usage

```
pdMatrix(object, factor)
```

Arguments

`object`	an object inheriting from class `pdMat`, representing a positive definite matrix.
`factor`	an optional logical value. If `TRUE`, a square-root factor of the positive-definite matrix represented by `object` is returned; else, if `FALSE`, the positive-definite matrix is returned. Defaults to `FALSE`.

Value

if `factor` is `FALSE` the positive-definite matrix represented by `object` is returned; else a square-root of the positive-definite matrix is returned.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. p. 162.

See Also

as.matrix.pdMat, pdClasses, pdFactor, pdMat, pdMatrix.reStruct, corMatrix

Examples

```
pd1 <- pdSymm(diag(1:4))
pdMatrix(pd1)
```

---

pdMatrix.reStruct *Extract Matrix or Square-Root Factor from Components of an reStruct Object*

Description

This method function extracts the positive-definite matrices corresponding to the pdMat elements of object, or square-root factors of the positive-definite matrices.

Usage

```
## S3 method for class 'reStruct'
pdMatrix(object, factor)
```

Arguments

object	an object inheriting from class "reStruct", representing a random effects structure and consisting of a list of pdMat objects.
factor	an optional logical value. If TRUE, square-root factors of the positive-definite matrices represented by the elements of object are returned; else, if FALSE, the positive-definite matrices are returned. Defaults to FALSE.

Value

a list with components given by the positive-definite matrices corresponding to the elements of object, or square-root factors of the positive-definite matrices.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. p. 162.

See Also

as.matrix.reStruct, reStruct, pdMat, pdMatrix, pdMatrix.pdMat

Examples

```
rs1 <- reStruct(pdSymm(diag(3), ~age+Sex, data = Orthodont))
pdMatrix(rs1)
```

---

pdNatural *General Positive-Definite Matrix in Natural Parametrization*

Description

This function is a constructor for the pdNatural class, representing a general positive-definite matrix, using a natural parametrization . If the matrix associated with object is of dimension n, it is represented by $n(n+1)/2$ parameters. Letting σ_{ij} denote the ij-th element of the underlying positive definite matrix and $\rho_{ij} = \sigma_i/\sqrt{\sigma_{ii}\sigma_{jj}},\ i \neq j$ denote the associated "correlations", the "natural" parameters are given by $\sqrt{\sigma_{ii}},\ i = 1, \ldots, n$ and $\log((1+\rho_{ij})/(1-\rho_{ij})),\ i \neq j$. Note that all natural parameters are individually unrestricted, but not jointly unrestricted (meaning that not all unrestricted vectors would give positive-definite matrices). Therefore, this parametrization should NOT be used for optimization. It is mostly used for deriving approximate confidence intervals on parameters following the optimization of an objective function. When value is numeric(0), an uninitialized pdMat object, a one-sided formula, or a vector of character strings, object is returned as an uninitialized pdSymm object (with just some of its attributes and its class defined) and needs to have its coefficients assigned later, generally using the coef or matrix replacement functions. If value is an initialized pdMat object, object will be constructed from as.matrix(value). Finally, if value is a numeric vector, it is assumed to represent the natural parameters of the underlying positive-definite matrix.

Usage

```
pdNatural(value, form, nam, data)
```

Arguments

value	an optional initialization value, which can be any of the following: a pdMat object, a positive-definite matrix, a one-sided linear formula (with variables separated by +), a vector of character strings, or a numeric vector. Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional one-sided linear formula specifying the row/column names for the matrix represented by object. Because factors may be present in form, the formula needs to be evaluated on a data.frame to resolve the names it defines. This argument is ignored when value is a one-sided formula. Defaults to NULL.
nam	an optional vector of character strings specifying the row/column names for the matrix represented by object. It must have length equal to the dimension of the underlying positive-definite matrix and unreplicated elements. This argument is ignored when value is a vector of character strings. Defaults to NULL.
data	an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.

Value

a pdNatural object representing a general positive-definite matrix in natural parametrization, also inheriting from class pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. p. 162.

See Also

as.matrix.pdMat, coef.pdMat, pdClasses, matrix<-.pdMat

Examples

```
pdNatural(diag(1:3))
```

---

pdSymm *General Positive-Definite Matrix*

Description

This function is a constructor for the pdSymm class, representing a general positive-definite matrix. If the matrix associated with object is of dimension n, it is represented by $n(n+1)/2$ unrestricted parameters, using the matrix-logarithm parametrization described in Pinheiro and Bates (1996). When value is numeric(0), an uninitialized pdMat object, a one-sided formula, or a vector of character strings, object is returned as an uninitialized pdSymm object (with just some of its attributes and its class defined) and needs to have its coefficients assigned later, generally using the coef or matrix replacement functions. If value is an initialized pdMat object, object will be constructed from as.matrix(value). Finally, if value is a numeric vector, it is assumed to represent the unrestricted coefficients of the matrix-logarithm parametrization of the underlying positive-definite matrix.

Usage

```
pdSymm(value, form, nam, data)
```

Arguments

value	an optional initialization value, which can be any of the following: a pdMat object, a positive-definite matrix, a one-sided linear formula (with variables separated by +), a vector of character strings, or a numeric vector. Defaults to numeric(0), corresponding to an uninitialized object.
form	an optional one-sided linear formula specifying the row/column names for the matrix represented by object. Because factors may be present in form, the formula needs to be evaluated on a data.frame to resolve the names it defines. This argument is ignored when value is a one-sided formula. Defaults to NULL.
nam	an optional vector of character strings specifying the row/column names for the matrix represented by object. It must have length equal to the dimension of the underlying positive-definite matrix and unreplicated elements. This argument is ignored when value is a vector of character strings. Defaults to NULL.
data	an optional data frame in which to evaluate the variables named in value and form. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying matrix. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.

Value

a pdSymm object representing a general positive-definite matrix, also inheriting from class pdMat.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289-296.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

as.matrix.pdMat, coef.pdMat, pdClasses, matrix<-.pdMat

Examples

```
pd1 <- pdSymm(diag(1:3), nam = c("A","B","C"))
pd1
```

Phenobarb *Phenobarbitol Kinetics*

Description

The Phenobarb data frame has 744 rows and 7 columns.

Format

This data frame contains the following columns:

Subject an ordered factor identifying the infant.

Wt a numeric vector giving the birth weight of the infant (kg).

Apgar an ordered factor giving the 5-minute Apgar score for the infant. This is an indication of health of the newborn infant.

ApgarInd a factor indicating whether the 5-minute Apgar score is < 5 or >= 5.

time a numeric vector giving the time when the sample is drawn or drug administered (hr).

dose a numeric vector giving the dose of drug administered (ug/kg).

conc a numeric vector giving the phenobarbital concentration in the serum (ug/L).

Details

Data from a pharmacokinetics study of phenobarbital in neonatal infants. During the first few days of life the infants receive multiple doses of phenobarbital for prevention of seizures. At irregular intervals blood samples are drawn and serum phenobarbital concentrations are determined. The data were originally given in Grasela and Donn(1985) and are analyzed in Boeckmann, Sheiner and Beal (1994), in Davidian and Giltinan (1995), and in Littell et al. (1996).

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.23)

Davidian, M. and Giltinan, D. M. (1995), *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, London. (section 6.6)

Grasela and Donn (1985), Neonatal population pharmacokinetics of phenobarbital derived from routine clinical data, *Developmental Pharmacology and Therapeutics*, **8**, 374-383.

Boeckmann, A. J., Sheiner, L. B., and Beal, S. L. (1994), *NONMEM Users Guide: Part V*, University of California, San Francisco.

Littell, R. C., Milliken, G. A., Stroup, W. W. and Wolfinger, R. D. (1996), *SAS System for Mixed Models*, SAS Institute, Cary, NC.

phenoModel *Model function for the Phenobarb data*

Description

A model function for a model used with the Phenobarb data. This function uses compiled C code to improve execution speed.

Usage

```
phenoModel(Subject, time, dose, lCl, lV)
```

Arguments

Subject	an integer vector of subject identifiers. These should be sorted in increasing order.
time	numeric. A vector of the times at which the sample was drawn or the drug administered (hr).
dose	numeric. A vector of doses of drug administered (ug/kg).
lCl	numeric. A vector of values of the natural log of the clearance parameter according to Subject and time.
lV	numeric. A vector of values of the natural log of the effective volume of distribution according to Subject and time.

Details

See the details section of Phenobarb for a description of the model function that phenoModel evaluates.

Value

a numeric vector of predicted phenobarbital concentrations.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000) *Mixed-effects Models in S and S-PLUS*, Springer. (section 6.4)

Pixel *X-ray pixel intensities over time*

Description

The `Pixel` data frame has 102 rows and 4 columns of data on the pixel intensities of CT scans of dogs over time

Format

This data frame contains the following columns:

Dog a factor with levels `1` to `10` designating the dog on which the scan was made

Side a factor with levels `L` and `R` designating the side of the dog being scanned

day a numeric vector giving the day post injection of the contrast on which the scan was made

pixel a numeric vector of pixel intensities

Source

Pinheiro, J. C. and Bates, D. M. (2000) *Mixed-effects Models in S and S-PLUS*, Springer.

Examples

```
fm1 <- lme(pixel ~ day + I(day^2), data = Pixel,
           random = list(Dog = ~ day, Side = ~ 1))
summary(fm1)
VarCorr(fm1)
```

plot.ACF *Plot an ACF Object*

Description

an `xyplot` of the autocorrelations versus the lags, with `type = "h"`, is produced. If `alpha > 0`, curves representing the critical limits for a two-sided test of level `alpha` for the autocorrelations are added to the plot.

Usage

```
## S3 method for class 'ACF'
plot(x, alpha, xlab, ylab, grid, ...)
```

Arguments

x	an object inheriting from class ACF, consisting of a data frame with two columns named lag and ACF, representing the autocorrelation values and the corresponding lags.
alpha	an optional numeric value with the significance level for testing if the autocorrelations are zero. Lines corresponding to the lower and upper critical values for a test of level alpha are added to the plot. Default is 0, in which case no lines are plotted.
xlab, ylab	optional character strings with the x- and y-axis labels. Default respectively to "Lag" and "Autocorrelation".
grid	an optional logical value indicating whether a grid should be added to plot. Default is FALSE.
...	optional arguments passed to the xyplot function.

Value

an xyplot Trellis plot.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

ACF, xyplot

Examples

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary)
plot(ACF(fm1, maxLag = 10), alpha = 0.01)
```

---

plot.augPred *Plot an augPred Object*

Description

A Trellis xyplot of predictions versus the primary covariate is generated, with a different panel for each value of the grouping factor. Predicted values are joined by lines, with different line types (colors) being used for each level of grouping. Original observations are represented by circles.

Usage

```
## S3 method for class 'augPred'
plot(x, key, grid, ...)
```

Arguments

x	an object of class "augPred".
key	an optional logical value, or list. If TRUE, a legend is included at the top of the plot indicating which symbols (colors) correspond to which prediction levels. If FALSE, no legend is included. If given as a list, key is passed down as an argument to the trellis function generating the plots (xyplot). Defaults to TRUE.
grid	an optional logical value indicating whether a grid should be added to plot. Default is FALSE.
...	optional arguments passed down to the trellis function generating the plots.

Value

A Trellis plot of predictions versus the primary covariate, with panels determined by the grouping factor.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

augPred, xyplot

Examples

```
fm1 <- lme(Orthodont)
plot(augPred(fm1, level = 0:1, length.out = 2))
```

---

plot.compareFits *Plot a compareFits Object*

Description

A Trellis dotplot of the values being compared, with different rows per group, is generated, with a different panel for each coefficient. Different symbols (colors) are used for each object being compared.

Usage

```
## S3 method for class 'compareFits'
plot(x, subset, key, mark, ...)
```

Arguments

x	an object of class "compareFits".
subset	an optional logical or integer vector specifying which rows of x should be used in the plots. If missing, all rows are used.
key	an optional logical value, or list. If TRUE, a legend is included at the top of the plot indicating which symbols (colors) correspond to which objects being compared. If FALSE, no legend is included. If given as a list, key is passed down as an argument to the trellis function generating the plots (dotplot). Defaults to TRUE.
mark	an optional numeric vector, of length equal to the number of coefficients being compared, indicating where vertical lines should be drawn in the plots. If missing, no lines are drawn.
...	optional arguments passed down to the trellis function generating the plots.

Value

A Trellis dotplot of the values being compared, with rows determined by the groups and panels by the coefficients.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

compareFits, pairs.compareFits, dotplot

Examples

```
example(compareFits) # cF12 <- compareFits(coef(lmList(Orthodont)), .. lme(*))
plot(cF12)
```

---

plot.gls *Plot a gls Object*

Description

Diagnostic plots for the linear model fit are obtained. The form argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a | operator) always implies that different panels are used for each level of the conditioning factor, according to a Trellis display. If form is a one-sided formula, histograms of the variable on the right hand side of the formula, before a | operator, are displayed (the Trellis function histogram is used). If form is two-sided and both its left and right hand side variables are numeric, scatter plots are displayed (the Trellis function xyplot is used). Finally, if form is two-sided and its left had side variable is a factor, box-plots of the right hand side variable by the levels of the left hand side variable are displayed (the Trellis function bwplot is used).

Usage

```
## S3 method for class 'gls'
plot(x, form, abline, id, idLabels, idResType, grid, ...)
```

Arguments

x	an object inheriting from class "gls", representing a generalized least squares fitted linear model.
form	an optional formula specifying the desired type of plot. Any variable present in the original data frame used to obtain x can be referenced. In addition, x itself can be referenced in the formula using the symbol ".". Conditional expressions on the right of a | operator can be used to define separate panels in a Trellis display. Default is resid(., type = "p") ~ fitted(.) , corresponding to a plot of the standardized residuals versus fitted values, both evaluated at the innermost level of nesting.
abline	an optional numeric value, or numeric vector of length two. If given as a single value, a horizontal line will be added to the plot at that coordinate; else, if given as a vector, its values are used as the intercept and slope for a line added to the plot. If missing, no lines are added to the plot.
id	an optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for a two-sided outlier test for the standardized residuals. Observations with absolute standardized residuals greater than the $1 - value/2$ quantile of the standard normal distribution are identified in the plot using idLabels. If given as a one-sided formula, its right hand side must evaluate to a logical, integer, or character vector which is used to identify observations in the plot. If missing, no observations are identified.
idLabels	an optional vector, or one-sided formula. If given as a vector, it is converted to character mode and used to label the observations identified according to id. If given as a one-sided formula, its right hand side must evaluate to a vector which is converted to character mode and used to label the identified observations. Default is the innermost grouping factor.
idResType	an optional character string specifying the type of residuals to be used in identifying outliers, when id is a numeric value. If "pearson", the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if "normalized", the normalized residuals (standardized residuals premultiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to "pearson".
grid	an optional logical value indicating whether a grid should be added to plot. Default depends on the type of Trellis plot used: if xyplot defaults to TRUE, else defaults to FALSE.
...	optional arguments passed to the Trellis plot function.

Value

a diagnostic Trellis plot.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gls, xyplot, bwplot, histogram

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
# standardized residuals versus fitted values by Mare
plot(fm1, resid(., type = "p") ~ fitted(.) | Mare, abline = 0)
# box-plots of residuals by Mare
plot(fm1, Mare ~ resid(.))
# observed versus fitted values by Mare
plot(fm1, follicles ~ fitted(.) | Mare, abline = c(0,1))
```

---

plot.intervals.lmList *Plot lmList Confidence Intervals*

Description

A Trellis dot-plot of the confidence intervals on the linear model coefficients is generated, with a different panel for each coefficient. Rows in the dot-plot correspond to the names of the lm components of the lmList object used to produce x. The lower and upper confidence limits are connected by a line segment and the estimated coefficients are marked with a "+".

This is based on function dotplot() from package **lattice**.

Usage

```
## S3 method for class 'intervals.lmList'
plot(x, xlab = "", ylab = attr(x, "groupsName"),
     strip = function(...) strip.default(..., style = 1),
     ...)
```

Arguments

x	an object inheriting from class "intervals.lmList", representing confidence intervals and estimates for the coefficients in the lm components of the lmList object used to produce x.
xlab, ylab	axis labels, each with a sensible default.
strip	a function or FALSE, see dotplot() from package **lattice**.
...	optional arguments passed to the dotplot function (see above).

Value

a Trellis plot with the confidence intervals on the coefficients of the individual `lm` components of the `lmList` that generated `x`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`intervals.lmList`, `lmList`, `dotplot`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
plot(intervals(fm1))
```

---

plot.lme *Plot an lme or nls object*

Description

Diagnostic plots for the linear mixed-effects fit are obtained. The `form` argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a `|` operator) always implies that different panels are used for each level of the conditioning factor, according to a Trellis display. If `form` is a one-sided formula, histograms of the variable on the right hand side of the formula, before a `|` operator, are displayed (the Trellis function `histogram` is used). If `form` is two-sided and both its left and right hand side variables are numeric, scatter plots are displayed (the Trellis function `xyplot` is used). Finally, if `form` is two-sided and its left had side variable is a factor, box-plots of the right hand side variable by the levels of the left hand side variable are displayed (the Trellis function `bwplot` is used).

Usage

```
## S3 method for class 'lme'
plot(x, form, abline, id, idLabels, idResType, grid, ...)
## S3 method for class 'nls'
plot(x, form, abline, id, idLabels, idResType, grid, ...)
```

Arguments

`x`	an object inheriting from class `"lme"`, representing a fitted linear mixed-effects model, or from `nls`, representing an fitted nonlinear least squares model.
`form`	an optional formula specifying the desired type of plot. Any variable present in the original data frame used to obtain `x` can be referenced. In addition, `x` itself can be referenced in the formula using the symbol `"."`. Conditional expressions

on the right of a | operator can be used to define separate panels in a Trellis display. Default is `resid(., type = "p") ~ fitted(.)`, corresponding to a plot of the standardized residuals versus fitted values, both evaluated at the innermost level of nesting.

`abline` an optional numeric value, or numeric vector of length two. If given as a single value, a horizontal line will be added to the plot at that coordinate; else, if given as a vector, its values are used as the intercept and slope for a line added to the plot. If missing, no lines are added to the plot.

`id` an optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for a two-sided outlier test for the standardized, or normalized residuals. Observations with absolute standardized (normalized) residuals greater than the $1 - value/2$ quantile of the standard normal distribution are identified in the plot using `idLabels`. If given as a one-sided formula, its right hand side must evaluate to a logical, integer, or character vector which is used to identify observations in the plot. If missing, no observations are identified.

`idLabels` an optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the observations identified according to `id`. If given as a one-sided formula, its right hand side must evaluate to a vector which is converted to character and used to label the identified observations. Default is the innermost grouping factor.

`idResType` an optional character string specifying the type of residuals to be used in identifying outliers, when `id` is a numeric value. If `"pearson"`, the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if `"normalized"`, the normalized residuals (standardized residuals premultiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to `"pearson"`.

`grid` an optional logical value indicating whether a grid should be added to plot. Default depends on the type of Trellis plot used: if `xyplot` defaults to `TRUE`, else defaults to `FALSE`.

`...` optional arguments passed to the Trellis plot function.

Value

a diagnostic Trellis plot.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lme, xyplot, bwplot, histogram

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
# standardized residuals versus fitted values by gender
```

```
plot(fm1, resid(., type = "p") ~ fitted(.) | Sex, abline = 0)
# box-plots of residuals by Subject
plot(fm1, Subject ~ resid(.))
# observed versus fitted values by Subject
plot(fm1, distance ~ fitted(.) | Subject, abline = c(0,1))
```

`plot.lmList` *Plot an lmList Object*

Description

Diagnostic plots for the linear model fits corresponding to the `x` components are obtained. The `form` argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a `|` operator) always implies that different panels are used for each level of the conditioning factor, according to a Trellis display. If `form` is a one-sided formula, histograms of the variable on the right hand side of the formula, before a `|` operator, are displayed (the Trellis function `histogram` is used). If `form` is two-sided and both its left and right hand side variables are numeric, scatter plots are displayed (the Trellis function `xyplot` is used). Finally, if `form` is two-sided and its left had side variable is a factor, box-plots of the right hand side variable by the levels of the left hand side variable are displayed (the Trellis function `bwplot` is used).

Usage

```
## S3 method for class 'lmList'
plot(x, form, abline, id, idLabels, grid, ...)
```

Arguments

`x`	an object inheriting from class "`lmList`", representing a list of `lm` objects with a common model.
`form`	an optional formula specifying the desired type of plot. Any variable present in the original data frame used to obtain `x` can be referenced. In addition, `x` itself can be referenced in the formula using the symbol "`.`". Conditional expressions on the right of a `|` operator can be used to define separate panels in a Trellis display. Default is `resid(., type = "pool") ~ fitted(.)`, corresponding to a plot of the standardized residuals (using a pooled estimate for the residual standard error) versus fitted values.
`abline`	an optional numeric value, or numeric vector of length two. If given as a single value, a horizontal line will be added to the plot at that coordinate; else, if given as a vector, its values are used as the intercept and slope for a line added to the plot. If missing, no lines are added to the plot.
`id`	an optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for a two-sided outlier test for the standardized residuals. Observations with absolute standardized residuals greater than the $1 - value/2$ quantile of the standard normal distribution are identified in the plot using `idLabels`. If given as a one-sided formula, its right hand side must evaluate to a logical, integer, or character vector which is used to identify observations in the plot. If missing, no observations are identified.

idLabels	an optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the observations identified according to id. If given as a one-sided formula, its right hand side must evaluate to a vector which is converted to character and used to label the identified observations. Default is getGroups(x).
grid	an optional logical value indicating whether a grid should be added to plot. Default depends on the type of Trellis plot used: if xyplot defaults to TRUE, else defaults to FALSE.
...	optional arguments passed to the Trellis plot function.

Value

a diagnostic Trellis plot.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lmList,predict.lm, xyplot, bwplot, histogram

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
# standardized residuals versus fitted values by gender
plot(fm1, resid(., type = "pool") ~ fitted(.) | Sex, abline = 0, id = 0.05)
# box-plots of residuals by Subject
plot(fm1, Subject ~ resid(.))
# observed versus fitted values by Subject
plot(fm1, distance ~ fitted(.) | Subject, abline = c(0,1))
```

---

plot.nffGroupedData *Plot an nffGroupedData Object*

Description

A Trellis dot-plot of the response by group is generated. If outer variables are specified, the combination of their levels are used to determine the panels of the Trellis display. The Trellis function dotplot is used.

Usage

```
## S3 method for class 'nffGroupedData'
plot(x, outer, inner, innerGroups, xlab, ylab, strip, panel, key,
     grid, ...)
```

Arguments

x	an object inheriting from class nffGroupedData, representing a groupedData object with a factor primary covariate and a single grouping level.
outer	an optional logical value or one-sided formula, indicating covariates that are outer to the grouping factor, which are used to determine the panels of the Trellis plot. If equal to TRUE, attr(object, "outer") is used to indicate the outer covariates. An outer covariate is invariant within the sets of rows defined by the grouping factor. Ordering of the groups is done in such a way as to preserve adjacency of groups with the same value of the outer variables. Defaults to NULL, meaning that no outer covariates are to be used.
inner	an optional logical value or one-sided formula, indicating a covariate that is inner to the grouping factor, which is used to associate points within each panel of the Trellis plot. If equal to TRUE, attr(object, "inner") is used to indicate the inner covariate. An inner covariate can change within the sets of rows defined by the grouping factor. Defaults to NULL, meaning that no inner covariate is present.
innerGroups	an optional one-sided formula specifying a factor to be used for grouping the levels of the inner covariate. Different colors, or symbols, are used for each level of the innerGroups factor. Default is NULL, meaning that no innerGroups covariate is present.
xlab	an optional character string with the label for the horizontal axis. Default is the y elements of attr(object, "labels") and attr(object, "units") pasted together.
ylab	an optional character string with the label for the vertical axis. Default is the grouping factor name.
strip	an optional function passed as the strip argument to the dotplot function. Default is strip.default(..., style = 1) (see trellis.args).
panel	an optional function used to generate the individual panels in the Trellis display, passed as the panel argument to the dotplot function.
key	an optional logical function or function. If TRUE and either inner or innerGroups are non-NULL, a legend for the different inner (innerGroups) levels is included at the top of the plot. If given as a function, it is passed as the key argument to the dotplot function. Default is TRUE is either inner or innerGroups are non-NULL and FALSE otherwise.
grid	this argument is included for consistency with the plot.nfnGroupedData method calling sequence. It is ignored in this method function.
...	optional arguments passed to the dotplot function.

Value

a Trellis dot-plot of the response by group.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Bates, D.M. and Pinheiro, J.C. (1997), "Software Design for Longitudinal Data", in "Modelling Longitudinal and Spatially Correlated Data: Methods, Applications and Future Directions", T.G. Gregoire (ed.), Springer-Verlag, New York.

See Also

groupedData, dotplot

Examples

```
plot(Machines)
plot(Machines, inner = TRUE)
```

---

plot.nfnGroupedData *Plot an nfnGroupedData Object*

Description

A Trellis plot of the response versus the primary covariate is generated. If outer variables are specified, the combination of their levels are used to determine the panels of the Trellis display. Otherwise, the levels of the grouping variable determine the panels. A scatter plot of the response versus the primary covariate is displayed in each panel, with observations corresponding to same inner group joined by line segments. The Trellis function `xyplot` is used.

Usage

```
## S3 method for class 'nfnGroupedData'
plot(x, outer, inner, innerGroups, xlab, ylab, strip, aspect, panel,
     key, grid, ...)
```

Arguments

`x`	an object inheriting from class `nfnGroupedData`, representing a `groupedData` object with a numeric primary covariate and a single grouping level.
`outer`	an optional logical value or one-sided formula, indicating covariates that are outer to the grouping factor, which are used to determine the panels of the Trellis plot. If equal to `TRUE`, `attr(object, "outer")` is used to indicate the outer covariates. An outer covariate is invariant within the sets of rows defined by the grouping factor. Ordering of the groups is done in such a way as to preserve adjacency of groups with the same value of the outer variables. Defaults to `NULL`, meaning that no outer covariates are to be used.
`inner`	an optional logical value or one-sided formula, indicating a covariate that is inner to the grouping factor, which is used to associate points within each panel of the Trellis plot. If equal to `TRUE`, `attr(object, "inner")` is used to indicate the inner covariate. An inner covariate can change within the sets of rows defined by the grouping factor. Defaults to `NULL`, meaning that no inner covariate is present.

innerGroups	an optional one-sided formula specifying a factor to be used for grouping the levels of the inner covariate. Different colors, or line types, are used for each level of the innerGroups factor. Default is NULL, meaning that no innerGroups covariate is present.
xlab, ylab	optional character strings with the labels for the plot. Default is the corresponding elements of attr(object, "labels") and attr(object, "units") pasted together.
strip	an optional function passed as the strip argument to the xyplot function. Default is strip.default(..., style = 1) (see trellis.args).
aspect	an optional character string indicating the aspect ratio for the plot passed as the aspect argument to the xyplot function. Default is "xy" (see trellis.args).
panel	an optional function used to generate the individual panels in the Trellis display, passed as the panel argument to the xyplot function.
key	an optional logical function or function. If TRUE and innerGroups is non-NULL, a legend for the different innerGroups levels is included at the top of the plot. If given as a function, it is passed as the key argument to the xyplot function. Default is TRUE if innerGroups is non-NULL and FALSE otherwise.
grid	an optional logical value indicating whether a grid should be added to plot. Default is TRUE.
...	optional arguments passed to the xyplot function.

Value

a Trellis plot of the response versus the primary covariate.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Bates, D.M. and Pinheiro, J.C. (1997), "Software Design for Longitudinal Data", in "Modelling Longitudinal and Spatially Correlated Data: Methods, Applications and Future Directions", T.G. Gregoire (ed.), Springer-Verlag, New York.

See Also

groupedData, xyplot

Examples

```
# different panels per Subject
plot(Orthodont)
# different panels per gender
plot(Orthodont, outer = TRUE)
```

`plot.nmGroupedData` *Plot an nmGroupedData Object*

Description

The `groupedData` object is summarized by the values of the `displayLevel` grouping factor (or the combination of its values and the values of the covariate indicated in `preserve`, if any is present). The collapsed data is used to produce a new `groupedData` object, with grouping factor given by the `displayLevel` factor, which is plotted using the appropriate `plot` method for `groupedData` objects with single level of grouping.

Usage

```
## S3 method for class 'nmGroupedData'
plot(x, collapseLevel, displayLevel, outer, inner,
    preserve, FUN, subset, key, grid, ...)
```

Arguments

`x`	an object inheriting from class `nmGroupedData`, representing a `groupedData` object with multiple grouping factors.
`collapseLevel`	an optional positive integer or character string indicating the grouping level to use when collapsing the data. Level values increase from outermost to innermost grouping. Default is the highest or innermost level of grouping.
`displayLevel`	an optional positive integer or character string indicating the grouping level to use for determining the panels in the Trellis display, when `outer` is missing. Default is `collapseLevel`.
`outer`	an optional logical value or one-sided formula, indicating covariates that are outer to the `displayLevel` grouping factor, which are used to determine the panels of the Trellis plot. If equal to `TRUE`, the `displayLevel` element `attr(object, "outer")` is used to indicate the outer covariates. An outer covariate is invariant within the sets of rows defined by the grouping factor. Ordering of the groups is done in such a way as to preserve adjacency of groups with the same value of the outer variables. Defaults to `NULL`, meaning that no outer covariates are to be used.
`inner`	an optional logical value or one-sided formula, indicating a covariate that is inner to the `displayLevel` grouping factor, which is used to associate points within each panel of the Trellis plot. If equal to `TRUE`, `attr(object, "outer")` is used to indicate the inner covariate. An inner covariate can change within the sets of rows defined by the grouping factor. Defaults to `NULL`, meaning that no inner covariate is present.
`preserve`	an optional one-sided formula indicating a covariate whose levels should be preserved when collapsing the data according to the `collapseLevel` grouping factor. The collapsing factor is obtained by pasting together the levels of the `collapseLevel` grouping factor and the values of the covariate to be preserved. Default is `NULL`, meaning that no covariates need to be preserved.

FUN	an optional summary function or a list of summary functions to be used for collapsing the data. The function or functions are applied only to variables in object that vary within the groups defined by collapseLevel. Invariant variables are always summarized by group using the unique value that they assume within that group. If FUN is a single function it will be applied to each non-invariant variable by group to produce the summary for that variable. If FUN is a list of functions, the names in the list should designate classes of variables in the data such as ordered, factor, or numeric. The indicated function will be applied to any non-invariant variables of that class. The default functions to be used are mean for numeric factors, and Mode for both factor and ordered. The Mode function, defined internally in gsummary, returns the modal or most popular value of the variable. It is different from the mode function that returns the S-language mode of the variable.
subset	an optional named list. Names can be either positive integers representing grouping levels, or names of grouping factors. Each element in the list is a vector indicating the levels of the corresponding grouping factor to be used for plotting the data. Default is NULL, meaning that all levels are used.
key	an optional logical value, or list. If TRUE, a legend is included at the top of the plot indicating which symbols (colors) correspond to which prediction levels. If FALSE, no legend is included. If given as a list, key is passed down as an argument to the trellis function generating the plots (xyplot). Defaults to TRUE.
grid	an optional logical value indicating whether a grid should be added to plot. Default is TRUE.
...	optional arguments passed to the Trellis plot function.

Value

a Trellis display of the data collapsed over the values of the collapseLevel grouping factor and grouped according to the displayLevel grouping factor.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Bates, D.M. and Pinheiro, J.C. (1997), "Software Design for Longitudinal Data", in "Modelling Longitudinal and Spatially Correlated Data: Methods, Applications and Future Directions", T.G. Gregoire (ed.), Springer-Verlag, New York.

See Also

groupedData, collapse.groupedData, plot.nfnGroupedData, plot.nffGroupedData

Examples

```
# no collapsing, panels by Dog
plot(Pixel, displayLevel = "Dog", inner = ~Side)
# collapsing by Dog, preserving day
plot(Pixel, collapseLevel = "Dog", preserve = ~day)
```

`plot.ranef.lme` *Plot a ranef.lme Object*

Description

Plots (class `"Trellis"` from package **lattice**) of the random effects from linear mixed effects model, i.e., the result of `ranef(lme(*))` (of class `"ranef.lme"`).

Usage

```
## S3 method for class 'ranef.lme'
plot(x, form = NULL, omitFixed = TRUE, level = Q,
     grid = TRUE, control, xlab, ylab, strip,
     ...)
```

Arguments

`x`	an object inheriting from class `"ranef.lme"`, representing the estimated coefficients or estimated random effects for the `lme` object from which it was produced.
`form`	an optional formula specifying the desired type of plot. • If given as a one-sided formula, a `dotplot()` of the estimated random effects (coefficients) grouped according to all combinations of the levels of the factors named in `form` is returned. • If given as a two-sided formula (or by default, `NULL`), an `xyplot()` Trellis display of the random effect (coefficient) versus the named covariates is returned. In `NULL` case the row names of the random effects (coefficients) are used (as covariates). See also 'Details:'.
`omitFixed`	an optional logical value indicating whether columns with values that are constant across groups should be omitted. Default is `TRUE`.
`level`	an optional integer value giving the level of grouping to be used for `x`. Only used when `x` is a list with different components for each grouping level. Defaults to the highest or innermost level of grouping.
`grid`	an optional logical value indicating whether a grid should be added to plot. Only applies to plots associated with two-sided formulas in `form`. Default is `TRUE`.

control	an optional list with control values for the plot, when form is given as a two-sided formula. The control values are referenced by name in the control list and only the ones to be modified from the default need to be specified. Available values include: drawLine, a logical value indicating whether a loess smoother should be added to the scatter plots and a line connecting the medians should be added to the boxplots (default is TRUE); span.loess, used as the span argument in the call to panel.loess (default is 2/3); degree.loess, used as the degree argument in the call to panel.loess (default is 1); cex.axis, the character expansion factor for the x-axis (default is 0.8); srt.axis, the rotation factor for the x-axis (default is 0); and mgp.axis, the margin parameters for the x-axis (default is c(2, 0.5, 0)).
xlab, ylab	axis labels, each with a sensible default.
strip	a function or FALSE, see dotplot() from package **lattice**.
...	optional arguments passed to the Trellis dotplot function.

Details

If form is missing, or is given as a one-sided formula, a Trellis dot-plot (via dotplot() from pkg **lattice**) of the random effects is generated, with a different panel for each random effect (coefficient). Rows in the dot-plot are determined by the form argument (if not missing) or by the row names of the random effects (coefficients). Single factors (~g) or crossed factors (~g1*g2) are allowed. For a single factor, its levels determine the dot-plot rows (with possibly multiple dots per row); otherwise, if form specifies a crossing of factors, the dot-plot rows are determined by all combinations of the levels of the individual factors in the formula.

If form is a two-sided formula, the left hand side must be a single random effect (coefficient) and the right hand side is formed by covariates in x separated by +. An xyplot() Trellis display is generated, with a different panel for each variable listed in the right hand side of form. Scatter plots are generated for numeric variables and boxplots are generated for categorical (factor or ordered) variables.

Value

a Trellis plot of the estimated random-effects (coefficients) versus covariates, or groups.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

ranef.lme, lme, dotplot.

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
plot(ranef(fm1))
fm1RE <- ranef(fm1, augFrame = TRUE)
plot(fm1RE, form = ~ Sex)
plot(fm1RE, form = age ~ Sex) # "connected" boxplots
```

`plot.ranef.lmList` *Plot a ranef.lmList Object*

Description

If `form` is missing, or is given as a one-sided formula, a Trellis dot-plot of the random effects is generated, with a different panel for each random effect (coefficient). Rows in the dot-plot are determined by the `form` argument (if not missing) or by the row names of the random effects (coefficients). If a single factor is specified in `form`, its levels determine the dot-plot rows (with possibly multiple dots per row); otherwise, if `form` specifies a crossing of factors, the dot-plot rows are determined by all combinations of the levels of the individual factors in the formula. The Trellis function `dotplot` is used in this method function.

If `form` is a two-sided formula, a Trellis display is generated, with a different panel for each variable listed in the right hand side of `form`. Scatter plots are generated for numeric variables and boxplots are generated for categorical (`factor` or `ordered`) variables.

Usage

```
## S3 method for class 'ranef.lmList'
plot(x, form, grid, control, ...)
```

Arguments

`x`	an object inheriting from class "`ranef.lmList`", representing the estimated coefficients or estimated random effects for the `lmList` object from which it was produced.
`form`	an optional formula specifying the desired type of plot. If given as a one-sided formula, a `dotplot` of the estimated random effects (coefficients) grouped according to all combinations of the levels of the factors named in `form` is returned. Single factors (`~g`) or crossed factors (`~g1*g2`) are allowed. If given as a two-sided formula, the left hand side must be a single random effects (coefficient) and the right hand side is formed by covariates in `x` separated by `+`. A Trellis display of the random effect (coefficient) versus the named covariates is returned in this case. Default is `NULL`, in which case the row names of the random effects (coefficients) are used.
`grid`	an optional logical value indicating whether a grid should be added to plot. Only applies to plots associated with two-sided formulas in `form`. Default is `FALSE`.
`control`	an optional list with control values for the plot, when `form` is given as a two-sided formula. The control values are referenced by name in the `control` list and only the ones to be modified from the default need to be specified. Available values include: `drawLine`, a logical value indicating whether a `loess` smoother should be added to the scatter plots and a line connecting the medians should be added to the boxplots (default is `TRUE`); `span.loess`, used as the `span` argument in the call to `panel.loess` (default is `2/3`); `degree.loess`, used as the `degree` argument in the call to `panel.loess` (default is `1`); `cex.axis`, the character expansion factor for the x-axis (default is `0.8`); `srt.axis`, the rotation factor

for the x-axis (default is `0`); and `mgp.axis`, the margin parameters for the x-axis (default is `c(2, 0.5, 0)`).

`...` optional arguments passed to the Trellis `dotplot` function.

Value

a Trellis plot of the estimated random-effects (coefficients) versus covariates, or groups.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lmList`, `dotplot`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
plot(ranef(fm1))
fm1RE <- ranef(fm1, augFrame = TRUE)
plot(fm1RE, form = ~ Sex)
plot(fm1RE, form = age ~ Sex)
```

---

plot.Variogram *Plot a Variogram Object*

Description

an `xyplot` of the semi-variogram versus the distances is produced. If `smooth = TRUE`, a `loess` smoother is added to the plot. If `showModel = TRUE` and `x` includes an `"modelVariog"` attribute, the corresponding semi-variogram is added to the plot.

Usage

```
## S3 method for class 'Variogram'
plot(x, smooth, showModel, sigma, span, xlab,
     ylab, type, ylim, grid, ...)
```

Arguments

`x` an object inheriting from class `"Variogram"`, consisting of a data frame with two columns named `variog` and `dist`, representing the semi-variogram values and the corresponding distances.

`smooth` an optional logical value controlling whether a `loess` smoother should be added to the plot. Defaults to `TRUE`, when `showModel` is `FALSE`.

showModel	an optional logical value controlling whether the semi-variogram corresponding to an "modelVariog" attribute of x, if any is present, should be added to the plot. Defaults to TRUE, when the "modelVariog" attribute is present.
sigma	an optional numeric value used as the height of a horizontal line displayed in the plot. Can be used to represent the process standard deviation. Default is NULL, implying that no horizontal line is drawn.
span	an optional numeric value with the smoothing parameter for the loess fit. Default is 0.6.
xlab, ylab	optional character strings with the x- and y-axis labels. Default respectively to "Distance" and "SemiVariogram".
type	an optional character indicating the type of plot. Defaults to "p".
ylim	an optional numeric vector with the limits for the y-axis. Defaults to c(0, max(x$variog)).
grid	an optional logical value indicating whether a grid should be added to plot. Default is FALSE.
...	optional arguments passed to the Trellis xyplot function.

Value

an xyplot Trellis plot.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

Variogram, xyplot, loess

Examples

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary)
plot(Variogram(fm1, form = ~ Time | Mare, maxDist = 0.7))
```

---

pooledSD *Extract Pooled Standard Deviation*

Description

The pooled estimated standard deviation is obtained by adding together the residual sum of squares for each non-null element of object, dividing by the sum of the corresponding residual degrees-of-freedom, and taking the square-root.

Usage

```
pooledSD(object)
```

Arguments

object	an object inheriting from class lmList.

Value

the pooled standard deviation for the non-null elements of object, with an attribute df with the number of degrees-of-freedom used in the estimation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lmList, lm

Examples

```
fm1 <- lmList(Orthodont)
pooledSD(fm1)
```

---

predict.gls *Predictions from a gls Object*

Description

The predictions for the linear model represented by object are obtained at the covariate values defined in newdata.

Usage

```
## S3 method for class 'gls'
predict(object, newdata, na.action, ...)
```

Arguments

object	an object inheriting from class "gls", representing a generalized least squares fitted linear model.
newdata	an optional data frame to be used for obtaining the predictions. All variables used in the linear model must be present in the data frame. If missing, the fitted values are returned.
na.action	a function that indicates what should happen when newdata contains NAs. The default action (na.fail) causes the function to print an error message and terminate if there are any incomplete observations.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the predicted values.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gls

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
newOvary <- data.frame(Time = c(-0.75, -0.5, 0, 0.5, 0.75))
predict(fm1, newOvary)
```

---

predict.gnls *Predictions from a gnls Object*

Description

The predictions for the nonlinear model represented by object are obtained at the covariate values defined in newdata.

Usage

```
## S3 method for class 'gnls'
predict(object, newdata, na.action, naPattern, ...)
```

Arguments

object	an object inheriting from class "gnls", representing a generalized nonlinear least squares fitted model.
newdata	an optional data frame to be used for obtaining the predictions. All variables used in the nonlinear model must be present in the data frame. If missing, the fitted values are returned.
na.action	a function that indicates what should happen when newdata contains NAs. The default action (na.fail) causes the function to print an error message and terminate if there are any incomplete observations.
naPattern	an expression or formula object, specifying which returned values are to be regarded as missing.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the predicted values.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gnls

Examples

```
fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            weights = varPower())
newSoybean <- data.frame(Time = c(10,30,50,80,100))
predict(fm1, newSoybean)
```

---

predict.lme *Predictions from an lme Object*

Description

The predictions at level i are obtained by adding together the population predictions (based only on the fixed effects estimates) and the estimated contributions of the random effects to the predictions at grouping levels less or equal to i. The resulting values estimate the best linear unbiased predictions (BLUPs) at level i. If group values not included in the original grouping factors are present in newdata, the corresponding predictions will be set to NA for levels greater or equal to the level at which the unknown groups occur.

Usage

```
## S3 method for class 'lme'
predict(object, newdata, level = Q, asList = FALSE,
        na.action = na.fail, ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model.
newdata	an optional data frame to be used for obtaining the predictions. All variables used in the fixed and random effects models, as well as the grouping factors, must be present in the data frame. If missing, the fitted values are returned.
level	an optional integer vector giving the level(s) of grouping to be used in obtaining the predictions. Level values increase from outermost to innermost grouping, with level zero corresponding to the population predictions. Defaults to the highest or innermost level of grouping.

asList	an optional logical value. If TRUE and a single value is given in level, the returned object is a list with the predictions split by groups; else the returned value is either a vector or a data frame, according to the length of level.
na.action	a function that indicates what should happen when newdata contains NAs. The default action (na.fail) causes the function to print an error message and terminate if there are any incomplete observations.
...	some methods for this generic require additional arguments. None are used in this method.

Value

if a single level of grouping is specified in level, the returned value is either a list with the predictions split by groups (asList = TRUE) or a vector with the predictions (asList = FALSE); else, when multiple grouping levels are specified in level, the returned object is a data frame with columns given by the predictions at different levels and the grouping factors.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lme, fitted.lme

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
newOrth <- data.frame(Sex = c("Male","Male","Female","Female","Male","Male"),
                      age = c(15, 20, 10, 12, 2, 4),
                      Subject = c("M01","M01","F30","F30","M04","M04"))
## The 'Orthodont' data has *no* 'F30', so predict  NA  at level 1 :
predict(fm1, newOrth, level = 0:1)
```

---

predict.lmList *Predictions from an lmList Object*

Description

If the grouping factor corresponding to object is included in newdata, the data frame is partitioned according to the grouping factor levels; else, newdata is repeated for all lm components. The predictions and, optionally, the standard errors for the predictions, are obtained for each lm component of object, using the corresponding element of the partitioned newdata, and arranged into a list with as many components as object, or combined into a single vector or data frame (if se.fit=TRUE).

Usage

```
## S3 method for class 'lmList'
predict(object, newdata, subset, pool, asList, se.fit, ...)
```

Arguments

object	an object inheriting from class "lmList", representing a list of lm objects with a common model.
newdata	an optional data frame to be used for obtaining the predictions. All variables used in the object model formula must be present in the data frame. If missing, the same data frame used to produce object is used.
subset	an optional character or integer vector naming the lm components of object from which the predictions are to be extracted. Default is NULL, in which case all components are used.
asList	an optional logical value. If TRUE, the returned object is a list with the predictions split by groups; else the returned value is a vector. Defaults to FALSE.
pool	an optional logical value indicating whether a pooled estimate of the residual standard error should be used. Default is attr(object, "pool").
se.fit	an optional logical value indicating whether pointwise standard errors should be computed along with the predictions. Default is FALSE.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a list with components given by the predictions (and, optionally, the standard errors for the predictions) from each lm component of object, a vector with the predictions from all lm components of object, or a data frame with columns given by the predictions and their corresponding standard errors.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lmList, predict.lm

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
predict(fm1, se.fit = TRUE)
```

predict.nlme *Predictions from an nlme Object*

Description

The predictions at level i are obtained by adding together the contributions from the estimated fixed effects and the estimated random effects at levels less or equal to i and evaluating the model function at the resulting estimated parameters. If group values not included in the original grouping factors are present in newdata, the corresponding predictions will be set to NA for levels greater or equal to the level at which the unknown groups occur.

Usage

```
## S3 method for class 'nlme'
predict(object, newdata, level = Q, asList = FALSE,
        na.action = na.fail, naPattern = NULL, ...)
```

Arguments

object	an object inheriting from class "nlme", representing a fitted nonlinear mixed-effects model.
newdata	an optional data frame to be used for obtaining the predictions. All variables used in the nonlinear model, the fixed and the random effects models, as well as the grouping factors, must be present in the data frame. If missing, the fitted values are returned.
level	an optional integer vector giving the level(s) of grouping to be used in obtaining the predictions. Level values increase from outermost to innermost grouping, with level zero corresponding to the population predictions. Defaults to the highest or innermost level of grouping (and is object$dims$Q).
asList	an optional logical value. If TRUE and a single value is given in level, the returned object is a list with the predictions split by groups; else the returned value is either a vector or a data frame, according to the length of level.
na.action	a function that indicates what should happen when newdata contains NAs. The default action (na.fail) causes the function to print an error message and terminate if there are any incomplete observations.
naPattern	an expression or formula object, specifying which returned values are to be regarded as missing.
...	some methods for this generic require additional arguments. None are used in this method.

Value

if a single level of grouping is specified in level, the returned value is either a list with the predictions split by groups (asList = TRUE) or a vector with the predictions (asList = FALSE); else, when multiple grouping levels are specified in level, the returned object is a data frame with columns given by the predictions at different levels and the grouping factors.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

nlme, fitted.lme

Examples

```
head(Loblolly) # groupedData  w/  'Seed' is grouping variable :
## Grouped Data: height ~ age | Seed
##    height age Seed
## 1    4.51   3  301
## 15  10.89   5  301
## ..  .....   .  ...

fm1 <- nlme(height ~ SSasymp(age, Asym, R0, lrc),  data = Loblolly,
            fixed = Asym + R0 + lrc ~ 1,
            random = Asym ~ 1, ## <---grouping--->  Asym ~ 1 | Seed
            start = c(Asym = 103, R0 = -8.5, lrc = -3.3))
fm1

age. <- seq(from = 2, to = 30, by = 2)
newLL.301 <- data.frame(age = age., Seed = 301)
newLL.329 <- data.frame(age = age., Seed = 329)
(p301 <- predict(fm1, newLL.301, level = 0:1))
(p329 <- predict(fm1, newLL.329, level = 0:1))
## Prediction are the same at level 0 :
all.equal(p301[,"predict.fixed"],
          p329[,"predict.fixed"])
## and differ by the 'Seed' effect at level 1 :
p301[,"predict.Seed"] -
p329[,"predict.Seed"]
```

---

print.summary.pdMat *Print a summary.pdMat Object*

Description

The standard deviations and correlations associated with the positive-definite matrix represented by `object` (considered as a variance-covariance matrix) are printed, together with the formula and the grouping level associated `object`, if any are present.

Usage

```
## S3 method for class 'summary.pdMat'
print(x, sigma, rdig, Level, resid, ...)
```

Arguments

x	an object inheriting from class "summary.pdMat", generally resulting from applying summary to an object inheriting from class "pdMat".
sigma	an optional numeric value used as a multiplier for the square-root factor of the positive-definite matrix represented by object (usually the estimated within-group standard deviation from a mixed-effects model). Defaults to 1.
rdig	an optional integer value with the number of significant digits to be used in printing correlations. Defaults to 3.
Level	an optional character string with a description of the grouping level associated with object (generally corresponding to levels of grouping in a mixed-effects model). Defaults to NULL.
resid	an optional logical value. If TRUE an extra row with the "residual" standard deviation given in sigma will be included in the output. Defaults to FALSE.
...	optional arguments passed to print.default; see the documentation on that method function.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

summary.pdMat,pdMat

Examples

```
pd1 <- pdCompSymm(3 * diag(2) + 1, form = ~age + age^2,
         data = Orthodont)
print(summary(pd1), sigma = 1.2, resid = TRUE)
```

---

print.varFunc *Print a varFunc Object*

Description

The class and the coefficients associated with x are printed.

Usage

```
## S3 method for class 'varFunc'
print(x, ...)
```

Arguments

x	an object inheriting from class "varFunc", representing a variance function structure.
...	optional arguments passed to print.default; see the documentation on that method function.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

summary.varFunc

Examples

```
vf1 <- varPower(0.3, form = ~age)
vf1 <- Initialize(vf1, Orthodont)
print(vf1)
```

---

qqnorm.gls *Normal Plot of Residuals from a gls Object*

Description

Diagnostic plots for assessing the normality of residuals the generalized least squares fit are obtained. The form argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a | operator) always implies that different panels are used for each level of the conditioning factor, according to a Trellis display.

Usage

```
## S3 method for class 'gls'
qqnorm(y, form, abline, id, idLabels, grid, ...)
```

Arguments

y	an object inheriting from class "gls", representing a generalized least squares fitted model.
form	an optional one-sided formula specifying the desired type of plot. Any variable present in the original data frame used to obtain y can be referenced. In addition, y itself can be referenced in the formula using the symbol ".". Conditional expressions on the right of a | operator can be used to define separate panels in a Trellis display. The expression on the right hand side of form and to the left of a | operator must evaluate to a residuals vector. Default is ~ resid(., type = "p"), corresponding to a normal plot of the standardized residuals.
abline	an optional numeric value, or numeric vector of length two. If given as a single value, a horizontal line will be added to the plot at that coordinate; else, if given as a vector, its values are used as the intercept and slope for a line added to the plot. If missing, no lines are added to the plot.

id	an optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for a two-sided outlier test for the standardized residuals (random effects). Observations with absolute standardized residuals (random effects) greater than the $1 - value/2$ quantile of the standard normal distribution are identified in the plot using idLabels. If given as a one-sided formula, its right hand side must evaluate to a logical, integer, or character vector which is used to identify observations in the plot. If missing, no observations are identified.
idLabels	an optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the observations identified according to id. If given as a one-sided formula, its right hand side must evaluate to a vector which is converted to character and used to label the identified observations. Default is the innermost grouping factor.
grid	an optional logical value indicating whether a grid should be added to plot. Default depends on the type of Trellis plot used: if xyplot defaults to TRUE, else defaults to FALSE.
...	optional arguments passed to the Trellis plot function.

Value

a diagnostic Trellis plot for assessing normality of residuals.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

gls, plot.gls

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
qqnorm(fm1, abline = c(0,1))
```

---

qqnorm.lme *Normal Plot of Residuals or Random Effects from an lme Object*

Description

Diagnostic plots for assessing the normality of residuals and random effects in the linear mixed-effects fit are obtained. The form argument gives considerable flexibility in the type of plot specification. A conditioning expression (on the right side of a | operator) always implies that different panels are used for each level of the conditioning factor, according to a Trellis display.

Usage

```
## S3 method for class 'lme'
qqnorm(y, form, abline, id, idLabels, grid, ...)
```

Arguments

y	an object inheriting from class "lme", representing a fitted linear mixed-effects model or from class "lmList", representing a list of lm objects, or from class "lm", representing a fitted linear model, or from class "nls", representing a nonlinear least squares fitted model.
form	an optional one-sided formula specifying the desired type of plot. Any variable present in the original data frame used to obtain y can be referenced. In addition, y itself can be referenced in the formula using the symbol ".". Conditional expressions on the right of a | operator can be used to define separate panels in a Trellis display. The expression on the right hand side of form and to the left of a | operator must evaluate to a residuals vector, or a random effects matrix. Default is ~ resid(., type = "p"), corresponding to a normal plot of the standardized residuals evaluated at the innermost level of nesting.
abline	an optional numeric value, or numeric vector of length two. If given as a single value, a horizontal line will be added to the plot at that coordinate; else, if given as a vector, its values are used as the intercept and slope for a line added to the plot. If missing, no lines are added to the plot.
id	an optional numeric value, or one-sided formula. If given as a value, it is used as a significance level for a two-sided outlier test for the standardized residuals (random effects). Observations with absolute standardized residuals (random effects) greater than the $1 - value/2$ quantile of the standard normal distribution are identified in the plot using idLabels. If given as a one-sided formula, its right hand side must evaluate to a logical, integer, or character vector which is used to identify observations in the plot. If missing, no observations are identified.
idLabels	an optional vector, or one-sided formula. If given as a vector, it is converted to character and used to label the observations identified according to id. If given as a one-sided formula, its right hand side must evaluate to a vector which is converted to character and used to label the identified observations. Default is the innermost grouping factor.
grid	an optional logical value indicating whether a grid should be added to plot. Default is FALSE.
...	optional arguments passed to the Trellis plot function.

Value

a diagnostic Trellis plot for assessing normality of residuals or random effects.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lme, plot.lme

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
## normal plot of standardized residuals by gender
qqnorm(fm1, ~ resid(., type = "p") | Sex, abline = c(0, 1))
## normal plots of random effects
qqnorm(fm1, ~ranef(.))
```

Quinidine *Quinidine Kinetics*

Description

The Quinidine data frame has 1471 rows and 14 columns.

Format

This data frame contains the following columns:

Subject a factor identifying the patient on whom the data were collected.

time a numeric vector giving the time (hr) at which the drug was administered or the blood sample drawn. This is measured from the time the patient entered the study.

conc a numeric vector giving the serum quinidine concentration (mg/L).

dose a numeric vector giving the dose of drug administered (mg). Although there were two different forms of quinidine administered, the doses were adjusted for differences in salt content by conversion to milligrams of quinidine base.

interval a numeric vector giving the when the drug has been given at regular intervals for a sufficiently long period of time to assume steady state behavior, the interval is recorded.

Age a numeric vector giving the age of the subject on entry to the study (yr).

Height a numeric vector giving the height of the subject on entry to the study (in.).

Weight a numeric vector giving the body weight of the subject (kg).

Race a factor with levels Caucasian, Latin, and Black identifying the race of the subject.

Smoke a factor with levels no and yes giving smoking status at the time of the measurement.

Ethanol a factor with levels none, current, former giving ethanol (alcohol) abuse status at the time of the measurement.

Heart a factor with levels No/Mild, Moderate, and Severe indicating congestive heart failure for the subject.

Creatinine an ordered factor with levels < 50 < >= 50 indicating the creatinine clearance (mg/min).

glyco a numeric vector giving the alpha-1 acid glycoprotein concentration (mg/dL). Often measured at the same time as the quinidine concentration.

Details

Verme et al. (1992) analyze routine clinical data on patients receiving the drug quinidine as a treatment for cardiac arrhythmia (atrial fibrillation or ventricular arrhythmias). All patients were receiving oral quinidine doses. At irregular intervals blood samples were drawn and serum concentrations of quinidine were determined. These data are analyzed in several publications, including Davidian and Giltinan (1995, section 9.3).

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.25)

Davidian, M. and Giltinan, D. M. (1995), *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, London.

Verme, C. N., Ludden, T. M., Clementi, W. A. and Harris, S. C. (1992), Pharmacokinetics of quinidine in male patients: A population analysis, *Clinical Pharmacokinetics*, **22**, 468-480.

---

quinModel *Model function for the Quinidine data*

Description

A model function for a model used with the `Quinidine` data. This function calls compiled C code.

Usage

```
quinModel(Subject, time, conc, dose, interval, lV, lKa, lCl)
```

Arguments

`Subject`	a factor identifying the patient on whom the data were collected.
`time`	a numeric vector giving the time (hr) at which the drug was administered or the blood sample drawn. This is measured from the time the patient entered the study.
`conc`	a numeric vector giving the serum quinidine concentration (mg/L).
`dose`	a numeric vector giving the dose of drug administered (mg). Although there were two different forms of quinidine administered, the doses were adjusted for differences in salt content by conversion to milligrams of quinidine base.
`interval`	a numeric vector giving the when the drug has been given at regular intervals for a sufficiently long period of time to assume steady state behavior, the interval is recorded.
`lV`	numeric. A vector of values of the natural log of the effective volume of distribution according to `Subject` and `time`.
`lKa`	numeric. A vector of values of the natural log of the absorption rate constant according to `Subject` and `time`.
`lCl`	numeric. A vector of values of the natural log of the clearance parameter according to `Subject` and `time`.

Details

See the details section of Quinidine for a description of the model function that quinModel evaluates.

Value

a numeric vector of predicted quinidine concentrations.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J. C. and Bates, D. M. (2000) *Mixed-effects Models in S and S-PLUS*, Springer. (section 8.2)

Rail *Evaluation of Stress in Railway Rails*

Description

The Rail data frame has 18 rows and 2 columns.

Format

This data frame contains the following columns:

Rail an ordered factor identifying the rail on which the measurement was made.

travel a numeric vector giving the travel time for ultrasonic head-waves in the rail (nanoseconds). The value given is the original travel time minus 36,100 nanoseconds.

Details

Devore (2000, Example 10.10, p. 427) cites data from an article in *Materials Evaluation* on "a study of travel time for a certain type of wave that results from longitudinal stress of rails used for railroad track."

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.26)

Devore, J. L. (2000), *Probability and Statistics for Engineering and the Sciences (5th ed)*, Duxbury, Boston, MA.

random.effects *Extract Random Effects*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include `lmList` and `lme`.

`random.effects` is an alias for `ranef`; methods are implemented for the latter.

Usage

```
random.effects(object, ...)
ranef(object, ...)
```

Arguments

`object`	any fitted model object from which random effects estimates can be extracted.
`...`	some methods for this generic function require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 100, 461.

See Also

`ranef.lmList`, `ranef.lme`

ranef.lme *Extract lme Random Effects*

Description

The estimated random effects at level i are represented as a data frame with rows given by the different groups at that level and columns given by the random effects. If a single level of grouping is specified, the returned object is a data frame; else, the returned object is a list of such data frames. Optionally, the returned data frame(s) may be augmented with covariates summarized over groups.

Usage

```
## S3 method for class 'lme'
ranef(object, augFrame, level, data, which, FUN,
      standard, omitGroupingFactor, subset, ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model.
augFrame	an optional logical value. If TRUE, the returned data frame is augmented with variables defined in data; else, if FALSE, only the coefficients are returned. Defaults to FALSE.
level	an optional vector of positive integers giving the levels of grouping to be used in extracting the random effects from an object with multiple nested grouping levels. Defaults to all levels of grouping.
data	an optional data frame with the variables to be used for augmenting the returned data frame when augFrame = TRUE. Defaults to the data frame used to fit object.
which	an optional positive integer vector specifying which columns of data should be used in the augmentation of the returned data frame. Defaults to all columns in data.
FUN	an optional summary function or a list of summary functions to be applied to group-varying variables, when collapsing data by groups. Group-invariant variables are always summarized by the unique value that they assume within that group. If FUN is a single function it will be applied to each non-invariant variable by group to produce the summary for that variable. If FUN is a list of functions, the names in the list should designate classes of variables in the frame such as ordered, factor, or numeric. The indicated function will be applied to any group-varying variables of that class. The default functions to be used are mean for numeric factors, and Mode for both factor and ordered. The Mode function, defined internally in gsummary, returns the modal or most popular value of the variable. It is different from the mode function that returns the S-language mode of the variable.
standard	an optional logical value indicating whether the estimated random effects should be "standardized" (i.e. divided by the estimate of the standard deviation of that group of random effects). Defaults to FALSE.
omitGroupingFactor	an optional logical value. When TRUE the grouping factor itself will be omitted from the group-wise summary of data but the levels of the grouping factor will continue to be used as the row names for the returned data frame. Defaults to FALSE.
subset	an optional expression indicating for which rows the random effects should be extracted.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame, or list of data frames, with the estimated random effects at the grouping level(s) specified in level and, optionally, other covariates summarized over groups. The returned object inherits from classes random.effects.lme and data.frame.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 100, 461.

See Also

coef.lme, gsummary, lme, plot.ranef.lme, random.effects

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
ranef(fm1)
random.effects(fm1)             # same as above
random.effects(fm1, augFrame = TRUE)
```

---

ranef.lmList *Extract lmList Random Effects*

Description

The difference between the individual lm components coefficients and their average is calculated.

Usage

```
## S3 method for class 'lmList'
ranef(object, augFrame, data, which, FUN, standard,
                omitGroupingFactor, ...)
```

Arguments

object	an object inheriting from class "lmList", representing a list of lm objects with a common model.
augFrame	an optional logical value. If TRUE, the returned data frame is augmented with variables defined in data; else, if FALSE, only the coefficients are returned. Defaults to FALSE.
data	an optional data frame with the variables to be used for augmenting the returned data frame when augFrame = TRUE. Defaults to the data frame used to fit object.
which	an optional positive integer vector specifying which columns of data should be used in the augmentation of the returned data frame. Defaults to all columns in data.

FUN — an optional summary function or a list of summary functions to be applied to group-varying variables, when collapsing data by groups. Group-invariant variables are always summarized by the unique value that they assume within that group. If FUN is a single function it will be applied to each non-invariant variable by group to produce the summary for that variable. If FUN is a list of functions, the names in the list should designate classes of variables in the frame such as ordered, factor, or numeric. The indicated function will be applied to any group-varying variables of that class. The default functions to be used are mean for numeric factors, and Mode for both factor and ordered. The Mode function, defined internally in gsummary, returns the modal or most popular value of the variable. It is different from the mode function that returns the S-language mode of the variable.

standard — an optional logical value indicating whether the estimated random effects should be "standardized" (i.e. divided by the corresponding estimated standard error). Defaults to FALSE.

omitGroupingFactor — an optional logical value. When TRUE the grouping factor itself will be omitted from the group-wise summary of data but the levels of the grouping factor will continue to be used as the row names for the returned data frame. Defaults to FALSE.

... — some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the differences between the individual lm coefficients in object and their average.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer, esp. pp. 100, 461.

See Also

fixed.effects.lmList, lmList, random.effects

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
ranef(fm1)
random.effects(fm1)             # same as above
```

RatPupWeight *The weight of rat pups*

Description

The RatPupWeight data frame has 322 rows and 5 columns.

Format

This data frame contains the following columns:

weight a numeric vector

sex a factor with levels Male Female

Litter an ordered factor with levels 9 < 8 < 7 < 4 < 2 < 10 < 1 < 3 < 5 < 6 < 21 < 22 < 24 < 27 < 26 < 25 < 23 < 17 < 11 < 14 < 13 < 15 < 16 < 20 < 19 < 18 < 12

Lsize a numeric vector

Treatment an ordered factor with levels Control < Low < High

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

recalc *Recalculate Condensed Linear Model Object*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include: corStruct, modelStruct, reStruct, and varFunc.

Usage

```
recalc(object, conLin, ...)
```

Arguments

object	any object which induces a recalculation of the condensed linear model object conLin.
conLin	a condensed linear model object, consisting of a list with components "Xy", corresponding to a regression matrix (X) combined with a response vector (y), and "logLik", corresponding to the log-likelihood of the underlying model.
...	some methods for this generic can take additional arguments.

Value

the recalculated condensed linear model object.

Note

This function is only used inside model fitting functions, such as `lme` and `gls`, that require recalculation of a condensed linear model object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`recalc.corStruct`, `recalc.modelStruct`, `recalc.reStruct`, `recalc.varFunc`

---

recalc.corStruct *Recalculate for corStruct Object*

Description

This method function pre-multiples the `"Xy"` component of `conLin` by the transpose square-root factor(s) of the correlation matrix (matrices) associated with `object` and adds the log-likelihood contribution of `object`, given by `logLik(object)`, to the `"logLik"` component of `conLin`.

Usage

```
## S3 method for class 'corStruct'
recalc(object, conLin, ...)
```

Arguments

`object`	an object inheriting from class `"corStruct"`, representing a correlation structure.
`conLin`	a condensed linear model object, consisting of a list with components `"Xy"`, corresponding to a regression matrix (`X`) combined with a response vector (`y`), and `"logLik"`, corresponding to the log-likelihood of the underlying model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the recalculated condensed linear model object.

Note

This method function is only used inside model fitting functions, such as `lme` and `gls`, that allow correlated error terms.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

corFactor, logLik.corStruct

---

recalc.modelStruct *Recalculate for a modelStruct Object*

Description

This method function recalculates the condensed linear model object using each element of `object` sequentially from last to first.

Usage

```
## S3 method for class 'modelStruct'
recalc(object, conLin, ...)
```

Arguments

`object`	an object inheriting from class `"modelStruct"`, representing a list of model components, such as `corStruct` and `varFunc` objects.
`conLin`	an optional condensed linear model object, consisting of a list with components `"Xy"`, corresponding to a regression matrix (`X`) combined with a response vector (`y`), and `"logLik"`, corresponding to the log-likelihood of the underlying model. Defaults to `attr(object, "conLin")`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the recalculated condensed linear model object.

Note

This method function is generally only used inside model fitting functions, such as `lme` and `gls`, that allow model components, such as correlated error terms and variance functions.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

recalc.corStruct, recalc.reStruct, recalc.varFunc

recalc.reStruct *Recalculate for an reStruct Object*

Description

The log-likelihood, or restricted log-likelihood, of the Gaussian linear mixed-effects model represented by `object` and `conLin` (assuming spherical within-group covariance structure), evaluated at `coef(object)` is calculated and added to the `logLik` component of `conLin`. The `settings` attribute of `object` determines whether the log-likelihood, or the restricted log-likelihood, is to be calculated. The computational methods for the (restricted) log-likelihood calculations are described in Bates and Pinheiro (1998).

Usage

```
## S3 method for class 'reStruct'
recalc(object, conLin, ...)
```

Arguments

`object`	an object inheriting from class `"reStruct"`, representing a random effects structure and consisting of a list of `pdMat` objects.
`conLin`	a condensed linear model object, consisting of a list with components `"Xy"`, corresponding to a regression matrix (`X`) combined with a response vector (`y`), and `"logLik"`, corresponding to the log-likelihood of the underlying model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the condensed linear model with its `logLik` component updated.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`logLik`, `lme`, `recalc`, `reStruct`

recalc.varFunc *Recalculate for varFunc Object*

Description

This method function pre-multiples the "Xy" component of conLin by a diagonal matrix with diagonal elements given by the weights corresponding to the variance structure represented by objecte and adds the log-likelihood contribution of object, given by logLik(object), to the "logLik" component of conLin.

Usage

```
## S3 method for class 'varFunc'
recalc(object, conLin, ...)
```

Arguments

object	an object inheriting from class "varFunc", representing a variance function structure.
conLin	a condensed linear model object, consisting of a list with components "Xy", corresponding to a regression matrix (X) combined with a response vector (y), and "logLik", corresponding to the log-likelihood of the underlying model.
...	some methods for this generic require additional arguments. None are used in this method.

Value

the recalculated condensed linear model object.

Note

This method function is only used inside model fitting functions, such as lme and gls, that allow heteroscedastic error terms.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

recalc, varWeights, logLik.varFunc

Relaxin *Assay for Relaxin*

Description

The `Relaxin` data frame has 198 rows and 3 columns.

Format

This data frame contains the following columns:

Run an ordered factor with levels `5` < `8` < `9` < `3` < `4` < `2` < `7` < `1` < `6`

conc a numeric vector

cAMP a numeric vector

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

Remifentanil *Pharmacokinetics of Remifentanil*

Description

Intravenous infusion of remifentanil (a strong analgesic) in different rates over varying time periods was applied to a total of 65 patients. Concentration measurements of remifentanil were taken along with several covariates resulting in the `Remifentanil` data frame with 2107 rows and 12 columns.

Usage

```
Remifentanil
```

Format

This data frame (of class `"groupedData"`, specifically `"nfnGroupedData"`) contains the following columns:

`ID`: numerical (patient) IDs.

`Subject`: an `ordered` factor with 65 levels (of the `ID`s): `30` < `21` < `25` < `23` < `29` < ..... < `36` < `6` < `5` < `10` < `9`.

`Time`: time from beginning of infusion in minutes (`numeric`).

`conc`: remifentanil concentration in [ng / ml] (numeric).

`Rate`: infusion rate in [µg / min].

`Amt`: amount of remifentanil given in the current time interval in [µg].

`Age`: age of the patient in years.

`Sex`: gender of the patient, a `factor` with levels `Female` and `Male`.

`Ht`: height of the patient in cm.

`Wt`: weight of the patient in kg.

`BSA`: body surface area (DuBois and DuBois 1916): $BSA := Wt^{0.425} \cdot Ht^{0.725} \cdot 0.007184$.

`LBM`: lean body mass (James 1976), with slightly different formula for men $LBM_m := 1.1Wt - 128(Wt/Ht)^2$, and women $LBM_f := 1.07Wt - 148(Wt/Ht)^2$.

Author(s)

of this help page: Niels Hagenbuch and Martin Maechler, SfS ETH Zurich.

Source

Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

References

Minto CF, Schnider TW, Egan TD, Youngs E, Lemmens HJM, Gambus PL, Billard V, Hoke JF, Moore KHP, Hermann DJ, Muir KT, Mandema JW, Shafer SL (1997). Influence of age and gender on the pharmacokinetics and pharmacodynamics of remifentanil: I. Model development. *Anesthesiology* **86** 1, 10–23. doi:10.1097/00000542199701000000004

Charles F. Minto, Thomas W. Schnider and Steven L. Shafer (1997). Pharmacokinetics and Pharmacodynamics of Remifentanil: II. Model Application. *Anesthesiology* **86** 1, 24–33. doi:10.1097/00000542199701000000005

Examples

```
plot(Remifentanil, type = "l", lwd = 2) # shows the 65 patients' remi profiles

## The same on  log-log  scale  (*more* sensible for modeling ?):
plot(Remifentanil, type = "l", lwd = 2, scales = list(log=TRUE))

str(Remifentanil)
summary(Remifentanil)

plot(xtabs(~Subject, Remifentanil))
summary(unclass(table(Remifentanil$Subject)))
## between 20 and 54 measurements per patient (median: 24; mean: 32.42)

## Only first measurement of each patient :
dim(Remi.1 <- Remifentanil[!duplicated(Remifentanil[,"ID"]),]) #  65 x 12

LBMfn <- function(Wt, Ht, Sex) ifelse(Sex == "Female",
                                      1.07 * Wt - 148*(Wt/Ht)^2,
                                      1.1  * Wt - 128*(Wt/Ht)^2)
with(Remi.1,
```

```
     stopifnot(all.equal(BSA, Wt^{0.425} * Ht^{0.725} * 0.007184,
                         tolerance = 1.5e-5),
               all.equal(LBM, LBMfn(Wt, Ht, Sex),
                         tolerance = 7e-7)
))

## Rate: typically  3 µg / kg body weight, but :
sunflowerplot(Rate ~ Wt, Remifentanil)
abline(0,3, lty=2, col=adjustcolor("black", 0.5))
```

`residuals.gls` *Extract gls Residuals*

Description

The residuals for the linear model represented by `object` are extracted.

Usage

```
## S3 method for class 'gls'
residuals(object, type, ...)
```

Arguments

`object`	an object inheriting from class `"gls"`, representing a generalized least squares fitted linear model, or from class `gnls`, representing a generalized nonlinear least squares fitted linear model.
`type`	an optional character string specifying the type of residuals to be used. If `"response"`, the "raw" residuals (observed - fitted) are used; else, if `"pearson"`, the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if `"normalized"`, the normalized residuals (standardized residuals pre-multiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to `"response"`.
`...`	some methods for this generic function require additional arguments. None are used in this method.

Value

a vector with the residuals for the linear model represented by `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`gls`

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
residuals(fm1)
```

---

residuals.glsStruct *Calculate glsStruct Residuals*

Description

The residuals for the linear model represented by `object` are extracted.

Usage

```
## S3 method for class 'glsStruct'
residuals(object, glsFit, ...)
```

Arguments

`object`	an object inheriting from class `"glsStruct"`, representing a list of linear model components, such as `corStruct` and `"varFunc"` objects.
`glsFit`	an optional list with components `logLik` (log-likelihood), `beta` (coefficients), `sigma` (standard deviation for error term), `varBeta` (coefficients' covariance matrix), `fitted` (fitted values), and `residuals` (residuals). Defaults to `attr(object, "glsFit")`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the residuals for the linear model represented by `object`.

Note

This method function is primarily used inside `gls` and `residuals.gls`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`gls`, `glsStruct`, `residuals.gls`, `fitted.glsStruct`

`residuals.gnlsStruct` *Calculate gnlsStruct Residuals*

Description

The residuals for the nonlinear model represented by `object` are extracted.

Usage

```
## S3 method for class 'gnlsStruct'
residuals(object, ...)
```

Arguments

`object`	an object inheriting from class "`gnlsStruct`", representing a list of model components, such as `corStruct` and `varFunc` objects, and attributes specifying the underlying nonlinear model and the response variable.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a vector with the residuals for the nonlinear model represented by `object`.

Note

This method function is primarily used inside `gnls` and `residuals.gnls`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`gnls`, `residuals.gnls`, `fitted.gnlsStruct`

`residuals.lme` *Extract lme Residuals*

Description

The residuals at level i are obtained by subtracting the fitted levels at that level from the response vector (and dividing by the estimated within-group standard error, if `type="pearson"`). The fitted values at level i are obtained by adding together the population fitted values (based only on the fixed effects estimates) and the estimated contributions of the random effects to the fitted values at grouping levels less or equal to i.

Usage

```
## S3 method for class 'lme'
residuals(object, level = Q,
          type = c("response", "pearson", "normalized"), asList = FALSE, ...)
```

Arguments

`object`	an object inheriting from class `"lme"`, representing a fitted linear mixed-effects model.
`level`	an optional integer vector giving the level(s) of grouping to be used in extracting the residuals from `object`. Level values increase from outermost to innermost grouping, with level zero corresponding to the population residuals. Defaults to the highest or innermost level of grouping.
`type`	an optional character string specifying the type of residuals to be used. If `"response"`, as by default, the "raw" residuals (observed - fitted) are used; else, if `"pearson"`, the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if `"normalized"`, the normalized residuals (standardized residuals pre-multiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided.
`asList`	an optional logical value. If `TRUE` and a single value is given in `level`, the returned object is a list with the residuals split by groups; else the returned value is either a vector or a data frame, according to the length of `level`. Defaults to `FALSE`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

if a single level of grouping is specified in `level`, the returned value is either a list with the residuals split by groups (`asList = TRUE`) or a vector with the residuals (`asList = FALSE`); else, when multiple grouping levels are specified in `level`, the returned object is a data frame with columns given by the residuals at different levels and the grouping factors. For a vector or data frame result the `naresid` method is applied.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

lme, fitted.lme

Examples

```
fm1 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
head(residuals(fm1, level = 0:1))
summary(residuals(fm1) /
        residuals(fm1, type = "p")) # constant scaling factor 1.432
```

---

residuals.lmeStruct *Calculate lmeStruct Residuals*

Description

The residuals at level i are obtained by subtracting the fitted values at that level from the response vector. The fitted values at level i are obtained by adding together the population fitted values (based only on the fixed effects estimates) and the estimated contributions of the random effects to the fitted values at grouping levels less or equal to i.

Usage

```
## S3 method for class 'lmeStruct'
residuals(object, level, conLin, lmeFit, ...)
```

Arguments

object	an object inheriting from class "lmeStruct", representing a list of linear mixed-effects model components, such as reStruct, corStruct, and varFunc objects.
level	an optional integer vector giving the level(s) of grouping to be used in extracting the residuals from object. Level values increase from outermost to innermost grouping, with level zero corresponding to the population fitted values. Defaults to the highest or innermost level of grouping.
conLin	an optional condensed linear model object, consisting of a list with components "Xy", corresponding to a regression matrix (X) combined with a response vector (y), and "logLik", corresponding to the log-likelihood of the underlying lme model. Defaults to attr(object, "conLin").
lmeFit	an optional list with components beta and b containing respectively the fixed effects estimates and the random effects estimates to be used to calculate the residuals. Defaults to attr(object, "lmeFit").
...	some methods for this generic accept optional arguments.

Value

if a single level of grouping is specified in `level`, the returned value is a vector with the residuals at the desired level; else, when multiple grouping levels are specified in `level`, the returned object is a matrix with columns given by the residuals at different levels.

Note

This method function is primarily used within the `lme` function.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lme`, `residuals.lme`, `fitted.lmeStruct`

---

residuals.lmList *Extract lmList Residuals*

Description

The residuals are extracted from each `lm` component of `object` and arranged into a list with as many components as `object`, or combined into a single vector.

Usage

```
## S3 method for class 'lmList'
residuals(object, type, subset, asList, ...)
```

Arguments

`object`	an object inheriting from class `"lmList"`, representing a list of `lm` objects with a common model.
`subset`	an optional character or integer vector naming the `lm` components of `object` from which the residuals are to be extracted. Default is `NULL`, in which case all components are used.
`type`	an optional character string specifying the type of residuals to be extracted. Options include `"response"` for the "raw" residuals (observed - fitted), `"pearson"` for the standardized residuals (raw residuals divided by the estimated residual standard error) using different standard errors for each `lm` fit, and `"pooled.pearson"` for the standardized residuals using a pooled estimate of the residual standard error. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to `"response"`.
`asList`	an optional logical value. If `TRUE`, the returned object is a list with the residuals split by groups; else the returned value is a vector. Defaults to `FALSE`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a list with components given by the residuals of each `lm` component of `object`, or a vector with the residuals for all `lm` components of `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lmList`, `fitted.lmList`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
residuals(fm1)
```

---

residuals.nlmeStruct *Calculate nlmeStruct Residuals*

Description

The residuals at level i are obtained by subtracting the fitted values at that level from the response vector. The fitted values at level i are obtained by adding together the contributions from the estimated fixed effects and the estimated random effects at levels less or equal to i and evaluating the model function at the resulting estimated parameters.

Usage

```
## S3 method for class 'nlmeStruct'
residuals(object, level, conLin, ...)
```

Arguments

`object`	an object inheriting from class `"nlmeStruct"`, representing a list of mixed-effects model components, such as `reStruct`, `corStruct`, and `varFunc` objects.
`level`	an optional integer vector giving the level(s) of grouping to be used in extracting the residuals from `object`. Level values increase from outermost to innermost grouping, with level zero corresponding to the population fitted values. Defaults to the highest or innermost level of grouping.
`conLin`	an optional condensed linear model object, consisting of a list with components `"Xy"`, corresponding to a regression matrix (`X`) combined with a response vector (`y`), and `"logLik"`, corresponding to the log-likelihood of the underlying nlme model. Defaults to `attr(object, "conLin")`.
`...`	optional arguments to the residuals generic. Not used.

Value

if a single level of grouping is specified in `level`, the returned value is a vector with the residuals at the desired level; else, when multiple grouping levels are specified in `level`, the returned object is a matrix with columns given by the residuals at different levels.

Note

This method function is primarily used within the `nlme` function.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Bates, D.M. and Pinheiro, J.C. (1998) "Computational methods for multilevel models" available in PostScript or PDF formats at http://nlme.stat.wisc.edu

See Also

`nlme`, `fitted.nlmeStruct`

---

reStruct *Random Effects Structure*

Description

This function is a constructor for the `reStruct` class, representing a random effects structure and consisting of a list of `pdMat` objects, plus a `settings` attribute containing information for the optimization algorithm used to fit the associated mixed-effects model.

Usage

```
reStruct(object, pdClass, REML, data)
## S3 method for class 'reStruct'
print(x, sigma, reEstimates, verbose, ...)
```

Arguments

`object`	any of the following: (i) a one-sided formula of the form `~x1+...+xn	g1/.../gm`, with `x1+...+xn` specifying the model for the random effects and `g1/.../gm` the grouping structure (`m` may be equal to 1, in which case no `/` is required). The random effects formula will be repeated for all levels of grouping, in the case of multiple levels of grouping; (ii) a list of one-sided formulas of the form `~x1+...+xn	g`, with possibly different random effects models for each grouping level. The order of nesting will be assumed the same as the order of the elements in the list; (iii) a one-sided formula of the form `~x1+...+xn`, or a

	pdMat object with a formula (i.e. a non-NULL value for formula(object)), or a list of such formulas or pdMat objects. In this case, the grouping structure formula will be derived from the data used to to fit the mixed-effects model, which should inherit from class groupedData; (iv) a named list of formulas or pdMat objects as in (iii), with the grouping factors as names. The order of nesting will be assumed the same as the order of the order of the elements in the list; (v) an reStruct object.
pdClass	an optional character string with the name of the pdMat class to be used for the formulas in object. Defaults to "pdLogChol" which corresponds to a general positive-definite matrix (Log-Cholesky parametrization).
REML	an optional logical value. If TRUE, the associated mixed-effects model will be fitted using restricted maximum likelihood; else, if FALSE, maximum likelihood will be used. Defaults to FALSE.
data	an optional data frame in which to evaluate the variables used in the random effects formulas in object. It is used to obtain the levels for factors, which affect the dimensions and the row/column names of the underlying pdMat objects. If NULL, no attempt is made to obtain information on factors appearing in the formulas. Defaults to the parent frame from which the function was called.
x	an object inheriting from class reStruct to be printed.
sigma	an optional numeric value used as a multiplier for the square-root factors of the pdMat components (usually the estimated within-group standard deviation from a mixed-effects model). Defaults to 1.
reEstimates	an optional list with the random effects estimates for each level of grouping. Only used when verbose = TRUE.
verbose	an optional logical value determining if the random effects estimates should be printed. Defaults to FALSE.
...	Optional arguments can be given to other methods for this generic. None are used in this method.

Value

an object inheriting from class reStruct, representing a random effects structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

groupedData, lme, pdMat, solve.reStruct, summary.reStruct, update.reStruct

Examples

```
rs1 <- reStruct(list(Dog = ~day, Side = ~1), data = Pixel)
rs1 # 2 entries "Uninitialized"
str(rs1) # a bit more
```

simulate.lme *Simulate Results from* lme *Models*

Description

The model object is fit to the data. Using the fitted values of the parameters, nsim new data vectors from this model are simulated. Both object and m2 are fit by maximum likelihood (ML) and/or by restricted maximum likelihood (REML) to each of the simulated data vectors.

Usage

```
## S3 method for class 'lme'
simulate(object, nsim = 1, seed = , m2,
         method = c("REML", "ML"), niterEM = c(40, 200), useGen, ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model, or a list containing an lme model specification. If given as a list, it should contain components fixed, data, and random with values suitable for a call to lme. This argument defines the null model.
m2	an "lme" object or a list, like object containing a second lme model specification. This argument defines the alternative model. If given as a list, only those parts of the specification that change between model object and m2 need to be specified.
seed	an optional integer that is passed to set.seed. Defaults to a random integer.
method	an optional character array. If it includes "REML" the models are fit by maximizing the restricted log-likelihood. If it includes "ML" the log-likelihood is maximized. Defaults to c("REML", "ML"), in which case both methods are used.
nsim	an optional positive integer specifying the number of simulations to perform. Defaults to 1. **This has changed. Previously the default was 1000.**
niterEM	an optional integer vector of length 2 giving the number of iterations of the EM algorithm to apply when fitting the object and m2 to each simulated set of data. Defaults to c(40,200).
useGen	an optional logical value. If TRUE, the nlminb optimizer is used with numerical derivatives of the log-likelihood. If FALSE, the nlm algorithm is used with an analytic gradient. The default depends on the "pdMat" classes used in object and m2: if both are standard classes (see pdClasses) then defaults to FALSE, otherwise defaults to TRUE.
...	optional additional arguments. None are used.

Value

an object of class simulate.lme with components null and alt. Each of these has components ML and/or REML which are matrices. An attribute called seed contains the seed that was used for the random number generator.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) *Mixed-Effects Models in S and S-PLUS*, Springer.

See Also

lme, set.seed

Examples

```
orthSim <-
   simulate.lme(list(fixed = distance ~ age, data = Orthodont,
                     random = ~ 1 | Subject),
                nsim = 3, # limited here for speed
                m2 = list(random = ~ age | Subject))
```

---

solve.pdMat *Calculate Inverse of a Positive-Definite Matrix*

Description

The positive-definite matrix represented by a is inverted and assigned to a.

Usage

```
## S3 method for class 'pdMat'
solve(a, b, ...)
```

Arguments

a	an object inheriting from class "pdMat", representing a positive definite matrix.
b	this argument is only included for consistency with the generic function and is not used in this method function.
...	some methods for this generic require additional arguments. None are used in this method.

Value

a pdMat object similar to a, but with coefficients corresponding to the inverse of the positive-definite matrix represented by a.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

pdMat

Examples

```
pd1 <- pdCompSymm(3 * diag(3) + 1)
solve(pd1)
```

`solve.reStruct` *Apply Solve to an reStruct Object*

Description

`Solve` is applied to each `pdMat` component of `a`, which results in inverting the positive-definite matrices they represent.

Usage

```
## S3 method for class 'reStruct'
solve(a, b, ...)
```

Arguments

`a`	an object inheriting from class `"reStruct"`, representing a random effects structure and consisting of a list of `pdMat` objects.
`b`	this argument is only included for consistency with the generic function and is not used in this method function.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

an `reStruct` object similar to `a`, but with the `pdMat` components representing the inverses of the matrices represented by the components of `a`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

solve.pdMat, reStruct

Examples

```
rs1 <- reStruct(list(A = pdSymm(diag(1:3), form = ~Score),
  B = pdDiag(2 * diag(4), form = ~Educ)))
solve(rs1)
```

Soybean *Growth of soybean plants*

Description

The Soybean data frame has 412 rows and 5 columns.

Format

This data frame contains the following columns:

Plot a factor giving a unique identifier for each plot.

Variety a factor indicating the variety; Forrest (F) or Plant Introduction #416937 (P).

Year a factor indicating the year the plot was planted.

Time a numeric vector giving the time the sample was taken (days after planting).

weight a numeric vector giving the average leaf weight per plant (g).

Details

These data are described in Davidian and Giltinan (1995, 1.1.3, p.7) as "Data from an experiment to compare growth patterns of two genotypes of soybeans: Plant Introduction #416937 (P), an experimental strain, and Forrest (F), a commercial variety."

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.27)

Davidian, M. and Giltinan, D. M. (1995), *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, London.

Examples

```
summary(fm1 <- nlsList(SSlogis, data = Soybean))
```

splitFormula *Split a Formula*

Description

Splits the right hand side of form into a list of subformulas according to the presence of sep. The left hand side of form, if present, will be ignored. The length of the returned list will be equal to the number of occurrences of sep in form plus one.

Usage

```
splitFormula(form, sep)
```

Arguments

form	a formula object.
sep	an optional character string specifying the separator to be used for splitting the formula. Defaults to "/".

Value

a list of formulas, corresponding to the split of form according to sep.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

formula

Examples

```
splitFormula(~ g1/g2/g3)
```

Spruce *Growth of Spruce Trees*

Description

The Spruce data frame has 1027 rows and 4 columns.

Format

This data frame contains the following columns:

Tree a factor giving a unique identifier for each tree.

days a numeric vector giving the number of days since the beginning of the experiment.

logSize a numeric vector giving the logarithm of an estimate of the volume of the tree trunk.

plot a factor identifying the plot in which the tree was grown.

Details

Diggle, Liang, and Zeger (1994, Example 1.3, page 5) describe data on the growth of spruce trees that have been exposed to an ozone-rich atmosphere or to a normal atmosphere.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.28)

Diggle, Peter J., Liang, Kung-Yee and Zeger, Scott L. (1994), *Analysis of longitudinal data*, Oxford University Press, Oxford.

`summary.corStruct` *Summarize a corStruct Object*

Description

This method function prepares `object` to be printed using the `print.summary` method, by changing its class and adding a `structName` attribute to it.

Usage

```
## S3 method for class 'corStruct'
summary(object, structName, ...)
```

Arguments

`object`	an object inheriting from class `"corStruct"`, representing a correlation structure.
`structName`	an optional character string defining the type of correlation structure associated with `object`, to be used in the `print.summary` method. Defaults to `class(object)[1]`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

an object identical to `object`, but with its class changed to `summary.corStruct` and an additional attribute `structName`. The returned value inherits from the same classes as `object`.

Author(s)

José Pinheiro and Douglas Bates

See Also

`corClasses`, `corNatural`, `Initialize.corStruct`, `summary`

Examples

```
cs1 <- corAR1(0.2)
summary(cs1)
```

summary.gls *Summarize a Generalized Least Squares* gls *Object*

Description

Additional information about the linear model fit represented by object is extracted and included as components of object.

Usage

```
## S3 method for class 'gls'
summary(object, verbose, ...)
```

Arguments

object	an object inheriting from class "gls", representing a generalized least squares fitted linear model.
verbose	an optional logical value used to control the amount of output when the object is printed. Defaults to FALSE.
...	some methods for this generic require additional arguments. None are used in this method.

Value

an object inheriting from class summary.gls with all components included in object (see glsObject for a full description of the components) plus the following components:

corBeta	approximate correlation matrix for the coefficients estimates
tTable	a matrix with columns Value, Std. Error, t-value, and p-value representing respectively the coefficients estimates, their approximate standard errors, the ratios between the estimates and their standard errors, and the associated p-value under a t approximation. Rows correspond to the different coefficients.
residuals	if more than five observations are used in the gls fit, a vector with the minimum, first quartile, median, third quartile, and maximum of the residuals distribution; else the residuals.
AIC	the Akaike Information Criterion corresponding to object.
BIC	the Bayesian Information Criterion corresponding to object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

AIC, BIC, gls, summary

Examples

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
summary(fm1)
coef(summary(fm1)) # "the matrix"
```

`summary.lme` *Summarize an lme Object*

Description

Additional information about the linear mixed-effects fit represented by `object` is extracted and included as components of `object`. The returned object has a `print` and a `coef` method, the latter returning the coefficient's `tTable`.

Usage

```
## S3 method for class 'lme'
summary(object, adjustSigma, verbose, ...)
## S3 method for class 'summary.lme'
print(x, verbose = FALSE, ...)
```

Arguments

`object`	an object inheriting from class `"lme"`, representing a fitted linear mixed-effects model.
`adjustSigma`	an optional logical value. If `TRUE` and the estimation method used to obtain `object` was maximum likelihood, the residual standard error is multiplied by $\sqrt{n_{obs}/(n_{obs} - n_{par})}$, converting it to a REML-like estimate. This argument is only used when a single fitted object is passed to the function. Default is `TRUE`.
`verbose`	an optional logical value used to control the amount of output in the `print.summary.lme` method. Defaults to `FALSE`.
`...`	additional optional arguments passed to methods, mainly for the `print` method.
`x`	a `"summary.lme"` object.

Value

an object inheriting from class `summary.lme` with all components included in `object` (see `lmeObject` for a full description of the components) plus the following components:

`corFixed`	approximate correlation matrix for the fixed effects estimates.
`tTable`	a matrix with columns named `Value`, `Std. Error`, `DF`, `t-value`, and `p-value` representing respectively the fixed effects estimates, their approximate standard errors, the denominator degrees of freedom, the ratios between the estimates and their standard errors, and the associated p-value from a t distribution. Rows correspond to the different fixed effects.

residuals	if more than five observations are used in the lme fit, a vector with the minimum, first quartile, median, third quartile, and maximum of the innermost grouping level residuals distribution; else the innermost grouping level residuals.
AIC	the Akaike Information Criterion corresponding to object.
BIC	the Bayesian Information Criterion corresponding to object.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

AIC, BIC, lme.

Examples

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
(s1 <- summary(fm1))
coef(s1) # the (coef | Std.E | t | P-v ) matrix
```

---

summary.lmList *Summarize an lmList Object*

Description

The summary.lm method is applied to each lm component of object to produce summary information on the individual fits, which is organized into a list of summary statistics. The returned object is suitable for printing with the print.summary.lmList method.

Usage

```
## S3 method for class 'lmList'
summary(object, pool, ...)
```

Arguments

object	an object inheriting from class "lmList", representing a list of lm fitted objects.
pool	an optional logical value indicating whether a pooled estimate of the residual standard error should be used. Default is attr(object, "pool").
...	some methods for this generic require additional arguments. None are used in this method.

Value

a list with summary statistics obtained by applying `summary.lm` to the elements of `object`, inheriting from class `summary.lmList`. The components of `value` are:

`call`	a list containing an image of the `lmList` call that produced `object`.
`coefficients`	a three dimensional array with summary information on the `lm` coefficients. The first dimension corresponds to the names of the `object` components, the second dimension is given by `"Value"`, `"Std. Error"`, `"t value"`, and `"Pr(>|t|)"`, corresponding, respectively, to the coefficient estimates and their associated standard errors, t-values, and p-values. The third dimension is given by the coefficients names.
`correlation`	a three dimensional array with the correlations between the individual `lm` coefficient estimates. The first dimension corresponds to the names of the `object` components. The third dimension is given by the coefficients names. For each coefficient, the rows of the associated array give the correlations between that coefficient and the remaining coefficients, by `lm` component.
`cov.unscaled`	a three dimensional array with the unscaled variances/covariances for the individual `lm` coefficient estimates (giving the estimated variance/covariance for the coefficients, when multiplied by the estimated residual errors). The first dimension corresponds to the names of the `object` components. The third dimension is given by the coefficients names. For each coefficient, the rows of the associated array give the unscaled covariances between that coefficient and the remaining coefficients, by `lm` component.
`df`	an array with the number of degrees of freedom for the model and for residuals, for each `lm` component.
`df.residual`	the total number of degrees of freedom for residuals, corresponding to the sum of residuals df of all `lm` components.
`fstatistics`	an array with the F test statistics and corresponding degrees of freedom, for each `lm` component.
`pool`	the value of the `pool` argument to the function.
`r.squared`	a vector with the multiple R-squared statistics for each `lm` component.
`residuals`	a list with components given by the residuals from individual `lm` fits.
`RSE`	the pooled estimate of the residual standard error.
`sigma`	a vector with the residual standard error estimates for the individual `lm` fits.
`terms`	the terms object used in fitting the individual `lm` components.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`lmList`, `summary`

Examples

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
summary(fm1)
```

`summary.modelStruct` *Summarize a modelStruct Object*

Description

This method function applies `summary` to each element of `object`.

Usage

```
## S3 method for class 'modelStruct'
summary(object, ...)
```

Arguments

`object`	an object inheriting from class `"modelStruct"`, representing a list of model components, such as `reStruct`, `corStruct` and `varFunc` objects.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a list with elements given by the summarized components of `object`. The returned value is of class `summary.modelStruct`, also inheriting from the same classes as `object`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`reStruct`, `summary`

Examples

```
lms1 <- lmeStruct(reStruct = reStruct(pdDiag(diag(2), ~age)),
   corStruct = corAR1(0.3))
summary(lms1)
```

summary.nlsList *Summarize an nlsList Object*

Description

The summary function is applied to each nls component of object to produce summary information on the individual fits, which is organized into a list of summary statistics. The returned object is suitable for printing with the print.summary.nlsList method.

Usage

```
## S3 method for class 'nlsList'
summary(object, ...)
```

Arguments

object	an object inheriting from class "nlsList", representing a list of nls fitted objects.
...	optional arguments to the summary.lmList method. One such optional argument is pool, a logical value indicating whether a pooled estimate of the residual standard error should be used. Default is attr(object, "pool").

Value

a list with summary statistics obtained by applying summary to the elements of object, inheriting from class summary.nlsList. The components of value are:

call	a list containing an image of the nlsList call that produced object.
parameters	a three dimensional array with summary information on the nls coefficients. The first dimension corresponds to the names of the object components, the second dimension is given by "Value", "Std. Error", "t value", and "Pr(>|t|)", corresponding, respectively, to the coefficient estimates and their associated standard errors, t-values, and p-values. The third dimension is given by the coefficients names.
correlation	a three dimensional array with the correlations between the individual nls coefficient estimates. The first dimension corresponds to the names of the object components. The third dimension is given by the coefficients names. For each coefficient, the rows of the associated array give the correlations between that coefficient and the remaining coefficients, by nls component.
cov.unscaled	a three dimensional array with the unscaled variances/covariances for the individual lm coefficient estimates (giving the estimated variance/covariance for the coefficients, when multiplied by the estimated residual errors). The first dimension corresponds to the names of the object components. The third dimension is given by the coefficients names. For each coefficient, the rows of the associated array give the unscaled covariances between that coefficient and the remaining coefficients, by nls component.

df	an array with the number of degrees of freedom for the model and for residuals, for each nls component.
df.residual	the total number of degrees of freedom for residuals, corresponding to the sum of residuals df of all nls components.
pool	the value of the pool argument to the function.
RSE	the pooled estimate of the residual standard error.
sigma	a vector with the residual standard error estimates for the individual lm fits.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

nlsList, summary

Examples

```
fm1 <- nlsList(SSasymp, Loblolly)
summary(fm1)
```

---

summary.pdMat *Summarize a pdMat Object*

Description

Attributes structName and noCorrelation, with the values of the corresponding arguments to the method function, are appended to object and its class is changed to summary.pdMat.

Usage

```
## S3 method for class 'pdMat'
summary(object, structName, noCorrelation, ...)
```

Arguments

object	an object inheriting from class "pdMat", representing a positive definite matrix.
structName	an optional character string with a description of the pdMat class. Default depends on the method function: "Blocked" for pdBlocked, "Compound Symmetry" for pdCompSymm, "Diagonal" for pdDiag, "Multiple of an Identity" for pdIdent, "General Positive-Definite, Natural Parametrization" for pdNatural, "General Positive-Definite" for pdSymm, and data.class(object) for pdMat.
noCorrelation	an optional logical value indicating whether correlations are to be printed in print.summary.pdMat. Default depends on the method function: FALSE for pdDiag and pdIdent, and TRUE for all other classes.
...	some methods for this generic require additional arguments. None are used in this method.

Value

an object similar to `object`, with additional attributes `structName` and `noCorrelation`, inheriting from class `summary.pdMat`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`print.summary.pdMat`, `pdMat`

Examples

```
summary(pdSymm(diag(4)))
```

`summary.varFunc` *Summarize "varFunc" Object*

Description

A `structName` attribute, with the value of corresponding argument, is appended to `object` and its class is changed to `summary.varFunc`.

Usage

```
## S3 method for class 'varFunc'
summary(object, structName, ...)
```

Arguments

`object`	an object inheriting from class `"varFunc"`, representing a variance function structure.
`structName`	an optional character string with a description of the `varFunc` class. Default depends on the method function: **for** `varComb`**:** `"Combination of variance functions"`, **for** `varConstPower`**:** `"Constant plus power of variance covariate"`, **for** `varConstProp`**:** `"Constant plus proportion of variance covariate"`, **for** `varExp`**:** `"Exponential of variance covariate"`, **for** `varIdent`**:** `"Different standard deviations per stratum"`, **for** `varPower`**:** `"Power of variance covariate"`, **for** `varFunc`**:** `data.class(object)`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

an object similar to `object`, with an additional attribute `structName`, inheriting from class `summary.varFunc`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`varClasses`, `varFunc`

Examples

```
vf1 <- varPower(0.3, form = ~age)
vf1 <- Initialize(vf1, Orthodont)
summary(vf1)
```

Tetracycline1 *Pharmacokinetics of tetracycline*

Description

The `Tetracycline1` data frame has 40 rows and 4 columns.

Format

This data frame contains the following columns:

conc a numeric vector

Time a numeric vector

Subject an ordered factor with levels 5 < 3 < 2 < 4 < 1

Formulation a factor with levels `tetrachel` `tetracyn`

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

Tetracycline2 *Pharmacokinetics of tetracycline*

Description

The Tetracycline2 data frame has 40 rows and 4 columns.

Format

This data frame contains the following columns:

conc a numeric vector

Time a numeric vector

Subject an ordered factor with levels 4 < 5 < 2 < 1 < 3

Formulation a factor with levels Berkmycin tetramycin

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

update.modelStruct *Update a modelStruct Object*

Description

This method function updates each element of object, allowing the access to data.

Usage

```
## S3 method for class 'modelStruct'
update(object, data, ...)
```

Arguments

object	an object inheriting from class "modelStruct", representing a list of model components, such as corStruct and varFunc objects.
data	a data frame in which to evaluate the variables needed for updating the elements of object.
...	some methods for this generic require additional arguments. None are used in this method.

Value

an object similar to object (same class, length, and names), but with updated elements.

Note

This method function is primarily used within model fitting functions, such as `lme` and `gls`, that allow model components such as variance functions.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

`reStruct`

`update.varFunc` *Update varFunc Object*

Description

If the `formula(object)` includes a `"."` term, representing a fitted object, the variance covariate needs to be updated upon completion of an optimization cycle (in which the variance function weights are kept fixed). This method function allows a reevaluation of the variance covariate using the current fitted object and, optionally, other variables in the original data.

Usage

```
## S3 method for class 'varFunc'
update(object, data, ...)
```

Arguments

`object`	an object inheriting from class `"varFunc"`, representing a variance function structure.
`data`	a list with a component named `"."` with the current version of the fitted object (from which fitted values, coefficients, and residuals can be extracted) and, if necessary, other variables used to evaluate the variance covariate(s).
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

if `formula(object)` includes a `"."` term, an `varFunc` object similar to `object`, but with the variance covariate reevaluated at the current fitted object value; else `object` is returned unchanged.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

needUpdate, covariate<-.varFunc

---

varClasses *Variance Function Classes*

Description

Standard classes of variance function structures (varFunc) available in the **nlme** package. Covariates included in the variance function, denoted by variance covariates, may involve functions of the fitted model object, such as the fitted values and the residuals. Different coefficients may be assigned to the levels of a classification factor.

Value

Available standard classes:

varExp	exponential of a variance covariate.
varPower	power of a variance covariate.
varConstPower	constant plus power of a variance covariate.
varConstProp	constant plus proportion of a variance covariate.
varIdent	constant variance(s), generally used to allow different variances according to the levels of a classification factor.
varFixed	fixed weights, determined by a variance covariate.
varComb	combination of variance functions.

Note

Users may define their own varFunc classes by specifying a constructor function and, at a minimum, methods for the functions coef, coef<-, and Initialize. For examples of these functions, see the methods for class varPower.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

varComb, varConstPower, varConstProp, varExp, varFixed, varIdent, varPower, summary.varFunc

varComb *Combination of Variance Functions*

Description

This function is a constructor for the varComb class, representing a combination of variance functions. The corresponding variance function is equal to the product of the variance functions of the varFunc objects listed in ....

Usage

```
varComb(...)
```

Arguments

...	objects inheriting from class varFunc representing variance function structures.

Value

a varComb object representing a combination of variance functions, also inheriting from class varFunc.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

varClasses, varWeights.varComb, coef.varComb

Examples

```
vf1 <- varComb(varIdent(form = ~1|Sex), varPower())
```

varConstPower *Constant Plus Power Variance Function*

Description

This function is a constructor for the varConstPower class, representing a constant plus power variance function structure. Letting v denote the variance covariate and $\sigma^2(v)$ denote the variance function evaluated at v, the constant plus power variance function is defined as $\sigma^2(v) = (\theta_1 + |v|_2^{\theta})^2$, where θ_1, θ_2 are the variance function coefficients. When a grouping factor is present, different θ_1, θ_2 are used for each factor level.

Usage

```
varConstPower(const, power, form, fixed)
```

Arguments

const, power	optional numeric vectors, or lists of numeric values, with, respectively, the coefficients for the constant and the power terms. Both arguments must have length one, unless a grouping factor is specified in form. If either argument has length greater than one, it must have names which identify its elements to the levels of the grouping factor defined in form. If a grouping factor is present in form and the argument has length one, its value will be assigned to all grouping levels. Only positive values are allowed for const. Default is numeric(0), which results in a vector of zeros of appropriate length being assigned to the coefficients when object is initialized (corresponding to constant variance equal to one).
form	an optional one-sided formula of the form ~ v, or ~ v | g, specifying a variance covariate v and, optionally, a grouping factor g for the coefficients. The variance covariate must evaluate to a numeric vector and may involve expressions using ".", representing a fitted model object from which fitted values (fitted(.)) and residuals (resid(.)) can be extracted (this allows the variance covariate to be updated during the optimization of an object function). When a grouping factor is present in form, a different coefficient value is used for each of its levels. Several grouping variables may be simultaneously specified, separated by the * operator, as in ~ v | g1 * g2 * g3. In this case, the levels of each grouping variable are pasted together and the resulting factor is used to group the observations. Defaults to ~ fitted(.) representing a variance covariate given by the fitted values of a fitted model object and no grouping factor.
fixed	an optional list with components const and/or power, consisting of numeric vectors, or lists of numeric values, specifying the values at which some or all of the coefficients in the variance function should be fixed. If a grouping factor is specified in form, the components of fixed must have names identifying which coefficients are to be fixed. Coefficients included in fixed are not allowed to vary during the optimization of an objective function. Defaults to NULL, corresponding to no fixed coefficients.

Value

a varConstPower object representing a constant plus power variance function structure, also inheriting from class varFunc.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

varClasses, varWeights.varFunc, coef.varConstPower

Examples

```
vf1 <- varConstPower(1.2, 0.2, form = ~age|Sex)
```

---

varConstProp *Constant Plus Proportion Variance Function*

Description

This function is a constructor for the varConstProp class, representing a variance function structure corresponding to a two-component error model (additive and proportional error). Letting v denote the variance covariate and $\sigma^2(v)$ denote the variance function evaluated at v, the two-component variance function is defined as $\sigma^2(v) = a^2 + b^2 * v^2$, where a is the additive component and b is the relative error component. In order to avoid overparameterisation of the model, it is recommended to use the possibility to fix sigma, preferably to a value of 1 (see examples).

Usage

```
varConstProp(const, prop, form, fixed)
```

Arguments

const, prop	optional numeric vectors, or lists of numeric values, with, respectively, the coefficients for the constant and the proportional error terms. Both arguments must have length one, unless a grouping factor is specified in form. If either argument has length greater than one, it must have names which identify its elements to the levels of the grouping factor defined in form. If a grouping factor is present in form and the argument has length one, its value will be assigned to all grouping levels. Only positive values are allowed for const. Default is 0.1 for both const and prop.

`form` an optional one-sided formula of the form `~ v`, or `~ v | g`, specifying a variance covariate `v` and, optionally, a grouping factor `g` for the coefficients. The variance covariate must evaluate to a numeric vector and may involve expressions using `"."`, representing a fitted model object from which fitted values (`fitted(.)`) and residuals (`resid(.)`) can be extracted (this allows the variance covariate to be updated during the optimization of an object function). When a grouping factor is present in `form`, a different coefficient value is used for each of its levels. Several grouping variables may be simultaneously specified, separated by the `*` operator, as in `~ v | g1 * g2 * g3`. In this case, the levels of each grouping variable are pasted together and the resulting factor is used to group the observations. Defaults to `~ fitted(.)` representing a variance covariate given by the fitted values of a fitted model object and no grouping factor.

`fixed` an optional list with components `const` and/or `power`, consisting of numeric vectors, or lists of numeric values, specifying the values at which some or all of the coefficients in the variance function should be fixed. If a grouping factor is specified in `form`, the components of `fixed` must have names identifying which coefficients are to be fixed. Coefficients included in `fixed` are not allowed to vary during the optimization of an objective function. Defaults to `NULL`, corresponding to no fixed coefficients.

Value

a `varConstProp` object representing a constant plus proportion variance function structure, also inheriting from class `varFunc`.

Note

The error model underlying this variance function structure can be understood to result from two uncorrelated sequences of standardized random variables (Lavielle(2015), p. 55) and has been proposed for use in analytical chemistry (Werner et al. (1978), Wilson et al. (2004)) and chemical degradation kinetics (Ranke and Meinecke (2019)). Note that the two-component error model proposed by Rocke and Lorenzato (1995) assumed a log-normal distribution of residuals at high absolute values, which is not compatible with the `varFunc` structures in package **nlme**.

Author(s)

José Pinheiro and Douglas Bates (for `varConstPower`) and Johannes Ranke (adaptation to `varConstProp()`).

References

Lavielle, M. (2015) *Mixed Effects Models for the Population Approach: Models, Tasks, Methods and Tools*, Chapman and Hall/CRC. doi:10.1201/b17203

Pinheiro, J.C., and Bates, D.M. (2000) *Mixed-Effects Models in S and S-PLUS*, Springer. doi:10.1007/b98882

Ranke, J., and Meinecke, S. (2019) Error Models for the Kinetic Evaluation of Chemical Degradation Data. *Environments* **6**(12), 124 doi:10.3390/environments6120124

Rocke, David M. and Lorenzato, Stefan (1995) A Two-Component Model for Measurement Error in Analytical Chemistry. *Technometrics* **37**(2), 176–184. doi:10.1080/00401706.1995.10484302

Werner, Mario, Brooks, Samuel H., and Knott, Lancaster B. (1978) Additive, Multiplicative, and Mixed Analytical Errors. *Clinical Chemistry* **24**(11), 1895–1898. doi:10.1093/clinchem/24.11.1895

Wilson, M.D., Rocke, D.M., Durbin, B. and Kahn, H.D (2004) Detection Limits and Goodness-of-Fit Measures for the Two-Component Model of Chemical Analytical Error. *Analytica Chimica Acta* 2004, 509, 197–208 doi:10.1016/j.aca.2003.12.047

See Also

varClasses, varWeights.varFunc, coef.varFunc

Examples

```
# Generate some synthetic data using the two-component error model and use
# different variance functions, also with fixed sigma in order to avoid
# overparameterisation in the case of a constant term in the variance function
times <- c(0, 1, 3, 7, 14, 28, 56, 120)
pred <- 100 * exp(- 0.03 * times)
sd_pred <- sqrt(3^2 + 0.07^2 * pred^2)
n_replicates <- 2

set.seed(123456)
syn_data <- data.frame(
  time = rep(times, each = n_replicates),
  value = rnorm(length(times) * n_replicates,
    rep(pred, each = n_replicates),
    rep(sd_pred, each = n_replicates)))
syn_data$value <- ifelse(syn_data$value < 0, NA, syn_data$value)

f_const <- gnls(value ~ SSasymp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3))
f_varPower <- gnls(value ~ SSasymp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3),
  weights = varPower())
f_varConstPower <- gnls(value ~ SSasymp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3),
  weights = varConstPower())
f_varConstPower_sf <- gnls(value ~ SSasymp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  control = list(sigma = 1),
  start = list(parent_0 = 100, lrc = -3),
  weights = varConstPower())
f_varConstProp <- gnls(value ~ SSasymp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3),
  weights = varConstProp())
f_varConstProp_sf <- gnls(value ~ SSasymp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3),
  control = list(sigma = 1),
```

```
  weights = varConstProp())

AIC(f_const, f_varPower, f_varConstPower, f_varConstPower_sf,
  f_varConstProp, f_varConstProp_sf)

# The error model parameters 3 and 0.07 are approximately recovered
intervals(f_varConstProp_sf)
```

VarCorr *Extract variance and correlation components*

Description

This function calculates the estimated variances, standard deviations, and correlations between the random-effects terms in a linear mixed-effects model, of class "lme", or a nonlinear mixed-effects model, of class "nlme". The within-group error variance and standard deviation are also calculated.

Usage

```
VarCorr(x, sigma = 1, ...)
## S3 method for class 'lme'
VarCorr(x, sigma = x$sigma, rdig = 3, ...)

## S3 method for class 'pdMat'
VarCorr(x, sigma = 1, rdig = 3, ...)
## S3 method for class 'pdBlocked'
VarCorr(x, sigma = 1, rdig = 3, ...)
```

Arguments

x	a fitted model object, usually an object inheriting from class "lme".
sigma	an optional numeric value used as a multiplier for the standard deviations. The default is x$sigma or 1 depending on class(x).
rdig	an optional integer value specifying the number of digits used to represent correlation estimates. Default is 3.
...	further optional arguments passed to other methods (none for the methods documented here).

Value

a matrix with the estimated variances, standard deviations, and correlations for the random effects. The first two columns, named Variance and StdDev, give, respectively, the variance and the standard deviations. If there are correlation components in the random effects model, the third column, named Corr, and the remaining unnamed columns give the estimated correlations among random effects within the same level of grouping. The within-group error variance and standard deviation are included as the last row in the matrix.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) *Mixed-Effects Models in S and S-PLUS*, Springer, esp. pp. 100, 461.

See Also

lme, nlme

Examples

```
fm1 <- lme(distance ~ age, data = Orthodont, random = ~age)
VarCorr(fm1)
```

---

varExp *Exponential Variance Function*

Description

This function is a constructor for the varExp class, representing an exponential variance function structure. Letting v denote the variance covariate and $\sigma^2(v)$ denote the variance function evaluated at v, the exponential variance function is defined as $\sigma^2(v) = \exp(2\theta v)$, where θ is the variance function coefficient. When a grouping factor is present, a different θ is used for each factor level.

Usage

```
varExp(value, form, fixed)
```

Arguments

value	an optional numeric vector, or list of numeric values, with the variance function coefficients. Value must have length one, unless a grouping factor is specified in form. If value has length greater than one, it must have names which identify its elements to the levels of the grouping factor defined in form. If a grouping factor is present in form and value has length one, its value will be assigned to all grouping levels. Default is numeric(0), which results in a vector of zeros of appropriate length being assigned to the coefficients when object is initialized (corresponding to constant variance equal to one).
form	an optional one-sided formula of the form ~ v, or ~ v | g, specifying a variance covariate v and, optionally, a grouping factor g for the coefficients. The variance covariate must evaluate to a numeric vector and may involve expressions using ".", representing a fitted model object from which fitted values (fitted(.)) and residuals (resid(.)) can be extracted (this allows the variance covariate to be updated during the optimization of an object function). When a grouping

factor is present in `form`, a different coefficient value is used for each of its levels. Several grouping variables may be simultaneously specified, separated by the `*` operator, like in `~ v | g1 * g2 * g3`. In this case, the levels of each grouping variable are pasted together and the resulting factor is used to group the observations. Defaults to `~ fitted(.)` representing a variance covariate given by the fitted values of a fitted model object and no grouping factor.

`fixed` an optional numeric vector, or list of numeric values, specifying the values at which some or all of the coefficients in the variance function should be fixed. If a grouping factor is specified in `form`, `fixed` must have names identifying which coefficients are to be fixed. Coefficients included in `fixed` are not allowed to vary during the optimization of an objective function. Defaults to `NULL`, corresponding to no fixed coefficients.

Value

a `varExp` object representing an exponential variance function structure, also inheriting from class `varFunc`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`varClasses`, `varWeights.varFunc`, `coef.varExp`

Examples

```
vf1 <- varExp(0.2, form = ~age|Sex)
```

---

varFixed *Fixed Variance Function*

Description

This function is a constructor for the `varFixed` class, representing a variance function with fixed variances. Letting v denote the variance covariate defined in `value`, the variance function $\sigma^2(v)$ for this class is $\sigma^2(v) = |v|$. The variance covariate v is evaluated once at initialization and remains fixed thereafter. No coefficients are required to represent this variance function.

Usage

```
varFixed(value)
```

Arguments

value	a one-sided formula of the form ~ v specifying a variance covariate v. Grouping factors are ignored.

Value

a varFixed object representing a fixed variance function structure, also inheriting from class varFunc.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

varClasses, varWeights.varFunc, varFunc

Examples

```
vf1 <- varFixed(~age)
```

---

varFunc *Variance Function Structure*

Description

If object is a one-sided formula, it is used as the argument to varFixed and the resulting object is returned. Else, if object already inherits from class "varFunc", such as all standard 'varClasses', it is returned unchanged.

Usage

```
varFunc(object)
```

Arguments

object	either an one-sided formula specifying a variance covariate, or an object inheriting from class "varFunc", representing a variance function structure.

Value

an object from class "varFunc", representing a variance function structure.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

summary.varFunc, varFixed, varWeights.varFunc, coef.varFunc

Examples

```
vf1 <- varFunc(~age)
```

---

varIdent *Constant Variance Function*

Description

This function is a constructor for the varIdent class, representing a constant variance function structure. If no grouping factor is present in form, the variance function is constant and equal to one, and no coefficients required to represent it. When form includes a grouping factor with $M > 1$ levels, the variance function allows M different variances, one for each level of the factor. For identifiability reasons, the coefficients of the variance function represent the ratios between the variances and a reference variance (corresponding to a reference group level). Therefore, only $M - 1$ coefficients are needed to represent the variance function. By default, if the elements in value are unnamed, the first group level is taken as the reference level.

Usage

```
varIdent(value, form, fixed)
```

Arguments

value	an optional numeric vector, or list of numeric values, with the variance function coefficients. If no grouping factor is present in form, this argument is ignored, as the resulting variance function contains no coefficients. If value has length one, its value is repeated for all coefficients in the variance function. If value has length greater than one, it must have length equal to the number of grouping levels minus one and names which identify its elements to the levels of the grouping factor. Only positive values are allowed for this argument. Default is numeric(0), which results in a vector of zeros of appropriate length being assigned to the coefficients when object is initialized (corresponding to constant variance equal to one).
form	an optional one-sided formula of the form ~ v, or ~ v | g, specifying a variance covariate v and, optionally, a grouping factor g for the coefficients. The variance covariate is ignored in this variance function. When a grouping factor is present in form, a different coefficient value is used for each of its levels less one reference level. Several grouping variables may be simultaneously specified, separated by the * operator, like in ~ v | g1 * g2 * g3. In this case, the levels of each grouping variable are pasted together and the resulting factor is used to group the observations. Defaults to ~ 1.

`fixed` an optional numeric vector, or list of numeric values, specifying the values at which some or all of the coefficients in the variance function should be fixed. It must have names identifying which coefficients are to be fixed. Coefficients included in `fixed` are not allowed to vary during the optimization of an objective function. Defaults to `NULL`, corresponding to no fixed coefficients.

Value

a `varIdent` object representing a constant variance function structure, also inheriting from class `varFunc`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`varClasses`, `varWeights.varFunc`, `coef.varIdent`

Examples

```
vf1 <- varIdent(c(Female = 0.5), form = ~ 1 | Sex)
```

Variogram *Calculate Semi-variogram*

Description

This function is generic; method functions can be written to handle specific classes of objects. Classes which already have methods for this function include `"gls"` and `"lme"`; there is also a default method for arbitrary vectors (of residuals). See the appropriate method documentation for a description of the arguments.

Usage

```
Variogram(object, distance, ...)
```

Arguments

`object` a numeric vector with the values to be used for calculating the semi-variogram, usually a residual vector from a fitted model.

`distance` a numeric vector with the pairwise distances corresponding to the elements of `object`. The order of the elements in `distance` must correspond to the pairs `(1,2)`, `(1,3)`, `...`, `(n-1,n)`, with `n` representing the length of `object`, and must have length `n(n-1)/2`.

`...` some methods for this generic function require additional arguments.

Value

will depend on the method function used; see the appropriate documentation.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

Variogram.corExp, Variogram.corGaus, Variogram.corLin, Variogram.corRatio, Variogram.corSpatial, Variogram.corSpher, Variogram.default, Variogram.gls, Variogram.lme, plot.Variogram

---

Variogram.corExp *Calculate Semi-variogram for a corExp Object*

Description

This method function calculates the semi-variogram values corresponding to the Exponential correlation model, using the estimated coefficients corresponding to `object`, at the distances defined by `distance`.

Usage

```
## S3 method for class 'corExp'
Variogram(object, distance, sig2, length.out, ...)
```

Arguments

`object`	an object inheriting from class *"corExp"*, representing an exponential spatial correlation structure.
`distance`	an optional numeric vector with the distances at which the semi-variogram is to be calculated. Defaults to `NULL`, in which case a sequence of length `length.out` between the minimum and maximum values of `getCovariate(object)` is used.
`sig2`	an optional numeric value representing the process variance. Defaults to `1`.
`length.out`	an optional integer specifying the length of the sequence of distances to be used for calculating the semi-variogram, when `distance = NULL`. Defaults to `50`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns `variog` and `dist` representing, respectively, the semi-variogram values and the corresponding distances. The returned value inherits from class `Variogram`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

`corExp`, `plot.Variogram`, `Variogram`

Examples

```
stopifnot(require("stats", quietly = TRUE))
cs1 <- corExp(3, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]
```

Variogram.corGaus *Calculate Semi-variogram for a corGaus Object*

Description

This method function calculates the semi-variogram values corresponding to the Gaussian correlation model, using the estimated coefficients corresponding to `object`, at the distances defined by `distance`.

Usage

```
## S3 method for class 'corGaus'
Variogram(object, distance, sig2, length.out, ...)
```

Arguments

`object`	an object inheriting from class `"corGaus"`, representing an Gaussian spatial correlation structure.
`distance`	an optional numeric vector with the distances at which the semi-variogram is to be calculated. Defaults to `NULL`, in which case a sequence of length `length.out` between the minimum and maximum values of `getCovariate(object)` is used.
`sig2`	an optional numeric value representing the process variance. Defaults to `1`.
`length.out`	an optional integer specifying the length of the sequence of distances to be used for calculating the semi-variogram, when `distance = NULL`. Defaults to `50`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns `variog` and `dist` representing, respectively, the semi-variogram values and the corresponding distances. The returned value inherits from class `Variogram`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

`corGaus`, `plot.Variogram`, `Variogram`

Examples

```
cs1 <- corGaus(3, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]
```

Variogram.corLin *Calculate Semi-variogram for a corLin Object*

Description

This method function calculates the semi-variogram values corresponding to the Linear correlation model, using the estimated coefficients corresponding to `object`, at the distances defined by `distance`.

Usage

```
## S3 method for class 'corLin'
Variogram(object, distance, sig2, length.out, ...)
```

Arguments

`object`	an object inheriting from class "`corLin`", representing an Linear spatial correlation structure.
`distance`	an optional numeric vector with the distances at which the semi-variogram is to be calculated. Defaults to `NULL`, in which case a sequence of length `length.out` between the minimum and maximum values of `getCovariate(object)` is used.
`sig2`	an optional numeric value representing the process variance. Defaults to `1`.
`length.out`	an optional integer specifying the length of the sequence of distances to be used for calculating the semi-variogram, when `distance = NULL`. Defaults to `50`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns `variog` and `dist` representing, respectively, the semi-variogram values and the corresponding distances. The returned value inherits from class `Variogram`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

`corLin`, `plot.Variogram`, `Variogram`

Examples

```
cs1 <- corLin(15, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]
```

Variogram.corRatio *Calculate Semi-variogram for a corRatio Object*

Description

This method function calculates the semi-variogram values corresponding to the Rational Quadratic correlation model, using the estimated coefficients corresponding to `object`, at the distances defined by `distance`.

Usage

```
## S3 method for class 'corRatio'
Variogram(object, distance, sig2, length.out, ...)
```

Arguments

`object`	an object inheriting from class "`corRatio`", representing an Rational Quadratic spatial correlation structure.
`distance`	an optional numeric vector with the distances at which the semi-variogram is to be calculated. Defaults to `NULL`, in which case a sequence of length `length.out` between the minimum and maximum values of `getCovariate(object)` is used.
`sig2`	an optional numeric value representing the process variance. Defaults to `1`.
`length.out`	an optional integer specifying the length of the sequence of distances to be used for calculating the semi-variogram, when `distance = NULL`. Defaults to `50`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns `variog` and `dist` representing, respectively, the semi-variogram values and the corresponding distances. The returned value inherits from class `Variogram`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

`corRatio`, `plot.Variogram` `Variogram`

Examples

```
cs1 <- corRatio(7, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]
```

Variogram.corSpatial *Calculate Semi-variogram for a corSpatial Object*

Description

This method function calculates the semi-variogram values corresponding to the model defined in `FUN`, using the estimated coefficients corresponding to `object`, at the distances defined by `distance`.

Usage

```
## S3 method for class 'corSpatial'
Variogram(object, distance, sig2, length.out, FUN, ...)
```

Arguments

`object`	an object inheriting from class `"corSpatial"`, representing spatial correlation structure.
`distance`	an optional numeric vector with the distances at which the semi-variogram is to be calculated. Defaults to `NULL`, in which case a sequence of length `length.out` between the minimum and maximum values of `getCovariate(object)` is used.
`sig2`	an optional numeric value representing the process variance. Defaults to `1`.
`length.out`	an optional integer specifying the length of the sequence of distances to be used for calculating the semi-variogram, when `distance = NULL`. Defaults to `50`.
`FUN`	a function of two arguments, the distance and the range corresponding to `object`, specifying the semi-variogram model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns `variog` and `dist` representing, respectively, the semi-variogram values and the corresponding distances. The returned value inherits from class `Variogram`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

`corSpatial`, `Variogram`, `Variogram.default`, `Variogram.corExp`, `Variogram.corGaus`, `Variogram.corLin`, `Variogram.corRatio`, `Variogram.corSpher`, `plot.Variogram`

Examples

```
cs1 <- corExp(3, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1, FUN = function(x, y) (1 - exp(-x/y)))[1:10,]
```

Variogram.corSpher *Calculate Semi-variogram for a corSpher Object*

Description

This method function calculates the semi-variogram values corresponding to the Spherical correlation model, using the estimated coefficients corresponding to `object`, at the distances defined by `distance`.

Usage

```
## S3 method for class 'corSpher'
Variogram(object, distance, sig2, length.out, ...)
```

Arguments

`object`	an object inheriting from class `"corSpher"`, representing an Spherical spatial correlation structure.
`distance`	an optional numeric vector with the distances at which the semi-variogram is to be calculated. Defaults to `NULL`, in which case a sequence of length `length.out` between the minimum and maximum values of `getCovariate(object)` is used.
`sig2`	an optional numeric value representing the process variance. Defaults to `1`.
`length.out`	an optional integer specifying the length of the sequence of distances to be used for calculating the semi-variogram, when `distance = NULL`. Defaults to `50`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns `variog` and `dist` representing, respectively, the semi-variogram values and the corresponding distances. The returned value inherits from class `Variogram`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

`corSpher`, `plot.Variogram`, `Variogram`

Examples

```
cs1 <- corSpher(15, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]
```

Variogram.default *Calculate Semi-variogram*

Description

This method function calculates the semi-variogram for an arbitrary vector `object`, according to the distances in `distance`. For each pair of elements x, y in `object`, the corresponding semi-variogram is $(x - y)^2/2$. The semi-variogram is useful for identifying and modeling spatial correlation structures in observations with constant expectation and constant variance.

Usage

```
## Default S3 method:
Variogram(object, distance, ...)
```

Arguments

`object`	a numeric vector with the values to be used for calculating the semi-variogram, usually a residual vector from a fitted model.
`distance`	a numeric vector with the pairwise distances corresponding to the elements of `object`. The order of the elements in `distance` must correspond to the pairs `(1,2)`, `(1,3)`, `...`, `(n-1,n)`, with `n` representing the length of `object`, and must have length `n(n-1)/2`.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns `variog` and `dist` representing, respectively, the semi-variogram values and the corresponding distances. The returned value inherits from class `Variogram`.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

`Variogram`, `Variogram.gls`, `Variogram.lme`, `plot.Variogram`

Examples

```
fm1 <- lm(follicles ~ sin(2 * pi * Time) + cos(2 * pi * Time), Ovary,
          subset = Mare == 1)
Variogram(resid(fm1), dist(1:29))[1:10,]
```

Variogram.gls *Calculate Semi-variogram for Residuals from a gls Object*

Description

This method function calculates the semi-variogram for the residuals from a `gls` fit. The semi-variogram values are calculated for pairs of residuals within the same group level, if a grouping factor is present. If `collapse` is different from `"none"`, the individual semi-variogram values are collapsed using either a robust estimator (`robust = TRUE`) defined in Cressie (1993), or the average of the values within the same distance interval. The semi-variogram is useful for modeling the error term correlation structure.

Usage

```
## S3 method for class 'gls'
Variogram(object, distance, form, resType, data,
      na.action, maxDist, length.out, collapse, nint, breaks,
      robust, metric, ...)
```

Arguments

`object`	an object inheriting from class `"gls"`, representing a generalized least squares fitted model.

distance — an optional numeric vector with the distances between residual pairs. If a grouping variable is present, only the distances between residual pairs within the same group should be given. If missing, the distances are calculated based on the values of the arguments form, data, and metric, unless object includes a corSpatial element, in which case the associated covariate (obtained with the getCovariate method) is used.

form — an optional one-sided formula specifying the covariate(s) to be used for calculating the distances between residual pairs and, optionally, a grouping factor for partitioning the residuals (which must appear to the right of a | operator in form). Default is ~1, implying that the observation order within the groups is used to obtain the distances.

resType — an optional character string specifying the type of residuals to be used. If "response", the "raw" residuals (observed - fitted) are used; else, if "pearson", the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if "normalized", the normalized residuals (standardized residuals pre-multiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to "pearson".

data — an optional data frame in which to interpret the variables in form. By default, the same data used to fit object is used.

na.action — a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes an error message to be printed and the function to terminate, if there are any incomplete observations.

maxDist — an optional numeric value for the maximum distance used for calculating the semi-variogram between two residuals. By default all residual pairs are included.

length.out — an optional integer value. When object includes a corSpatial element, its semi-variogram values are calculated and this argument is used as the length.out argument to the corresponding Variogram method. Defaults to 50.

collapse — an optional character string specifying the type of collapsing to be applied to the individual semi-variogram values. If equal to "quantiles", the semi-variogram values are split according to quantiles of the distance distribution, with equal number of observations per group, with possibly varying distance interval lengths. Else, if "fixed", the semi-variogram values are divided according to distance intervals of equal lengths, with possibly different number of observations per interval. Else, if "none", no collapsing is used and the individual semi-variogram values are returned. Defaults to "quantiles".

nint — an optional integer with the number of intervals to be used when collapsing the semi-variogram values. Defaults to 20.

robust — an optional logical value specifying if a robust semi-variogram estimator should be used when collapsing the individual values. If TRUE the robust estimator is used. Defaults to FALSE.

breaks — an optional numeric vector with the breakpoints for the distance intervals to be used in collapsing the semi-variogram values. If not missing, the option specified in collapse is ignored.

metric	an optional character string specifying the distance metric to be used. The currently available options are "euclidean" for the root sum-of-squares of distances; "maximum" for the maximum difference; and "manhattan" for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to "euclidean".
...	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns variog and dist representing, respectively, the semi-variogram values and the corresponding distances. If the semi-variogram values are collapsed, an extra column, n.pairs, with the number of residual pairs used in each semi-variogram calculation, is included in the returned data frame. If object includes a corSpatial element, a data frame with its corresponding semi-variogram is included in the returned value, as an attribute "modelVariog". The returned value inherits from class Variogram.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

gls, Variogram, Variogram.default, Variogram.lme, plot.Variogram

Examples

```
fm1 <- gls(weight ~ Time * Diet, BodyWeight)
Vm1 <- Variogram(fm1, form = ~ Time | Rat)
print(head(Vm1), digits = 3)
```

Variogram.lme *Calculate Semi-variogram for Residuals from an lme Object*

Description

This method function calculates the semi-variogram for the within-group residuals from an lme fit. The semi-variogram values are calculated for pairs of residuals within the same group. If collapse is different from "none", the individual semi-variogram values are collapsed using either a robust estimator (robust = TRUE) defined in Cressie (1993), or the average of the values within the same distance interval. The semi-variogram is useful for modeling the error term correlation structure.

Usage

```
## S3 method for class 'lme'
Variogram(object, distance, form, resType, data,
       na.action, maxDist, length.out, collapse, nint, breaks,
       robust, metric, ...)
```

Arguments

object	an object inheriting from class "lme", representing a fitted linear mixed-effects model.
distance	an optional numeric vector with the distances between residual pairs. If a grouping variable is present, only the distances between residual pairs within the same group should be given. If missing, the distances are calculated based on the values of the arguments form, data, and metric, unless object includes a corSpatial element, in which case the associated covariate (obtained with the getCovariate method) is used.
form	an optional one-sided formula specifying the covariate(s) to be used for calculating the distances between residual pairs and, optionally, a grouping factor for partitioning the residuals (which must appear to the right of a | operator in form). Default is ~1, implying that the observation order within the groups is used to obtain the distances.
resType	an optional character string specifying the type of residuals to be used. If "response", the "raw" residuals (observed - fitted) are used; else, if "pearson", the standardized residuals (raw residuals divided by the corresponding standard errors) are used; else, if "normalized", the normalized residuals (standardized residuals pre-multiplied by the inverse square-root factor of the estimated error correlation matrix) are used. Partial matching of arguments is used, so only the first character needs to be provided. Defaults to "pearson".
data	an optional data frame in which to interpret the variables in form. By default, the same data used to fit object is used.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes an error message to be printed and the function to terminate, if there are any incomplete observations.
maxDist	an optional numeric value for the maximum distance used for calculating the semi-variogram between two residuals. By default all residual pairs are included.
length.out	an optional integer value. When object includes a corSpatial element, its semi-variogram values are calculated and this argument is used as the length.out argument to the corresponding Variogram method. Defaults to 50.
collapse	an optional character string specifying the type of collapsing to be applied to the individual semi-variogram values. If equal to "quantiles", the semi-variogram values are split according to quantiles of the distance distribution, with equal number of observations per group, with possibly varying distance interval lengths. Else, if "fixed", the semi-variogram values are divided according to distance intervals of equal lengths, with possibly different number of observations per interval. Else, if "none", no collapsing is used and the individual semi-variogram values are returned. Defaults to "quantiles".

nint	an optional integer with the number of intervals to be used when collapsing the semi-variogram values. Defaults to 20.
robust	an optional logical value specifying if a robust semi-variogram estimator should be used when collapsing the individual values. If TRUE the robust estimator is used. Defaults to FALSE.
breaks	an optional numeric vector with the breakpoints for the distance intervals to be used in collapsing the semi-variogram values. If not missing, the option specified in collapse is ignored.
metric	an optional character string specifying the distance metric to be used. The currently available options are "euclidean" for the root sum-of-squares of distances; "maximum" for the maximum difference; and "manhattan" for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to "euclidean".
...	some methods for this generic require additional arguments. None are used in this method.

Value

a data frame with columns variog and dist representing, respectively, the semi-variogram values and the corresponding distances. If the semi-variogram values are collapsed, an extra column, n.pairs, with the number of residual pairs used in each semi-variogram calculation, is included in the returned data frame. If object includes a corSpatial element, a data frame with its corresponding semi-variogram is included in the returned value, as an attribute "modelVariog". The returned value inherits from class Variogram.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Cressie, N.A.C. (1993), "Statistics for Spatial Data", J. Wiley & Sons.

See Also

lme, Variogram, Variogram.default, Variogram.gls, plot.Variogram

Examples

```
fm1 <- lme(weight ~ Time * Diet, data=BodyWeight, ~ Time | Rat)
Variogram(fm1, form = ~ Time | Rat, nint = 10, robust = TRUE)
```

varPower *Power Variance Function*

Description

This function is a constructor for the varPower class, representing a power variance function structure. Letting v denote the variance covariate and $\sigma^2(v)$ denote the variance function evaluated at v, the power variance function is defined as $\sigma^2(v) = |v|^{2\theta}$, where θ is the variance function coefficient. When a grouping factor is present, a different θ is used for each factor level.

Usage

```
varPower(value, form, fixed)
```

Arguments

value	an optional numeric vector, or list of numeric values, with the variance function coefficients. Value must have length one, unless a grouping factor is specified in form. If value has length greater than one, it must have names which identify its elements to the levels of the grouping factor defined in form. If a grouping factor is present in form and value has length one, its value will be assigned to all grouping levels. Default is numeric(0), which results in a vector of zeros of appropriate length being assigned to the coefficients when object is initialized (corresponding to constant variance equal to one).
form	an optional one-sided formula of the form ~ v, or ~ v | g, specifying a variance covariate v and, optionally, a grouping factor g for the coefficients. The variance covariate must evaluate to a numeric vector and may involve expressions using ".", representing a fitted model object from which fitted values (fitted(.)) and residuals (resid(.)) can be extracted (this allows the variance covariate to be updated during the optimization of an object function). When a grouping factor is present in form, a different coefficient value is used for each of its levels. Several grouping variables may be simultaneously specified, separated by the * operator, like in ~ v | g1 * g2 * g3. In this case, the levels of each grouping variable are pasted together and the resulting factor is used to group the observations. Defaults to ~ fitted(.) representing a variance covariate given by the fitted values of a fitted model object and no grouping factor.
fixed	an optional numeric vector, or list of numeric values, specifying the values at which some or all of the coefficients in the variance function should be fixed. If a grouping factor is specified in form, fixed must have names identifying which coefficients are to be fixed. Coefficients included in fixed are not allowed to vary during the optimization of an objective function. Defaults to NULL, corresponding to no fixed coefficients.

Value

a varPower object representing a power variance function structure, also inheriting from class varFunc.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

varWeights.varFunc, coef.varPower

Examples

```
vf1 <- varPower(0.2, form = ~age|Sex)
```

---

varWeights *Extract Variance Function Weights*

Description

The inverse of the standard deviations corresponding to the variance function structure represented by object are returned.

Usage

```
varWeights(object)
```

Arguments

object	an object inheriting from class varFunc, representing a variance function structure.

Value

if object has a weights attribute, its value is returned; else NULL is returned.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

logLik.varFunc, varWeights

Examples

```
vf1 <- varPower(form=~age)
vf1 <- Initialize(vf1, Orthodont)
coef(vf1) <- 0.3
varWeights(vf1)[1:10]
```

`varWeights.glsStruct` *Variance Weights for glsStruct Object*

Description

If `object` includes a `varStruct` component, the inverse of the standard deviations of the variance function structure represented by the corresponding `varFunc` object are returned; else, a vector of ones of length equal to the number of observations in the data frame used to fit the associated linear model is returned.

Usage

```
## S3 method for class 'glsStruct'
varWeights(object)
```

Arguments

`object`	an object inheriting from class "`glsStruct`", representing a list of linear model components, such as `corStruct` and "`varFunc`" objects.

Value

if `object` includes a `varStruct` component, a vector with the corresponding variance weights; else, or a vector of ones.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`varWeights`

varWeights.lmeStruct *Variance Weights for lmeStruct Object*

Description

If `object` includes a `varStruct` component, the inverse of the standard deviations of the variance function structure represented by the corresponding `varFunc` object are returned; else, a vector of ones of length equal to the number of observations in the data frame used to fit the associated linear mixed-effects model is returned.

Usage

```
## S3 method for class 'lmeStruct'
varWeights(object)
```

Arguments

`object`	an object inheriting from class `"lmeStruct"`, representing a list of linear mixed-effects model components, such as `reStruct`, `corStruct`, and `varFunc` objects.

Value

if `object` includes a `varStruct` component, a vector with the corresponding variance weights; else, or a vector of ones.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

References

Pinheiro, J.C., and Bates, D.M. (2000) "Mixed-Effects Models in S and S-PLUS", Springer.

See Also

`varWeights`

Wafer *Modeling of Analog MOS Circuits*

Description

The Wafer data frame has 400 rows and 4 columns.

Format

This data frame contains the following columns:

Wafer a factor with levels 1 2 3 4 5 6 7 8 9 10

Site a factor with levels 1 2 3 4 5 6 7 8

voltage a numeric vector

current a numeric vector

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

Wheat *Yields by growing conditions*

Description

The Wheat data frame has 48 rows and 4 columns.

Format

This data frame contains the following columns:

Tray an ordered factor with levels 3 < 1 < 2 < 4 < 5 < 6 < 8 < 9 < 7 < 12 < 11 < 10

Moisture a numeric vector

fertilizer a numeric vector

DryMatter a numeric vector

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

Wheat2 *Wheat Yield Trials*

Description

The Wheat2 data frame has 224 rows and 5 columns.

Format

This data frame contains the following columns:

Block an ordered factor with levels 4 < 2 < 3 < 1

variety a factor with levels ARAPAHOE BRULE BUCKSKIN CENTURA CENTURK78 CHEYENNE CODY COLT GAGE HOMESTEAD KS831374 LANCER LANCOTA NE83404 NE83406 NE83407 NE83432 NE83498 NE83T12 NE84557 NE85556 NE85623 NE86482 NE86501 NE86503 NE86507 NE86509 NE86527 NE86582 NE86606 NE86607 NE86T666 NE87403 NE87408 NE87409 NE87446 NE87451 NE87457 NE87463 NE87499 NE87512 NE87513 NE87522 NE87612 NE87613 NE87615 NE87619 NE87627 NORKAN REDLAND ROUGHRIDER SCOUT66 SIOUXLAND TAM107 TAM200 VONA

yield a numeric vector

latitude a numeric vector

longitude a numeric vector

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York.

[.pdMat *Subscript a pdMat Object*

Description

This method function extracts sub-matrices from the positive-definite matrix represented by x.

Usage

```
## S3 method for class 'pdMat'
x[i, j, drop = TRUE]
## S3 replacement method for class 'pdMat'
x[i, j] <- value
```

Arguments

x	an object inheriting from class "pdMat" representing a positive-definite matrix.
i, j	optional subscripts applying respectively to the rows and columns of the positive-definite matrix represented by object. When i (j) is omitted, all rows (columns) are extracted.
drop	a logical value. If TRUE, single rows or columns are converted to vectors. If FALSE the returned value retains its matrix representation.
value	a vector, or matrix, with the replacement values for the relevant piece of the matrix represented by x.

Value

if i and j are identical, the returned value will be pdMat object with the same class as x. Otherwise, the returned value will be a matrix. In the case a single row (or column) is selected, the returned value may be converted to a vector, according to the rules above.

Author(s)

José Pinheiro and Douglas Bates <bates@stat.wisc.edu>

See Also

[, pdMat

Examples

```
pd1 <- pdSymm(diag(3))
pd1[1, , drop = FALSE]
pd1[1:2, 1:2] <- 3 * diag(2)
```

Index

∗ **attribute**
groupedData, 125
∗ **datasets**
Alfalfa, 11
Assay, 21
bdf, 26
BodyWeight, 27
Cefamandole, 28
Dialyzer, 73
Earthquake, 77
ergoStool, 78
Fatigue, 79
Gasoline, 93
Glucose, 118
Glucose2, 118
Gun, 129
IGF, 129
Machines, 170
MathAchieve, 170
MathAchSchool, 171
Meat, 174
Milk, 174
Muscle, 176
Nitrendipene, 183
Oats, 197
Orthodont, 198
Ovary, 199
Oxboys, 199
Oxide, 200
PBG, 205
Phenobarb, 225
Pixel, 227
Quinidine, 258
Rail, 260
RatPupWeight, 265
Relaxin, 270
Remifentanil, 270
Soybean, 284
Spruce, 285
Tetracycline1, 295
Tetracycline2, 296
Wafer, 326
Wheat, 326
Wheat2, 327
∗ **data**
balancedGrouped, 25
gapply, 92
isBalanced, 140
∗ **hplot**
plot.lme, 233
∗ **manip**
asTable, 22
groupedData, 125
gsummary, 127
∗ **models**
[.pdMat, 327
ACF, 8
ACF.gls, 9
ACF.lme, 10
allCoef, 12
anova.gls, 13
anova.lme, 15
as.matrix.corStruct, 18
as.matrix.pdMat, 19
as.matrix.reStruct, 20
asOneFormula, 21
augPred, 23
Coef, 29
coef.corStruct, 30
coef.gnls, 31
coef.lme, 32
coef.lmList, 33
coef.modelStruct, 35
coef.pdMat, 36
coef.reStruct, 37
coef.varFunc, 38
collapse, 39
collapse.groupedData, 40

compareFits, 42
comparePred, 43
corAR1, 44
corARMA, 46
corCAR1, 47
corClasses, 49
corCompSymm, 50
corExp, 51
corFactor, 53
corFactor.corStruct, 54
corGaus, 55
corLin, 57
corMatrix, 58
corMatrix.corStruct, 59
corMatrix.pdMat, 61
corMatrix.reStruct, 62
corNatural, 63
corRatio, 64
corSpatial, 66
corSpher, 67
corSymm, 69
Covariate, 71
Covariate.varFunc, 72
Dim, 74
Dim.corSpatial, 74
Dim.corStruct, 75
Dim.pdMat, 77
fdHess, 80
fitted.glsStruct, 81
fitted.gnlsStruct, 82
fitted.lme, 83
fitted.lmeStruct, 84
fitted.lmList, 85
fitted.nlmeStruct, 86
fixed.effects, 87
fixef.lmList, 88
formula.pdBlocked, 89
formula.pdMat, 90
formula.reStruct, 91
getCovariate, 94
getCovariate.corStruct, 95
getCovariate.data.frame, 96
getCovariate.varFunc, 97
getCovariateFormula, 98
getData, 98
getData.gls, 99
getData.lme, 100
getData.lmList, 101
getGroups, 102
getGroups.corStruct, 103
getGroups.data.frame, 104
getGroups.gls, 105
getGroups.lme, 106
getGroups.lmList, 107
getGroups.varFunc, 108
getGroupsFormula, 109
getResponse, 110
getResponseFormula, 110
getVarCov, 111
gls, 112
glsControl, 114
glsObject, 116
glsStruct, 117
gnls, 119
gnlsControl, 121
gnlsObject, 122
gnlsStruct, 124
Initialize, 130
Initialize.corStruct, 131
Initialize.glsStruct, 132
Initialize.lmeStruct, 132
Initialize.reStruct, 133
Initialize.varFunc, 134
intervals, 135
intervals.gls, 136
intervals.lme, 137
intervals.lmList, 139
isInitialized, 141
LDEsysMat, 142
lme, 143
lme.groupedData, 146
lme.lmList, 148
lmeControl, 150
lmeObject, 152
lmeStruct, 154
lmList, 155
lmList.groupedData, 156
logDet, 157
logDet.corStruct, 158
logDet.pdMat, 159
logDet.reStruct, 160
logLik.corStruct, 161
logLik.glsStruct, 162
logLik.gnls, 163
logLik.gnlsStruct, 164
logLik.lme, 165

logLik.lmeStruct, 166
logLik.lmList, 167
logLik.reStruct, 168
logLik.varFunc, 169
Matrix, 171
Matrix.pdMat, 172
Matrix.reStruct, 173
model.matrix.reStruct, 175
Names, 177
Names.formula, 178
Names.pdBlocked, 179
Names.pdMat, 180
Names.reStruct, 181
needUpdate, 182
needUpdate.modelStruct, 182
nlme, 184
nlme.nlsList, 187
nlmeControl, 189
nlmeObject, 191
nlmeStruct, 193
nlsList, 194
nlsList.selfStart, 196
pairs.compareFits, 201
pairs.lme, 202
pairs.lmList, 203
pdBlocked, 205
pdClasses, 207
pdCompSymm, 208
pdConstruct, 209
pdConstruct.pdBlocked, 210
pdDiag, 212
pdFactor, 214
pdFactor.reStruct, 215
pdIdent, 216
pdLogChol, 217
pdMat, 219
pdMatrix, 220
pdMatrix.reStruct, 221
pdNatural, 222
pdSymm, 223
phenoModel, 226
plot.ACF, 227
plot.augPred, 228
plot.compareFits, 229
plot.gls, 230
plot.intervals.lmList, 232
plot.lme, 233
plot.lmList, 235
plot.nffGroupedData, 236
plot.nfnGroupedData, 238
plot.nmGroupedData, 240
plot.ranef.lme, 242
plot.ranef.lmList, 244
plot.Variogram, 245
pooledSD, 246
predict.gls, 247
predict.gnls, 248
predict.lme, 249
predict.lmList, 250
predict.nlme, 252
print.summary.pdMat, 253
print.varFunc, 254
qqnorm.gls, 255
qqnorm.lme, 256
quinModel, 259
random.effects, 261
ranef.lme, 261
ranef.lmList, 263
recalc, 265
recalc.corStruct, 266
recalc.modelStruct, 267
recalc.reStruct, 268
recalc.varFunc, 269
residuals.gls, 272
residuals.glsStruct, 273
residuals.gnlsStruct, 274
residuals.lme, 275
residuals.lmeStruct, 276
residuals.lmList, 277
residuals.nlmeStruct, 278
reStruct, 279
simulate.lme, 281
solve.pdMat, 282
solve.reStruct, 283
splitFormula, 284
summary.corStruct, 286
summary.gls, 287
summary.lme, 288
summary.lmList, 289
summary.modelStruct, 291
summary.nlsList, 292
summary.pdMat, 293
summary.varFunc, 294
update.modelStruct, 296
update.varFunc, 297
varClasses, 298

varComb, 299
varConstPower, 300
varConstProp, 301
VarCorr, 304
varExp, 305
varFixed, 306
varFunc, 307
varIdent, 308
Variogram, 309
Variogram.corExp, 310
Variogram.corGaus, 311
Variogram.corLin, 312
Variogram.corRatio, 313
Variogram.corSpatial, 314
Variogram.corSpher, 315
Variogram.default, 316
Variogram.gls, 317
Variogram.lme, 319
varPower, 322
varWeights, 323
varWeights.glsStruct, 324
varWeights.lmeStruct, 325
[, *328*
[.groupedData *(*groupedData*)*, 125
[.pdBlocked *(*[.pdMat*)*, 327
[.pdMat, 327
[.reStruct *(*reStruct*)*, 279
[<-.pdMat *(*[.pdMat*)*, 327

ACF, 8, *228*
ACF.gls, *8*, 9, *11*
ACF.lme, *8*, *10*, 10, *45*
AIC, *14*, *17*, *287*, *289*
Alfalfa, 11
all.vars, *21*
allCoef, 12
anova.gls, 13
anova.lme, 15
as.data.frame.groupedData *(*groupedData*)*, 125
as.matrix, *172*
as.matrix.corStruct, 18
as.matrix.pdMat, 19, *20*, *61*, *207*, *209*, *212, 213*, *217, 218*, *221*, *223, 224*
as.matrix.reStruct, 20, *62*, *221*
asOneFormula, 21
Assay, 21
asTable, 22, *25*
augPred, 23, *44*, *229*
balancedGrouped, *23*, 25
bdf, 26
BIC, *14*, *17*, *287*, *289*
BodyWeight, 27
bwplot, *232*, *234*, *236*

Cefamandole, 28
class, *304*
Coef, 29
coef, *29*, *42*, *49*, *113*, *144*, *207*, *288*, *298*
coef.corAR1 *(*coef.corStruct*)*, 30
coef.corARMA *(*corARMA*)*, 46
coef.corARMAd *(*coef.corStruct*)*, 30
coef.corCAR1 *(*coef.corStruct*)*, 30
coef.corCompSymm *(*coef.corStruct*)*, 30
coef.corLin *(*coef.corStruct*)*, 30
coef.corNatural *(*coef.corStruct*)*, 30
coef.corSpatial *(*coef.corStruct*)*, 30
coef.corSpher *(*coef.corStruct*)*, 30
coef.corStruct, 30
coef.corSymm *(*coef.corStruct*)*, 30
coef.gnls, 31
coef.lme, 32, *263*
coef.lmList, 33
coef.modelStruct, 35
coef.pdBlocked *(*coef.pdMat*)*, 36
coef.pdCompSymm *(*coef.pdMat*)*, 36
coef.pdDiag *(*coef.pdMat*)*, 36
coef.pdIdent *(*coef.pdMat*)*, 36
coef.pdMat, 36, *38*, *207*, *209*, *212, 213*, *217, 218*, *223, 224*
coef.pdNatural *(*coef.pdMat*)*, 36
coef.pdSymm *(*coef.pdMat*)*, 36
coef.reStruct, 37
coef.summary.nlsList *(*coef.corStruct*)*, 30
coef.varComb, *299*
coef.varComb *(*coef.varFunc*)*, 38
coef.varConstPower, *301*
coef.varConstPower *(*coef.varFunc*)*, 38
coef.varConstProp *(*coef.varFunc*)*, 38
coef.varExp, *306*
coef.varExp *(*coef.varFunc*)*, 38
coef.varFixed *(*coef.varFunc*)*, 38
coef.varFunc, 38, *303*, *308*
coef.varIdent, *309*
coef.varIdent *(*coef.varFunc*)*, 38
coef.varPower, *323*
coef.varPower *(*coef.varFunc*)*, 38

coef<- (Coef), 29
coef<-.corAR1 (coef.corStruct), 30
coef<-.corARMA (coef.corStruct), 30
coef<-.corCAR1 (coef.corStruct), 30
coef<-.corCompSymm (coef.corStruct), 30
coef<-.corLin (coef.corStruct), 30
coef<-.corNatural (coef.corStruct), 30
coef<-.corSpatial (coef.corStruct), 30
coef<-.corSpher (coef.corStruct), 30
coef<-.corStruct (coef.corStruct), 30
coef<-.corSymm (coef.corStruct), 30
coef<-.modelStruct (coef.modelStruct), 35
coef<-.pdBlocked (coef.pdMat), 36
coef<-.pdMat (coef.pdMat), 36
coef<-.reStruct (coef.reStruct), 37
coef<-.varComb (coef.varFunc), 38
coef<-.varConstPower (coef.varFunc), 38
coef<-.varConstProp (coef.varFunc), 38
coef<-.varExp (coef.varFunc), 38
coef<-.varFixed (coef.varFunc), 38
coef<-.varIdent (coef.varFunc), 38
coef<-.varPower (coef.varFunc), 38
coefficients<- (Coef), 29
collapse, 39
collapse.groupedData, *40*, 40, *241*
compareFits, 42, *201*, *230*
comparePred, *42*, 43
contrasts, *176*
corAR1, *31*, 44, *47*, *49*
corARMA, *31*, *45*, 46, *49*
corCAR1, *31*, 47, *49*
corClasses, *19*, *45*, *47, 48*, 49, *50*, *52, 53*, *58*, *105*, *113, 114*, *117*, *119*, *121*, *124*, *144–146*, *148*, *154*, *185, 186*, *188*, *193*, *286*
corCompSymm, *31*, *49*, 50
corExp, *31*, *49*, 51, *67*, *310, 311*
corFactor, 53, *54*, *267*
corFactor.corAR1 (corFactor.corStruct), 54
corFactor.corARMA (corFactor.corStruct), 54
corFactor.corCAR1 (corFactor.corStruct), 54
corFactor.corCompSymm (corFactor.corStruct), 54
corFactor.corNatural (corFactor.corStruct), 54
corFactor.corSpatial (corFactor.corStruct), 54
corFactor.corStruct, *54*, 54, *60*
corFactor.corSymm (corFactor.corStruct), 54
corGaus, *31*, *49*, 55, *67*, *311, 312*
corLin, *31*, *49*, 57, *67*, *312, 313*
corMatrix, *19, 20*, *49*, 58, *62*, *221*
corMatrix.corAR1 (corMatrix.corStruct), 59
corMatrix.corARMA (corMatrix.corStruct), 59
corMatrix.corCAR1 (corMatrix.corStruct), 59
corMatrix.corCompSymm (corMatrix.corStruct), 59
corMatrix.corNatural (corMatrix.corStruct), 59
corMatrix.corSpatial (corMatrix.corStruct), 59
corMatrix.corStruct, *54*, *59*, 59, *158*
corMatrix.corSymm (corMatrix.corStruct), 59
corMatrix.pdBlocked (corMatrix.pdMat), 61
corMatrix.pdCompSymm (corMatrix.pdMat), 61
corMatrix.pdDiag (corMatrix.pdMat), 61
corMatrix.pdIdent (corMatrix.pdMat), 61
corMatrix.pdMat, *59*, 61
corMatrix.pdSymm (corMatrix.pdMat), 61
corMatrix.reStruct, *59*, 62
corNatural, 63, *286*
corRatio, *31*, *49*, 64, *67*, *313, 314*
corSpatial, *31*, 66, *74, 75*, *314, 315*
corSpher, *31*, *49*, *67*, 67, *315, 316*
corStruct, *18*, *29, 30*, *35*, *54*, *59*, *74*, *76*, *113*, *131*, *144*, *158*, *161*, *266*, *286*
corStruct (corClasses), 49
corSymm, *31*, *49*, 69
Covariate, 71
Covariate.varFunc, 72
covariate<- (Covariate), 71
covariate<-.varFunc (Covariate.varFunc), 72

Dialyzer, 73
Dim, 74, *75–77*

Dim.corSpatial, *45*, *74*, 74, *76*
Dim.corStruct, *74, 75*, 75, *131*
Dim.pdCompSymm *(*Dim.pdMat*)*, 77
Dim.pdDiag *(*Dim.pdMat*)*, 77
Dim.pdIdent *(*Dim.pdMat*)*, 77
Dim.pdMat, *74*, 77
Dim.pdNatural *(*Dim.pdMat*)*, 77
Dim.pdSymm *(*Dim.pdMat*)*, 77
dist, *52*, *56*, *58*, *65*, *67*, *69*
dotplot, *230*, *232, 233*, *238*, *242, 243*, *245*

Earthquake, 77
ergoStool, 78

factor, *271*
Fatigue, 79
fdHess, 80
fitted, *113*, *144*
fitted.glsStruct, 81, *273*
fitted.gnlsStruct, 82, *274*
fitted.lme, 83, *85*, *250*, *253*, *276*
fitted.lmeStruct, 84, *277*
fitted.lmList, 85, *278*
fitted.nlmeStruct, 86, *279*
fixed.effects, 87, *144*
fixed.effects.lmList, *35*, *264*
fixed.effects.lmList *(*fixef.lmList*)*, 88
fixef *(*fixed.effects*)*, 87
fixef.lmList, *87*, 88
formula, *21*, *91*, *127*, *153*, *178*, *285*
formula.pdBlocked, 89
formula.pdMat, 90
formula.reStruct, 91, *176*
function, *232*, *243*

gapply, 92, *127*
Gasoline, 93
getCovariate, *71*, 94, *95*, *98*
getCovariate.corSpatial
*(*getCovariate.corStruct*)*, 95
getCovariate.corStruct, *94*, 95
getCovariate.data.frame, *94*, 96
getCovariate.varFunc, *72*, *94*, 97
getCovariateFormula, *94*, *96*, 98
getData, 98, *99–101*
getData.gls, *99*, 99
getData.gnls *(*getData.gls*)*, 99
getData.lme, *99*, 100
getData.lmList, *99*, 101
getData.nlme *(*getData.lme*)*, 100
getData.nls *(*getData.lme*)*, 100
getGroups, *24*, *44*, 102, *103*, *109*, *128*
getGroups.corStruct, *102*, 103
getGroups.data.frame, *102*, 104
getGroups.gls, *102*, 105
getGroups.lme, *102*, 106
getGroups.lmList, *102*, 107
getGroups.varFunc, *102*, 108
getGroupsFormula, *102*, *104*, 109
getGroupsFormula.gls, *109*
getGroupsFormula.lme, *109*
getGroupsFormula.lmList, *109*
getGroupsFormula.reStruct, *109*
getOption, *195*
getResponse, 110, *111*
getResponseFormula, *110*, 110
getVarCov, 111
gls, *9*, *13, 14*, *17*, *81*, *99*, *105*, *111, 112*, 112, *115–117*, *132*, *136, 137*, *162*, *165*, *231, 232*, *247, 248*, *255, 256*, *272, 273*, *287*, *317*, *319*
glsControl, *113, 114*, 114
glsObject, *113, 114*, 116, *287*
glsStruct, *81*, *114*, *117*, 117, *132*, *162*, *273*, *324*
Glucose, 118
Glucose2, 118
gnls, *14*, *17*, *31*, *82*, 119, *122–124*, *163, 164*, *248, 249*, *274*
gnlsControl, *121*, 121
gnlsObject, *121*, 122
gnlsStruct, *82*, *121*, 124, *164*, *274*
groupedData, *23*, *25*, *41*, 125, *128*, *140*, *143, 144*, *146*, *148*, *157*, *196, 197*, *238, 239*, *241*, *270*, *280*
gsummary, *33*, *35*, *92*, *127*, 127, *263*
Gun, 129

histogram, *232*, *234*, *236*

IGF, 129
Initialize, *31*, *36*, 130, *132–135*, *141*, *298*
Initialize.corAR1
*(*Initialize.corStruct*)*, 131
Initialize.corARMA
*(*Initialize.corStruct*)*, 131
Initialize.corCAR1
*(*Initialize.corStruct*)*, 131

Initialize.corCompSymm (Initialize.corStruct), 131
Initialize.corLin (Initialize.corStruct), 131
Initialize.corNatural, *63*
Initialize.corNatural (Initialize.corStruct), 131
Initialize.corSpatial (Initialize.corStruct), 131
Initialize.corSpher (Initialize.corStruct), 131
Initialize.corStruct, *45*, *47*, *48*, *50*, *52*, *54*, *56*, *58*, *60*, *65*, *67*, *69*, *131*, 131, *132*, *133*, *286*
Initialize.corSymm, *70*
Initialize.corSymm (Initialize.corStruct), 131
Initialize.glsStruct, *131*, 132
Initialize.gnlsStruct (gnlsStruct), 124
Initialize.lmeStruct, *131*, 132
Initialize.reStruct, *131*, *133*, 133
Initialize.varComb (Initialize.varFunc), 134
Initialize.varConstPower (Initialize.varFunc), 134
Initialize.varConstProp (Initialize.varFunc), 134
Initialize.varExp (Initialize.varFunc), 134
Initialize.varFixed (Initialize.varFunc), 134
Initialize.varFunc, *131–133*, 134
Initialize.varIdent (Initialize.varFunc), 134
Initialize.varPower (Initialize.varFunc), 134
intervals, 135, *137–139*
intervals.gls, *136*, 136
intervals.lme, *136*, 137
intervals.lmList, *136*, 139, *232*, *233*
isBalanced, *23*, *25*, 140
isInitialized, *131*, 141

LDEsysMat, 142
list, *150*
lm, *156*, *157*, *247*
lme, *10*, *14*, *16*, *17*, *32*, *33*, *83–85*, *100*, *106*, *111*, *112*, *127*, *133*, *138*, 143, *148*, *150*, *152–154*, *165*, *166*, *186*, *202*, *203*, *233*, *234*, *242*, *243*, *249*, *250*, *257*, *258*, *262*, *263*, *268*, *275–277*, *280–282*, *288*, *289*, *304*, *305*, *320*, *321*
lme.formula, *146*, *148*
lme.groupedData, *143*, *145*, 146
lme.lmList, *143*, *145*, 148, *156*, *157*
lmeControl, *144*, *145*, 150
lmeObject, *144*, *145*, *148*, *150*, 152, *288*
lmeStruct, *13*, *84*, *133*, *145*, 154, *166*, *276*, *325*
lmList, *34*, *35*, *85*, *88*, *101*, *107*, *139*, *143*, *145*, *148*, *150*, 155, *157*, *167*, *189*, *203*, *204*, *233*, *235*, *236*, *245*, *247*, *251*, *257*, *263*, *264*, *277*, *278*, *289*, *290*
lmList.formula, *156*, *157*
lmList.groupedData, *155*, 156
loess, *246*
logDet, 157, *158–160*
logDet.corStruct, *158*, 158, *161*
logDet.pdBlocked (logDet.pdMat), 159
logDet.pdCompSymm (logDet.pdMat), 159
logDet.pdDiag (logDet.pdMat), 159
logDet.pdIdent (logDet.pdMat), 159
logDet.pdMat, *158*, 159
logDet.pdNatural (logDet.pdMat), 159
logDet.pdSymm (logDet.pdMat), 159
logDet.reStruct, *158*, 160
logical, *151*, *156*, *157*, *195*, *196*
logLik, *158*, *268*
logLik.corStruct, *158*, 161, *165*, *267*
logLik.gls, *14*
logLik.gls (logLik.lme), 165
logLik.glsStruct, 162, *165*
logLik.gnls, 163, *164*
logLik.gnlsStruct, 164
logLik.lme, *17*, *161–163*, 165, *166–169*
logLik.lmeStruct, *165*, 166
logLik.lmeStructInt (logLik.lmeStruct), 166
logLik.lmList, *165*, 167
logLik.reStruct, *165*, 168
logLik.varComb (logLik.varFunc), 169
logLik.varFunc, *165*, 169, *269*, *323*

Machines, 170
MathAchieve, 170
MathAchSchool, 171

Matrix, 171
Matrix.pdMat, 172
Matrix.reStruct, 173
matrix<- (Matrix), 171
matrix<-.pdBlocked (Matrix.pdMat), 172
matrix<-.pdMat (Matrix.pdMat), 172
matrix<-.reStruct (Matrix.reStruct), 173
Meat, 174
Milk, 174
model.matrix, *176*, *178*
model.matrix.default, *144*
model.matrix.reStruct, 175
Muscle, 176

na.fail, *113*, *144*
Names, 177, *178–180*
Names.formula, *177*, 178
Names.listForm (Names.formula), 178
Names.pdBlocked, *177*, 179, *180*
Names.pdMat, *177*, *179*, 180, *181*
Names.reStruct, *177*, 181
Names<- (Names), 177
Names<-.pdBlocked (Names.pdBlocked), 179
Names<-.pdMat (Names.pdMat), 180
Names<-.reStruct (Names.reStruct), 181
napredict, *83*
naresid, *275*
needUpdate, 182, *183*, *298*
needUpdate.corStruct
(needUpdate.modelStruct), 182
needUpdate.modelStruct, *182*, 182
needUpdate.reStruct
(needUpdate.modelStruct), 182
Nitrendipene, 183
nlm, *190, 191*, *281*
nlme, *14*, *17*, *87*, 184, *189*, *191–193*, *252, 253*, *279*, *304, 305*
nlme.formula, *187*
nlme.nlsList, *184*, *186*, 187, *195*, *197*
nlmeControl, *186*, 189
nlmeObject, *186*, *189*, 191
nlmeStruct, *13*, *86*, *186*, *191*, 193, *278*
nlminb, *115*, *122*, *151, 152*, *190*, *281*
nls, *195*, *197*
nlsList, *186, 187*, 194, *196, 197*, *292, 293*
nlsList.formula, *197*
nlsList.selfStart, *194, 195*, 196
numeric, *270*

Oats, 197
offset, *113*, *144*
optim, *115*, *122*, *151, 152*, *191*
ordered, *270*
Orthodont, 198
Ovary, 199
Oxboys, 199
Oxide, 200

pairs.compareFits, *42*, 201, *203, 204*, *230*
pairs.lme, *201*, 202, *204*
pairs.lmList, *201*, *203*, 203
PBG, 205
pdBlocked, *89*, *179*, 205, *208*, *212*
pdClasses, *145*, *185, 186*, *207*, 207, *209*, *212*, *213*, *217, 218*, *220, 221*, *223, 224*, *281*
pdCompSymm, *208*, 208, *210*, *220*
pdConstruct, *207*, 209, *212*
pdConstruct.pdBlocked, 210
pdDiag, *208*, *210*, 212, *220*
pdFactor, *208*, 214, *215*, *221*
pdFactor.pdMat, *215*
pdFactor.reStruct, 215
pdIdent, *208*, *210*, 216, *220*
pdLogChol, *208*, 217
pdMat, *19, 20*, *29*, *36–38*, *61, 62*, *77*, *89, 90*, *134*, *143, 144*, *159, 160*, *168*, *172, 173*, *180, 181*, *208*, 219, *221*, *254*, *280–283*, *293, 294*, *328*
pdMatrix, *61*, *207, 208*, *214*, 220, *221*
pdMatrix.pdMat, *221*
pdMatrix.reStruct, *215*, *221*, 221
pdNatural, *63*, *138*, *208*, *210*, *220*, 222
pdSymm, *208*, *210*, *220*, 223
Phenobarb, 225, *226*
phenoModel, 226
Pixel, 227
plot.ACF, *8*, *10, 11*, 227
plot.augPred, *24*, 228
plot.compareFits, *42*, *201*, 229
plot.gls, *114*, 230, *256*
plot.intervals.lmList, *139*, 232
plot.lme, *145*, 233, *258*
plot.lmList, *156*, 235
plot.nffGroupedData, *127*, 236, *241*
plot.nfnGroupedData, *127*, 238, *241*
plot.nls (plot.lme), 233
plot.nmGroupedData, *41*, *127*, 240

plot.pdMat (pdMat), 219
plot.ranef.lme, *33*, 242, *263*
plot.ranef.lmList, *35*, 244
plot.simulate.lme (simulate.lme), 281
plot.Variogram, 245, *310–317*, *319*, *321*
pooledSD, *156*, 246
predict, *24*
predict.gls, *114*, 247
predict.gnls, *121*, 248
predict.lm, *236*, *251*
predict.lme, *145*, 249
predict.lmList, *156*, 250
predict.nlme, 252
print, *288*
print.anova.lme, *14*, *17*
print.anova.lme (anova.lme), 15
print.compareFits (compareFits), 42
print.corNatural (corNatural), 63
print.intervals.gls, *137*
print.intervals.gls (intervals.gls), 136
print.intervals.lme, *138*
print.intervals.lme (intervals.lme), 137
print.intervals.lmList (intervals.lmList), 139
print.lmList (lmList), 155
print.ranef.lme (ranef.lme), 261
print.reStruct (reStruct), 279
print.simulate.lme (simulate.lme), 281
print.summary.lme (summary.lme), 288
print.summary.pdMat, 253, *294*
print.varComb (print.varFunc), 254
print.VarCorr.lme (VarCorr), 304
print.VarCov (getVarCov), 111
print.varFunc, 254

qqnorm.gls, *114*, 255
qqnorm.lm (qqnorm.lme), 256
qqnorm.lme, *145*, 256
qqnorm.lmList (qqnorm.lme), 256
qqnorm.nls (qqnorm.lme), 256
Quinidine, 258, *260*
quinModel, 259

Rail, 260
random.effects, *42*, *144*, 261, *263*, *264*
random.effects.lme (ranef.lme), 261
random.effects.lmList, *88*
random.effects.lmList (ranef.lmList), 263
ranef, *242*
ranef (random.effects), 261
ranef.lme, *33*, *242*, *243*, *261*, 261
ranef.lmList, *35*, *244*, *261*, 263
RatPupWeight, 265
recalc, 265, *268*, *269*
recalc.corAR1 (recalc.corStruct), 266
recalc.corARMA (recalc.corStruct), 266
recalc.corCAR1 (recalc.corStruct), 266
recalc.corCompSymm (recalc.corStruct), 266
recalc.corNatural (recalc.corStruct), 266
recalc.corSpatial (recalc.corStruct), 266
recalc.corStruct, *54*, *266*, 266, *267*
recalc.corSymm (recalc.corStruct), 266
recalc.modelStruct, *266*, 267
recalc.reStruct, *266*, *267*, 268
recalc.varFunc, *266*, *267*, 269
recalc.varIdent (recalc.varFunc), 269
Relaxin, 270
Remifentanil, 270
resid, *113*, *144*
residuals.gls, *114*, 272, *273*
residuals.glsStruct, *81*, *117*, 273
residuals.gnls, *274*
residuals.gnls (residuals.gls), 272
residuals.gnlsStruct, *82*, *124*, 274
residuals.lme, *84*, *145*, 275, *277*
residuals.lmeStruct, *85*, *154*, 276
residuals.lmList, *85*, *156*, 277
residuals.nlmeStruct, *87*, *193*, 278
reStruct, *20*, *29*, *37*, *38*, *62*, *91*, *127*, *134*, *144*, *145*, *154*, *160*, *168*, *173*, *175*, *176*, *181*, *186*, *193*, *215*, *220*, *221*, *268*, 279, *283*, *291*, *297*

selfStart, *119*, *196*, *197*
set.seed, *282*
simulate.lme, *145*, 281
solve.pdBlocked (solve.pdMat), 282
solve.pdDiag (solve.pdMat), 282
solve.pdIdent (solve.pdMat), 282
solve.pdLogChol (solve.pdMat), 282
solve.pdMat, *220*, 282, *283*
solve.pdNatural (solve.pdMat), 282
solve.pdSymm (solve.pdMat), 282
solve.reStruct, *280*, 283

Soybean, 284
splitFormula, 284
splom, *201*, *203*, *204*
Spruce, 285
stop, *122*, *151*
summary, *128*, *254*, *286*, *287*, *290*, *291*, *293*
summary.corAR1 *(*summary.corStruct*)*, 286
summary.corARMA *(*summary.corStruct*)*, 286
summary.corCAR1 *(*summary.corStruct*)*, 286
summary.corCompSymm *(*summary.corStruct*)*, 286
summary.corExp *(*summary.corStruct*)*, 286
summary.corGaus *(*summary.corStruct*)*, 286
summary.corLin *(*summary.corStruct*)*, 286
summary.corNatural, *63*
summary.corNatural *(*summary.corStruct*)*, 286
summary.corRatio *(*summary.corStruct*)*, 286
summary.corSpher *(*summary.corStruct*)*, 286
summary.corStruct, *45*, *47–50*, *52*, *56*, *58*, *65*, *67*, *69*, 286
summary.corSymm, *70*
summary.corSymm *(*summary.corStruct*)*, 286
summary.gls, *114*, 287
summary.lme, *145*, 288
summary.lmList, *156*, 289
summary.modelStruct, 291
summary.nlsList, *195*, 292
summary.pdBlocked *(*summary.pdMat*)*, 293
summary.pdCompSymm *(*summary.pdMat*)*, 293
summary.pdDiag *(*summary.pdMat*)*, 293
summary.pdIdent *(*summary.pdMat*)*, 293
summary.pdLogChol *(*summary.pdMat*)*, 293
summary.pdMat, *220*, *254*, 293
summary.pdNatural *(*summary.pdMat*)*, 293
summary.pdSymm *(*summary.pdMat*)*, 293
summary.reStruct, *280*
summary.reStruct *(*summary.modelStruct*)*, 291
summary.varComb *(*summary.varFunc*)*, 294
summary.varConstPower *(*summary.varFunc*)*, 294
summary.varConstProp *(*summary.varFunc*)*, 294
summary.varExp *(*summary.varFunc*)*, 294
summary.varFixed *(*summary.varFunc*)*, 294
summary.varFunc, *255*, 294, *298*, *308*
summary.varIdent *(*summary.varFunc*)*, 294
summary.varPower *(*summary.varFunc*)*, 294

table, *140*
terms, *153*, *178*
terms.object, *153*
Tetracycline1, 295
Tetracycline2, 296
try, *195*
tryCatch, *156*, *195*

update.corStruct *(*update.modelStruct*)*, 296
update.formula, *112*, *143*, *194*
update.gls *(*gls*)*, 112
update.groupedData *(*groupedData*)*, 125
update.lme *(*lme*)*, 143
update.lmList *(*lmList*)*, 155
update.modelStruct, 296
update.nlsList *(*nlsList*)*, 194
update.reStruct, *280*
update.reStruct *(*update.modelStruct*)*, 296
update.varComb *(*update.varFunc*)*, 297
update.varConstPower *(*update.varFunc*)*, 297
update.varConstProp *(*update.varFunc*)*, 297
update.varExp *(*update.varFunc*)*, 297
update.varExpon *(*update.varFunc*)*, 297
update.varFunc, 297
update.varPower *(*update.varFunc*)*, 297

varClasses, *71*, *113*, *114*, *120*, *121*, *144–146*, *149*, *185*, *186*, *188*, *295*, 298, *299*, *301*, *303*, *306*, *307*, *309*
varComb, *298*, 299
varConstPower, *298*, 300, *302*
varConstProp, *298*, 301
VarCorr, 304
varExp, *298*, 305
varFixed, *113*, *144*, *298*, 306, *307*, *308*
varFunc, *29*, *35*, *38*, *39*, *71*, *72*, *81*, *113*, *114*, *117*, *121*, *124*, *135*, *144*, *145*, *154*, *162*, *169*, *186*, *193*, *254*, *269*, *273*, *294*, *295*, *297*, *302*, *307*, 307, *324*
varIdent, *298*, 308
Variogram, *245*, *246*, 309, *311–317*, *319*, *321*

Variogram.corExp, *310*, 310, *315*
Variogram.corGaus, *310*, 311, *315*
Variogram.corLin, *310*, 312, *315*
Variogram.corRatio, *310*, 313, *315*
Variogram.corSpatial, *310*, 314
Variogram.corSpher, *310*, *315*, 315
Variogram.default, *310*, *315*, 316, *319*, *321*
Variogram.gls, *310*, *317*, 317, *321*
Variogram.lme, *310*, *317*, *319*, 319
varPower, *298*, 322
varWeights, *269*, *323*, 323, *324, 325*
varWeights.glsStruct, 324
varWeights.lmeStruct, 325
varWeights.varComb, *299*
varWeights.varFunc, *301*, *303*, *306–309*, *323*

Wafer, 326
warning, *122*, *151*, *156*, *190*, *195*
Wheat, 326
Wheat2, 327

xyplot, *201*, *203, 204*, *228, 229*, *232*, *234*, *236*, *239*, *242, 243*, *246*