Package 'easy.glmnet'

October 26, 2026

Type Package

Title Functions to Simplify the Use of 'glmnet' for Machine Learning

Version 1.2

Date 2026-09-25

Description Provides several functions to simplify using the 'glmnet' package: a) converting data frames into matrices ready for 'glmnet'; b) imputing missing variables multiple times; c) fitting and applying prediction models straightforwardly; d) assigning observations to folds in a balanced way; e) cross-validate the models and estimating the accuracy and calibration; f) selecting the most representative model across imputations and folds; and g) getting the count and relevance of the model regressors; as described in several publications: Solanes et al. (2022) <doi:10.1038/s41537-022-00309-w>, Palau et al. (2023) <doi:10.1016/j.rpsm.2023.01.001>, Salazar de Pablo et al. (2025) <doi:10.1038/s41380-025-03244-1>.

License GPL-3

Imports cocor, doParallel, foreach, glmnet, multisite.accuracy, parallel, pROC, survival

Suggests knitr, rmarkdown

VignetteBuilder knitr

LazyData true

Depends R (>= 3.5)

NeedsCompilation no

Author Joaquim Radua [aut, cre] (ORCID: <https://orcid.org/0000-0003-1240-5438>)

Maintainer Joaquim Radua <quimradua@gmail.com>

Repository CRAN

Date/Publication 2026-10-26 11:00:02 UTC

Contents

assign.folds 2
cv 3

data.frame2glmnet.matrix 6
glmnet_fit 7
glmnet_get.items.relevance 9
glmnet_get.main.model 11
impute.glmnet.matrix_fit 12
patients 15
simple.accuracy 16
simple.accuracy.test 18
simple.calibration 21
surv2binary 23

Index **25**

`assign.folds` *Assign observations to folds in a balanced way*

Description

Function to assign observations to folds, ensuring a similar distribution across folds (and sites).

Usage

```
assign.folds(y, family = c("binomial", "cox", "gaussian"), site = NULL, nfolds = 10)
```

Arguments

`y`	response to be predicted. A binary vector for `"binomial"`, an object of class `"Surv"` for `"cox"`, or a numeric vector for `"gaussian"`.
`family`	distribution of `y`: `"binomial"`, `"cox"`, or `"gaussian"`.
`site`	vector or factor with the sites' names, or NULL for studies conducted in a single site.
`nfolds`	number of folds.

Details

If `family` is `"binomial"`, the function randomly assigns the folds separately for the two outcomes. If `family` is `"gaussian"`, the function randomly assigns the folds separately for ranges of the outcome. If `family` is `"cox"`, the function randomly assigns the folds separately for ranges of time and censorship. If `site` is not null, the function randomly assigns the folds separately for each site.

Value

A numeric vector with the fold assigned to each observation

Author(s)

Joaquim Radua and Aleix Solanes

References

Solanes, A., Mezquida, G., Janssen, J., Amoretti, S., Lobo, A., Gonzalez-Pinto, A., Arango, C., Vieta, E., Castro-Fornieles, J., Berge, D., Albacete, A., Gine, E., Parellada, M., Bernardo, M.; PEPs group (collaborators); Pomarol-Clotet, E., Radua, J. (2022) Combining MRI and clinical data to detect high relapse risk after the first episode of psychosis. *Schizophrenia*, **8**, 100, doi:10.1038/s41537-022-00309-w.

See Also

cv for conducting a cross-validation; `vignette("easy.glmnet")` for a tutorial on cross-validation; `vignette("other-outcomes-and-sites")` for a tutorial including multisite data.

Examples

```
# Create random y (numeric)
y = rnorm(200, sample(c(1, 10), 200, replace = TRUE))

# Assign folds
fold = assign.folds(y, "gaussian", nfolds = 4)

# Check that the distribution of y is similar across folds
oldpar = par(mfrow = c(2, 2))
for (i in 1:4) {
  hist(y[which(fold == i)], main = paste("Fold", i), xlab = "y")
}
par(oldpar)
```

---

cv *Conduct cross-validation*

Description

Function to easily cross-validate (including fold assignation, merging fold outputs, etc).

Usage

```
cv(x, y, family = c("binomial", "cox", "gaussian"), fit_fun, predict_fun, site = NULL,
covar = NULL, nfolds = 10, pred.format = NA, verbose = TRUE, ...)
## S3 method for class 'cv'
print(x, ...)
```

Arguments

x	predictors. A matrix or data.frame (rows are observations and columns are variables) or a vector of factor (if only one predictor). For `print`, an object of class `"cv"`, as obtained with cv.
y	response to be predicted. A binary vector for "binomial", a "Surv" object for "cox", or a numeric vector for "gaussian".

family	distribution of y: "binomial", "cox", or "gaussian".
fit_fun	function to create the prediction model using the training subsets. It can have between two and four arguments (the first two are compulsory, the third one if site is not null, and the four if covar is not null): x_training (training X data.frame), y_training (training Y outcomes), site_training (training site names), and covar_training (training covariates). It must return the overall prediction model, which may be a list of the different submodels used in different steps and/or derived from different imputations.
predict_fun	function to apply the prediction model to the test sets. It can have between two and four arguments (the first two are compulsory, the third one if site is not null, and the four if covar is not null): model (the overall prediction model), x_test (test X data.frame), site_test (test site names), and covar_test (test covariates). It must return the predictions.
site	vector or factor with the sites' names, or NULL for studies conducted in a single site.
covar	other covariates that can be passed to fit_fun and predict_fun. A matrix or data.frame (rows are observations and columns are variables) or a vector of factor (if only one covariate).
nfolds	number of folds.
pred.format	format of the predictions returned by each fold. E.g., if the prediction is an array, use NA.
verbose	(optional) logical, whether to print some messages during execution.
...	other arguments that can be passed to fit_fun and predict_fun functions (ignored by print).

Details

This function iteratively divides the dataset into a training dataset, with which fits the model using the function fit_fun, and a test dataset, to which applies the model using the function predict_fun. It saves the models fit with the training datasets and the predictions obtained in the test datasets. The folds are assigned automatically using assign.folds, accounting for the site if this is not null.

In multisite studies, it is convenient to remove the differences between sites in the predictors within fit_fun and predict_fun, e.g., estimating them in the training set with combat_fit and removing them from the training and test sets with combat_apply, both from the package **combat.enigma** (Radua et al., 2020). A benchmark study found that ComBat is enough to remove the effects of the site for lasso models, but not for other techniques such as random forests, support vector machines, or Gaussian processes (Solanes et al., 2023; https://www.imardgroup.com/mareos-benchmark/benchmark.html). See the vignette vignette("other-outcomes-and-sites") for an example.

Value

An object of class "cv", which is a list with the family, the models fit in each fold, and the predictions, a data.frame with the fold (fold), the site (site, if not null), the actual outcome (y), and the predictions (y.pred, or y1.pred, y2.pred, etc., if the predictions have several columns) of each observation, in the original order of the observations.

Printing this object shows the accuracy and calibration of the predictions, estimated with `simple.accuracy` and `simple.calibration`, or, if `site` is not null, the accuracy estimated with `multisite.accuracy` using its default parameters. For binomial outcomes, it shows the balanced accuracy if the predictions are binary (e.g., rounded probabilities), or the area under the ROC curve and the calibration otherwise.

Author(s)

Joaquim Radua

References

Radua, J., Koutsouleris, N. (2024) Ten simple rules for using machine learning in mental health research. *Biological Psychiatry*, **96**, 511–513, doi:10.1016/j.biopsych.2023.11.012.

Radua, J., Vieta, E., Shinohara, R., Kochunov, P., Quide, Y., Green, M.J., et al. (2020) Increased power by harmonizing structural MRI site differences with the ComBat batch adjustment method in ENIGMA. *NeuroImage*, **218**, 116956, doi:10.1016/j.neuroimage.2020.116956.

Solanes, A., Gosling, C.J., Fortea, L., Ortuno, M., Lopez-Soley, E., Llufriu, S., et al. (2023) Removing the effects of the site in brain imaging machine-learning - Measurement and extendable benchmark. *NeuroImage*, **265**, 119800, doi:10.1016/j.neuroimage.2022.119800.

See Also

`glmnet_predict` for obtaining predictions, `simple.accuracy` for estimating accuracy, `multisite.accuracy` for estimating accuracy in multisite machine-learning models, `simple.calibration` for estimating calibration; `vignette("easy.glmnet")` for a tutorial on cross-validation.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
  list(
    lasso = glmnet_fit(x_training, y_training, family = "binomial")
  )
}
predict_fun = function (m, x_test) {
  glmnet_predict(m$lasso, x_test)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show accuracy and calibration
res

# Balanced accuracy, sensitivity, and specificity
simple.accuracy(res$predictions$y, round(res$predictions$y.pred), "bac")
```

`data.frame2glmnet.matrix` *Convert a data.frame into a matrix ready for glmnet*

Description

Function to convert categorical variables into dummy variables ready for `glmnet_fit` and `glmnet_predict`. Additionally, it also removes constant columns.

Usage

```
data.frame2glmnet.matrix_fit(x)
data.frame2glmnet.matrix(m, x)
```

Arguments

`m`	model to conduct the conversion, obtained with `data.frame2glmnet.matrix_fit`.
`x`	data.frame to be converted.

Details

Note that the returned matrix might differ from the design matrix of a linear model because for categoric variables with more than two levels, it creates as many dummy variables as levels (which is ok for lasso).

Value

A matrix ready for `glmnet_fit` and `glmnet_predict`.

Author(s)

Joaquim Radua and Aleix Solanes

See Also

`glmnet_predict` for obtaining predictions, `cv` for conducting a cross-validation; `vignette("easy.glmnet")` for a tutorial including categorical variables.

Examples

```
# Create random x (predictors) and y (binary)
x = cbind(
  as.data.frame(matrix(rnorm(10000), ncol = 20)),
  matrix(sample(letters, 2500, TRUE), ncol = 5)
)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation, including conversion to matrix
```

```
fit_fun = function (x_training, y_training) {
  m = list(
    matrix = data.frame2glmnet.matrix_fit(x_training)
  )
  x_mat = data.frame2glmnet.matrix(m$matrix, x_training)
  m$lasso = glmnet_fit(x_mat, y_training, family = "binomial")
  m
}
predict_fun = function (m, x_test) {
  x_mat = data.frame2glmnet.matrix(m$matrix, x_test)
  glmnet_predict(m$lasso, x_mat)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show accuracy and calibration
res

# Balanced accuracy, sensitivity, and specificity
simple.accuracy(res$predictions$y, round(res$predictions$y.pred), "bac")
```

`glmnet_fit` *Obtain and use a glmnet prediction model*

Description

Function to easily fit and apply glmnet models (including best lambda estimation, etc).

Usage

```
glmnet_fit(x, y, family = c("binomial", "cox", "gaussian"),
           type.measure = "default", nfolds = 10, min.beta = 1e-12,
           cox.a0 = TRUE, ...)
glmnet_predict(m, x)
## S3 method for class 'glmnet_fit'
coef(object, ...)
## S3 method for class 'glmnet_fit'
predict(object, newdata, ...)
## S3 method for class 'glmnet_fit'
print(x, ...)
```

Arguments

`x`	input matrix of dimension nobs x nvars; each row is an observation vector. It can be easily obtained with `data.frame2glmnet.matrix`. For `print`, an object of class `"glmnet_fit"`.
`y`	response to be predicted. A binary vector for "binomial", a "Surv" object for "cox", or a numeric vector for "gaussian".

family	distribution of y: "binomial", "cox", or "gaussian".
m, object	lasso model to conduct the prediction, obtained with glmnet_fit.
newdata	input matrix to which the model is applied, as x in glmnet_predict.
type.measure	loss used in the internal cross-validation to select lambda, passed to cv.glmnet. "default" corresponds, in cv.glmnet, to "deviance" for "binomial" and "cox", and "mse" for "gaussian". Other options may be "class", "auc", "mse", or "mae" for "binomial"; "C" for "cox"; and "mae" for "gaussian". Note that, before version 1.2, the type.measure for "binomial" was "class"; use type.measure = "class" to reproduce the results of previous versions.
nfolds	number of folds.
min.beta	minimum value of betas.
cox.a0	logical, if TRUE estimates an artificial intercept for "cox" to keep the mean of linear predictors constant across folds. Note that, before version 1.2, this intercept was not estimated; use cox.a0 = FALSE to reproduce the results of previous versions.
...	further arguments passed to the function glmnet, such as standardize (a logical flag for x variable standardization, though the coefficients are always returned on the original scale). Ignored by coef, predict, and print.

Details

The function glmnet_fit mainly calls the function glmnet to fit a generalized linear model with lasso regularization, though with some extra code to make the call easier: it allow x to have a single column, it conducts an internal cross-validation using the function cv.glmnet to select the regularization parameter lambda automatically, and it removes the negligible coefficients.

Value

glmnet_fit returns an object of class "glmnet_fit", which is briefly a list with the intercept ("a0"; for "cox", the artificial intercept if cox.a0 is TRUE, or NULL otherwise) and regressors ("beta") of the model; it also includes the indices ("i") and standard deviations ("sd") of the regressors and the "family" of the response.

glmnet_predict and predict return a numeric vector with the predictions: probabilities for "binomial", linear predictors for "cox", and predicted values for "gaussian". coef returns the intercept (except for "cox") and the coefficients, and print shows them.

Author(s)

Joaquim Radua and Aleix Solanes

References

Solanes, A., Mezquida, G., Janssen, J., Amoretti, S., Lobo, A., Gonzalez-Pinto, A., Arango, C., Vieta, E., Castro-Fornieles, J., Berge, D., Albacete, A., Gine, E., Parellada, M., Bernardo, M.; PEPs group (collaborators); Pomarol-Clotet, E., Radua, J. (2022) Combining MRI and clinical data to detect high relapse risk after the first episode of psychosis. *Schizophrenia*, **8**, 100, doi:10.1038/s41537-022-00309-w.

Palau, P., Solanes, A., Madre, M., Saez-Francas, N., Sarro, S., Moro, N., Verdolini, N., Sanchez, M., Alonso-Lana, S., Amann, B.L., Romaguera, A., Martin-Subero, M., Fortea, L., Fuentes-Claramonte, P., Garcia-Leon, M.A., Munuera, J., Canales-Rodriguez, E.J., Fernandez-Corcuera, P., Brambilla, P., Vieta, E., Pomarol-Clotet, E., Radua, J. (2023) Improved estimation of the risk of manic relapse by combining clinical and brain scan data. *Spanish Journal of Psychiatry and Mental Health*, **16**, 235–243, doi:10.1016/j.rpsm.2023.01.001.

See Also

cv for conducting a cross-validation; `vignette("easy.glmnet")` for a tutorial; `vignette("other-outcomes-and-sites")` for a tutorial on continuous and survival outcomes.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
  list(
    lasso = glmnet_fit(x_training, y_training, family = "binomial")
  )
}
predict_fun = function (m, x_test) {
  glmnet_predict(m$lasso, x_test)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show accuracy and calibration
res

# Balanced accuracy, sensitivity, and specificity
simple.accuracy(res$predictions$y, round(res$predictions$y.pred), "bac")
```

---

glmnet_get.items.relevance

Get the relevance or count of the model items

Description

Functions to calculate the relevance of the items of a model or of a list of models or the count of the items of a list of models.

Usage

```
glmnet_get.items.relevance(x, childname = NULL)
glmnet_get.items.count(x, childname = NULL)
```

Arguments

x	an object of class "glmnet_fit", a list of objects of class "glmnet_fit", or a list of objects that have a child of class "glmnet_fit".
childname	name of the child of class "glmnet_fit" (if x is a list of objects that have a child of class "glmnet_fit").

Details

The relevance is calculated as abs(standardized_coefficient) / sum(abs(standardized_coefficients)), as in the function lasso_vars. The count is the proportion of models in which the variable is selected.

Value

A numeric vector representing the relevance or count of the items of the model.

Author(s)

Joaquim Radua, based on the previous work of others (see Details)

References

Palau, P., Solanes, A., Madre, M., Saez-Francas, N., Sarro, S., Moro, N., Verdolini, N., Sanchez, M., Alonso-Lana, S., Amann, B.L., Romaguera, A., Martin-Subero, M., Fortea, L., Fuentes-Claramonte, P., Garcia-Leon, M.A., Munuera, J., Canales-Rodriguez, E.J., Fernandez-Corcuera, P., Brambilla, P., Vieta, E., Pomarol-Clotet, E., Radua, J. (2023) Improved estimation of the risk of manic relapse by combining clinical and brain scan data. *Spanish Journal of Psychiatry and Mental Health*, **16**, 235–243, doi:10.1016/j.rpsm.2023.01.001.

Salazar de Pablo, G., Radua, J., Frearson, G., Young, A.H., Arango, C., Kelleher, I., Sharma, A., Uhlhaas, P.J., Solmi, M., Fusar-Poli, P., Guinart, D., Correll, C.U. (2025) Development and validation of a prognostic model and risk calculator for the estimation of bipolar-spectrum disorder risk in hospitalised adolescents with non-psychotic/non-bipolar mental disorders. *Molecular Psychiatry*, **31**, 1333–1342, doi:10.1038/s41380-025-03244-1.

Radua, J., Rivas-Fernandez, M.A., Lindin, M., Diaz, F., Zurron, M., Pereiro, A.X., Lojo-Seoane, C., De Prisco, M., Oliva, V., Vilajosana, E., Vieta, E., Galdo-Alvarez, S. Submitted

See Also

glmnet_predict for obtaining predictions, cv for conducting a cross-validation; vignette("evaluating-and-interpreti for a tutorial; vignette("missing-data") for a tutorial with several models per fold.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
```

```
  list(
    lasso = glmnet_fit(x_training, y_training, family = "binomial")
  )
}
predict_fun = function (m, x_test) {
  glmnet_predict(m$lasso, x_test)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show the relevance of the predictors
relevance = glmnet_get.items.relevance(res$models, "lasso")
relevance = relevance[which(relevance >= 0.01)] # Select items with >=1% relevance
round(relevance, 2)
```

`glmnet_get.main.model` *Get the main glmnet model across imputations and folds*

Description

Function to choose the `glmnet` model most similar to the other models on the list according to the Dice coefficient.

Usage

```
glmnet_get.main.model(x, childname = NULL, verbose = TRUE)
```

Arguments

`x`	a list of objects of class `"glmnet_fit"` or a list of objects that have a child of class `"glmnet_fit"`.
`childname`	name of the child of class `"glmnet_fit"` (if `x` is a list of objects that have a child of class `"glmnet_fit"`).
`verbose`	(optional) logical, whether to print some messages during execution.

Details

If there are several instances of the most similar model, it averages them.

Value

An object of class `"glmnet_fit"`, representing the model most similar to the other models of the list according to the Dice coefficient.

Author(s)

Joaquim Radua

References

Sobregrau, P., Bailles, E., Radua, J., Carreno, M., Donaire, A., Setoain, X., Bargallo, N., Rumia, J., Sanchez-Vives, M.V., Pintor, L. (2024) Design and validation of a diagnostic suspicion checklist to differentiate epileptic from psychogenic nonepileptic seizures (PNES-DSC). *Journal of Psychosomatic Research*, **180**, 111656, doi:10.1016/j.jpsychores.2024.111656.

See Also

`glmnet_predict` for obtaining predictions. `cv` for conducting a cross-validation; `vignette("evaluating-and-interpreti` for a tutorial; `vignette("missing-data")` for a tutorial with several models per fold.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
  list(
    lasso = glmnet_fit(x_training, y_training, family = "binomial")
  )
}
predict_fun = function (m, x_test) {
  glmnet_predict(m$lasso, x_test)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show the main model
lasso = glmnet_get.main.model(res$models, "lasso")
lasso
```

---

impute.glmnet.matrix_fit

Impute missing variables in a glmnet matrix multiple times

Description

Function to impute, multiple times, the missing variables in a `glmnet.matrix`. `impute.glmnet.matrix_fit` finds the "lasso" models to conduct the imputations, and `impute.glmnet.matrix` does the imputations (in the same or a different dataset).

Usage

```
impute.glmnet.matrix_fit(x, ncores = 1, verbose = TRUE)
impute.glmnet.matrix(m, x, nimp = 20, verbose = TRUE)
```

Arguments

m	model to conduct the imputations, obtained with impute.glmnet.matrix_fit.
x	input matrix for glmnet of dimension nobs x nvars; each row is an observation vector. It can be easily obtained with data.frame2glmnet.matrix.
ncores	number of number of worker nodes (for parallelization).
nimp	number of imputations
verbose	(optional) logical, whether to print some messages during execution.

Details

The user can then obtain a prediction from each dataset and combine the predictions using Rubin's rules (which usually means just averaging them). Note also that this function may take a lot of time, because impute.glmnet.matrix_fit fits imputation models for all variables (not only for those with missing values in x), so that impute.glmnet.matrix can then impute new individuals with missing values in any variable. The results are reproducible with set.seed, also when using several worker nodes.

Value

A list of complete matrixes ready for glmnet_fit and glmnet_predict.

Author(s)

Joaquim Radua and Aleix Solanes

References

Solanes, A., Mezquida, G., Janssen, J., Amoretti, S., Lobo, A., Gonzalez-Pinto, A., Arango, C., Vieta, E., Castro-Fornieles, J., Berge, D., Albacete, A., Gine, E., Parellada, M., Bernardo, M.; PEPs group (collaborators); Pomarol-Clotet, E., Radua, J. (2022) Combining MRI and clinical data to detect high relapse risk after the first episode of psychosis. *Schizophrenia*, **8**, 100, doi:10.1038/s41537-022-00309-w.

Palau, P., Solanes, A., Madre, M., Saez-Francas, N., Sarro, S., Moro, N., Verdolini, N., Sanchez, M., Alonso-Lana, S., Amann, B.L., Romaguera, A., Martin-Subero, M., Fortea, L., Fuentes-Claramonte, P., Garcia-Leon, M.A., Munuera, J., Canales-Rodriguez, E.J., Fernandez-Corcuera, P., Brambilla, P., Vieta, E., Pomarol-Clotet, E., Radua, J. (2023) Improved estimation of the risk of manic relapse by combining clinical and brain scan data. *Spanish Journal of Psychiatry and Mental Health*, **16**, 235–243, doi:10.1016/j.rpsm.2023.01.001.

Salazar de Pablo, G., Radua, J., Frearson, G., Young, A.H., Arango, C., Kelleher, I., Sharma, A., Uhlhaas, P.J., Solmi, M., Fusar-Poli, P., Guinart, D., Correll, C.U. (2025) Development and validation of a prognostic model and risk calculator for the estimation of bipolar-spectrum disorder risk in hospitalised adolescents with non-psychotic/non-bipolar mental disorders. *Molecular Psychiatry*, in Press, doi:10.1038/s41380-025-03244-1.

See Also

glmnet_predict for obtaining predictions. cv for conducting a cross-validation; vignette("missing-data") for a tutorial.

Examples

```
# Quick example

# Create random x with missing values
x = matrix(rnorm(300), ncol = 3)
x = x + rnorm(1) * x[,sample(1:3)] + rnorm(1) * x[,sample(1:3)]
x[sample(1:300, 30)] = NA

# Impute missing values
m_impute = impute.glmnet.matrix_fit(x, ncores = 1)
x_imputed = impute.glmnet.matrix(m_impute, x)


# Complete example (it might take some time even if the example is simple...)

  # Create random x (predictors) and y (binary)
  x = matrix(rnorm(4000), ncol = 20)
  x = x + rnorm(1) * x[,sample(1:20)] + rnorm(1) * x[,sample(1:20)]
  y = 1 * (plogis(x[,1] - x[,2] + rnorm(200, 0, 0.1)) > 0.5)

  # Make some x missing values
  x[sample(1:4000, 400)] = NA

  # Predict y via cross-validation, including imputations
  fit_fun = function (x_training, y_training) {
    m = list(
      impute = impute.glmnet.matrix_fit(x_training, ncores = 1),
      lasso = list()
    )
    x_imputed = impute.glmnet.matrix(m$impute, x_training)
    for (imp in 1:length(x_imputed)) {
      m$lasso[[imp]] = glmnet_fit(x_imputed[[imp]], y_training, family = "binomial")
    }
    m
  }
  predict_fun = function (m, x_test) {
    x_imputed = impute.glmnet.matrix(m$impute, x_test)
    y_pred = NULL
    for (imp in 1:length(x_imputed)) {
      y_pred = cbind(y_pred, glmnet_predict(m$lasso[[imp]], x_imputed[[imp]]))
    }
    apply(y_pred, 1, mean)
  }
  # Only 2 folds to ensure the example runs quickly
 res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

  # Show accuracy and calibration
  res
```

patients *Simulated data of patients followed for relapse*

Description

Simulated datasets used in the vignettes of the package. `patients` contains 100 patients, to be used for building and cross-validating the prediction models, and `new_patients` contains 900 other patients from the same population, to be used for checking how the models actually perform in new patients.

Usage

```
patients
new_patients
```

Format

Data frames with 100 (`patients`) and 900 (`new_patients`) rows and the following 52 columns:

`X1`, ..., `X40` numeric predictors, with values between 0 and 50.

`C1`, `C2`, `C3` categorical predictors (factors with levels `"A"`, `"B"`, and `"C"`).

`L1`, `L2`, `L3` numeric laboratory measures.

`site` hospital of the patient (factor). The patients of `patients` come from four hospitals, and those of `new_patients` from a fifth hospital.

`relapse` binary outcome: 1 if the patient relapsed during the first year, 0 otherwise.

`needs_surgery` binary outcome: 1 if the patient needed surgery, 0 otherwise.

`score_pre`, `score_post` symptom scores at baseline and at follow-up.

`surv` survival outcome (an object of class `"Surv"`): years until relapse, or until the last follow-up for patients who did not relapse.

Details

The relapse depends on 10 of the 40 numeric predictors plus random noise. The baseline symptom score strongly depends on 10 other numeric predictors, and the follow-up score only depends on the baseline score, with a correlation of about 0.2, so that the change between the two scores shows strong regression to the mean. The categorical predictors are unrelated to the outcomes. The need of surgery is unrelated to the predictors. The laboratory measures are also unrelated to the outcomes, but their values differ between hospitals, which also differ in their proportion of patients needing surgery, so they can be used to show the effects of the site.

See Also

the vignettes of the package, which use these data: `vignette("easy.glmnet")`, `vignette("evaluating-and-interpreti`
`vignette("missing-data")`, and `vignette("other-outcomes-and-sites")`.

Examples

```
dim(patients)
table(patients$relapse)
```

`simple.accuracy` *Estimate accuracy in unicentric machine learning studies*

Description

Function to estimate the accuracy of the predictions of a unicentric machine-learning model, along with its 95% confidence (or credible) intervals.

Usage

```
simple.accuracy(y, y.pred, estimate = c("auc", "bac", "cor", "hr", "mse"))
```

Arguments

`y`	actual value of the variable that you want to predict. For `"auc"` estimates, it must be a binary vector. For `"bac"` estimates, it must be either a binary vector or a factor. For `"cor"` and `"mse"` estimates, it must be a numeric vector. For `"hr"` estimates, it must be an object of class `"Surv"`.
`y.pred`	predicted value of the variable. For `"auc"`, `"cor"`, `"hr"`, and `"mse"` estimates, it must be a numeric vector. For `"bac"` estimates, it must be either a binary vector or a factor with the same levels as `y`.
`estimate`	`"auc"` (area under the ROC curve, when `y` is a binary variable and `y.pred` is a continuous variable), `"bac"` (balanced accuracy, along with sensitivity and specificity when both `y` and `y.pred` are binary variables; or along with the sensitivity to detect each group when both `y` and `y.pred` are factors), `"cor"` and `"mse"` (correlation and mean squared error, when both `y` and `y.pred` are numeric variables), `"hr"` (hazard ratio, when `y` is an object of class `"Surv"` and `y.pred` is a numeric variable).

Details

The 95% intervals are estimated as follows:

- For `"auc"`, the DeLong confidence interval, as implemented in `ci.auc`.
- For `"bac"`, the Jeffreys confidence intervals of the sensitivity and specificity (or of the sensitivity to detect each group), i.e., the 2.5 and 97.5 percentiles of a Beta(x + 0.5, n - x + 0.5) distribution, where x is the number of correctly predicted individuals and n the number of individuals of the group, with the lower (upper) limit set to 0 (1) when x = 0 (x = n), as recommended by Brown et al. (2001). For the balanced accuracy itself, the credible interval proposed by Brodersen et al. (2010), i.e., the 2.5 and 97.5 percentiles of the distribution of the mean of independent Beta(x + 1, n - x + 1) variables, obtained by numerical convolution.

- For "cor", the confidence interval based on the Fisher z transformation, as implemented in cor.test.
- For "hr", the Wald confidence interval of the Cox regression of y on y.pred. The function provides both the hazard ratio per unit of y.pred (hr) and per standard deviation of y.pred (hr.per.sd, i.e., y.pred is standardized before fitting the Cox model). Only the latter is comparable between models, because the former depends on the scale of the predictions.
- For "mse", the t-based confidence intervals of the mean squared error of the predictions (mse.pred) and of simply predicting the mean of y for everyone (mse.mean). The function also provides their ratio (mse.pred_div_mse.mean), which is lower than 1 when the predictions are better than the mean.

Note that when the predictions are obtained via cross-validation (e.g., with cv), they are not completely independent, because the models of the different folds share most of their training individuals. Thus, the intervals may be somewhat too narrow.

Value

A data frame with the estimated accuracy and the lower (.lwr) and upper (.upr) limits of its 95% interval.

Author(s)

Joaquim Radua

References

Brodersen, K.H., Ong, C.S., Stephan, K.E., Buhmann, J.M. (2010) The balanced accuracy and its posterior distribution. Proceedings of the 20th International Conference on Pattern Recognition, 3121–3124, doi:10.1109/ICPR.2010.764.

Brown, L.D., Cai, T.T., DasGupta, A. (2001) Interval estimation for a binomial proportion. Statistical Science, 16, 101–133, doi:10.1214/ss/1009213286.

DeLong, E.R., DeLong, D.M., Clarke-Pearson, D.L. (1988) Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics, 44, 837–845, doi:10.2307/2531595.

Salazar de Pablo, G., Radua, J., Frearson, G., Young, A.H., Arango, C., Kelleher, I., Sharma, A., Uhlhaas, P.J., Solmi, M., Fusar-Poli, P., Guinart, D., Correll, C.U. (2025) Development and validation of a prognostic model and risk calculator for the estimation of bipolar-spectrum disorder risk in hospitalised adolescents with non-psychotic/non-bipolar mental disorders. *Molecular Psychiatry*, in Press, doi:10.1038/s41380-025-03244-1.

Solanes, A., Mezquida, G., Janssen, J., Amoretti, S., Lobo, A., Gonzalez-Pinto, A., Arango, C., Vieta, E., Castro-Fornieles, J., Berge, D., Albacete, A., Gine, E., Parellada, M., Bernardo, M.; PEPs group (collaborators); Pomarol-Clotet, E., Radua, J. (2022) Combining MRI and clinical data to detect high relapse risk after the first episode of psychosis. *Schizophrenia*, **8**, 100, doi:10.1038/s41537-022-00309-w.

See Also

simple.accuracy.test for comparing the accuracy of two sets of predictions, simple.calibration for estimating their calibration, and multisite.accuracy for estimating the accuracy of the predictions of a multisite machine-learning model; vignette("easy.glmnet") for a tutorial on balanced accuracy; vignette("evaluating-and-interpreting") for a tutorial on the AUC.

Examples

```
# Simulate data
y = c(scale(rnorm(400)))
y.pred = c(scale(y + rnorm(400)))

# Simple accuracy estimates
print(simple.accuracy(y, y.pred, "mse"))
print(simple.accuracy(y, y.pred, "cor"))

# Binary data
y = rbinom(400, 1, 0.5)
y.pred = plogis(2 * (y - 0.5) + rnorm(400))
print(simple.accuracy(y, y.pred, "auc"))
print(simple.accuracy(y, round(y.pred), "bac"))
```

---

simple.accuracy.test *Compare the accuracy of two sets of predictions in unicentric studies*

Description

Function to compare the accuracy of two sets of predictions of the same individuals, e.g., the predictions of a full model and those of a simpler model, in a unicentric machine-learning study.

Usage

```
simple.accuracy.test(y, y.pred1, y.pred2, estimate = c("auc", "bac", "cor", "hr", "mse"),
                     alternative = "two.sided", nperm = 1000)
```

Arguments

y	actual value of the variable that you want to predict. For "auc" estimates, it must be a binary vector. For "bac" estimates, it must be either a binary vector or a factor. For "cor" and "mse" estimates, it must be a numeric vector. For "hr" estimates, it must be a Surv object.
y.pred1	first set of predictions of the variable. For "auc", "cor", "hr", and "mse" estimates, it must be a numeric vector with the same length as y. For "bac" estimates, it must be either a binary vector or a factor with the same levels as y.
y.pred2	second set of predictions of the variable, with the same format as y.pred1.

estimate	"auc" (area under the ROC curve), "bac" (balanced accuracy, along with sensitivity and specificity, or the sensitivity to detect each group when y is a factor), "cor" (correlation), "hr" (hazard ratio, though the actual comparison is between hazard ratios per SD), or "mse" (mean squared error). See simple.accuracy.
alternative	"two.sided" to test whether the accuracy of the two sets of predictions differs, or "less" / "greater" to test whether the accuracy of y.pred1 is less / greater than that of y.pred2 (i.e., a higher AUC, balanced accuracy, correlation, or hazard ratio per SD, or a lower mean squared error).
nperm	number of permutations, only used for "hr" estimates.

Details

The two sets of predictions must correspond to the same individuals, so the tests account for the fact that they are paired:

- For "auc", the function uses the DeLong test for two correlated ROC curves, as implemented in roc.test.
- For "bac", the function compares the sensitivity and the specificity (or the sensitivity to detect each group) separately, using the McNemar mid-p test, which Fagerland et al. (2013) recommend over the exact (conditional) McNemar test, which is too conservative. It does not provide a p-value for the balanced accuracy itself.
- For "cor", the function compares the two dependent overlapping correlations (i.e., the correlations of y with y.pred1 and with y.pred2) with the Williams (1959) test, as implemented in cocor.dep.groups.overlap of the package **cocor**.
- For "hr", the function compares the hazard ratios per standard deviation of the predictions (i.e., the predictions are standardized before fitting the Cox model) using a paired permutation test, in which the two predictions of each individual are randomly swapped nperm times. As the test is random, use set.seed to obtain reproducible p-values, and note that the smallest possible p-value is 1 / (nperm + 1).
- For "mse", the function compares the squared errors with a paired t-test.

A typical use is to assess whether a full model predicts better than a simpler model, e.g., whether a model predicting symptom change adds information beyond a model that uses only the baseline severity, as recommended to rule out that the apparent accuracy is due to regression to the mean (Radua et al., 2026). In this case, y.pred1 would be the predictions of the full model, y.pred2 those of the simpler model, and alternative = "greater".

Note that when the predictions are obtained via cross-validation (e.g., with cv), they are not completely independent, because the models of the different folds share most of their training individuals. Thus, the p-values may be somewhat optimistic.

Value

A data frame with the estimated accuracy of each set of predictions, the p-value(s), and the test used.

Author(s)

Joaquim Radua

References

DeLong, E.R., DeLong, D.M., Clarke-Pearson, D.L. (1988) Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics, 44, 837–845, doi:10.2307/2531595.

Diedenhofen, B., Musch, J. (2015) cocor: a comprehensive solution for the statistical comparison of correlations. PLoS ONE, 10, e0121945, doi:10.1371/journal.pone.0121945.

Fagerland, M.W., Lydersen, S., Laake, P. (2013) The McNemar test for binary matched-pairs data: mid-p and asymptotic are better than exact conditional. BMC Medical Research Methodology, 13, 91, doi:10.1186/147122881391.

Radua, J., Oliva, V., Vieta, E., De Prisco, M. (2026) Regression to the mean inflates accuracy in machine learning prediction of symptom change. Nature Mental Health, doi:10.1038/s44220026-006866.

Solanes, A., Mezquida, G., Janssen, J., Amoretti, S., Lobo, A., Gonzalez-Pinto, A., Arango, C., Vieta, E., Castro-Fornieles, J., Berge, D., Albacete, A., Gine, E., Parellada, M., Bernardo, M.; PEPs group (collaborators); Pomarol-Clotet, E., Radua, J. (2022) Combining MRI and clinical data to detect high relapse risk after the first episode of psychosis. *Schizophrenia*, **8**, 100, doi:10.1038/s41537-022-00309-w.

Williams, E.J. (1959) The comparison of regression variables. Journal of the Royal Statistical Society, Series B, 21, 396–399.

See Also

`simple.accuracy` for estimating the accuracy of one set of predictions, `cv` for conducting a cross-validation; `vignette("other-outcomes-and-sites")` for a tutorial on regression to the mean.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation, using all predictors or only the first five
fit_fun = function (x_training, y_training) {
  glmnet_fit(x_training, y_training, family = "binomial")
}
predict_fun = function (m, x_test) {
  glmnet_predict(m, x_test)
}
# Only 2 folds to ensure the example runs quickly
res_full = cv(x, y, family = "binomial", fit_fun = fit_fun,
              predict_fun = predict_fun, nfolds = 2)
res_simple = cv(x[, 1:5], y, family = "binomial", fit_fun = fit_fun,
                predict_fun = predict_fun, nfolds = 2)

# Test whether the full model has a greater AUC than the simple model
# (cv returns the predictions in the original order of the individuals).
# As y only depends on the first five predictors, the full model should not
# be better.
simple.accuracy.test(y, res_full$predictions$y.pred, res_simple$predictions$y.pred,
```

```
                    "auc", alternative = "greater")
```

`simple.calibration` *Estimate the calibration of the predictions*

Description

Function to estimate the calibration of the predictions of a prediction model, i.e., the agreement between the predicted and the observed outcomes, along with its 95% confidence intervals.

Usage

```
simple.calibration(y, y.pred, family = c("binomial", "cox", "gaussian"))
```

Arguments

`y`	actual value of the variable that you want to predict. A binary vector for `"binomial"`, an object of class `"Surv"` for `"cox"`, or a numeric vector for `"gaussian"`.
`y.pred`	predicted value of the variable, e.g., as obtained from `glmnet_predict`. A numeric vector with the same length as `y`: probabilities for `"binomial"`, linear predictors for `"cox"`, or predicted values for `"gaussian"`.
`family`	distribution of `y`: `"binomial"`, `"cox"`, or `"gaussian"`.

Details

The calibration-in-the-large assesses whether the predictions are, on average, too high or too low, and its ideal value is 0. For `"binomial"`, it is the intercept of a logistic regression of `y` on the logit of `y.pred`, with the slope fixed to 1 (i.e., using the logit of `y.pred` as an offset); a positive value indicates that the predicted probabilities are too low. For `"gaussian"`, it is the intercept of a linear regression of `y` on `y.pred` with the slope fixed to 1, i.e., the mean of `y - y.pred`. It is not estimated for `"cox"` because the linear predictors do not include the baseline hazard.

The calibration slope assesses whether the predictions are too extreme or too modest, and its ideal value is 1. It is the slope of a logistic regression of `y` on the logit of `y.pred` for `"binomial"`, of a Cox regression of `y` on `y.pred` for `"cox"`, and of a linear regression of `y` on `y.pred` for `"gaussian"`. A slope lower than 1 indicates that the predictions are too extreme, whereas a slope higher than 1 indicates that they are too modest.

The 95% confidence intervals are profile likelihood intervals for `"binomial"`, Wald intervals for `"cox"`, and t-based intervals for `"gaussian"`.

To avoid optimistic estimates, the predictions should be obtained in individuals not used to fit the model, e.g., via `cv`. Note, however, that predictions obtained via cross-validation are not completely independent, because the models of the different folds share most of their training individuals, so the confidence intervals may be somewhat too narrow. Note also that the calibration-in-the-large is not informative if the proportion of events in the sample differs from that of the target population (e.g., in case-control samples).

Value

A data frame with the calibration-in-the-large (`calibr.large`, except for `"cox"`) and the calibration slope (`calibr.slope`), along with the lower (`.lwr`) and upper (`.upr`) limits of their 95% confidence intervals.

Author(s)

Joaquim Radua

References

Palau, P., Solanes, A., Madre, M., Saez-Francas, N., Sarro, S., Moro, N., Verdolini, N., Sanchez, M., Alonso-Lana, S., Amann, B.L., Romaguera, A., Martin-Subero, M., Fortea, L., Fuentes-Claramonte, P., Garcia-Leon, M.A., Munuera, J., Canales-Rodriguez, E.J., Fernandez-Corcuera, P., Brambilla, P., Vieta, E., Pomarol-Clotet, E., Radua, J. (2023) Improved estimation of the risk of manic relapse by combining clinical and brain scan data. *Spanish Journal of Psychiatry and Mental Health*, **16**, 235–243, doi:10.1016/j.rpsm.2023.01.001.

Radua, J., Rivas-Fernandez, M.A., Lindin, M., Diaz, F., Zurron, M., Pereiro, A.X., Lojo-Seoane, C., De Prisco, M., Oliva, V., Vilajosana, E., Vieta, E., Galdo-Alvarez, S. Submitted

Van Calster, B., McLernon, D.J., van Smeden, M., Wynants, L., Steyerberg, E.W.; Topic Group 'Evaluating diagnostic tests and prediction models' of the STRATOS initiative (2019) Calibration: the Achilles heel of predictive analytics. BMC Medicine, 17, 230, doi:10.1186/s1291601914667.

See Also

`simple.accuracy` for estimating the accuracy of the predictions, `cv` for conducting a cross-validation; `vignette("evaluating-and-interpreting")` for a tutorial.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
  glmnet_fit(x_training, y_training, family = "binomial")
}
predict_fun = function (m, x_test) {
  glmnet_predict(m, x_test)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Estimate the calibration
simple.calibration(res$predictions$y, res$predictions$y.pred, "binomial")
```

surv2binary *Convert a "Surv" object into binary variables at different time points*

Description

Function to convert a "Surv" object (e.g., the predictions obtained from glmnet_predict using a "cox" model) into a list of binary variables (e.g., as obtained from glmnet_predict using a "binomial" model) at different time points.

Usage

```
surv2binary(x, subset_id = NULL)
```

Arguments

x	a "Surv" object.
subset_id	vector or factor identifying the subsets for which the conversion will be performed separately.

Details

This function is useful, for instance, to estimate the AUC at different timepoints from "cox" predictions.

Value

A list of times and binary variables.

Author(s)

Joaquim Radua

References

Salazar de Pablo, G., Radua, J., Frearson, G., Young, A.H., Arango, C., Kelleher, I., Sharma, A., Uhlhaas, P.J., Solmi, M., Fusar-Poli, P., Guinart, D., Correll, C.U. (2025) Development and validation of a prognostic model and risk calculator for the estimation of bipolar-spectrum disorder risk in hospitalised adolescents with non-psychotic/non-bipolar mental disorders. *Molecular Psychiatry*, in Press, doi:10.1038/s41380-025-03244-1.

See Also

glmnet_predict for obtaining "cox" predictions. cv for conducting a cross-validation; vignette("other-outcomes-and-s
for a tutorial.

Examples

```
library(survival)

# Create random x (predictors) and y (survival)
x = matrix(rnorm(5000), ncol = 10)
time = rexp(500)
y = Surv(time, plogis(x[,1] / pmax(1, time^2) + rnorm(500)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x, y) {
  glmnet_fit(x, y, family = "cox")
}
predict_fun = function (m, x) {
  glmnet_predict(m, x)
}
res = cv(x, y, family = "cox", fit_fun = fit_fun, predict_fun = predict_fun)

# Convert y to binary
y.binary = surv2binary(y)

# Calculate and plot AUC for binary y at each timepoint
time_auc = NULL
for (i in 1:length(y.binary)) {
  status_i = y.binary[[i]]$status
  # The status of the individuals censored before the timepoint is unknown (NA)
  known = which(!is.na(status_i))
  if (length(unique(status_i[known])) == 2) {
    time_auc = rbind(time_auc, data.frame(
      time = y.binary[[i]]$time,
      auc = simple.accuracy(status_i[known], res$predictions$y.pred[known], "auc")$auc
    ))
  }
}
plot(time_auc$time, time_auc$auc, type = "l", xlab = "Time", ylab = "AUC", ylim = 0:1)
abline(h = 0.5)
```

Index

∗ **datasets**
    patients, 15

assign.folds, 2, *4*

ci.auc, *16*
coef.glmnet_fit *(*glmnet_fit*)*, 7
cor.test, *17*
cv, *3*, 3, *6*, *9*, *10*, *12*, *13*, *17*, *19–23*
cv.glmnet, *8*

data.frame2glmnet.matrix, 6, *7*, *13*
data.frame2glmnet.matrix_fit, *6*
data.frame2glmnet.matrix_fit
        *(*data.frame2glmnet.matrix*)*, 6

glmnet, *8*
glmnet_fit, *6*, 7, *8*, *13*
glmnet_get.items.count
        *(*glmnet_get.items.relevance*)*, 9
glmnet_get.items.relevance, 9
glmnet_get.main.model, 11
glmnet_predict, *5*, *6*, *10*, *12*, *13*, *21*, *23*
glmnet_predict *(*glmnet_fit*)*, 7

impute.glmnet.matrix
        *(*impute.glmnet.matrix_fit*)*, 12
impute.glmnet.matrix_fit, 12

lasso_vars, *10*

multisite.accuracy, *5*, *18*

new_patients *(*patients*)*, 15

patients, 15
predict.glmnet_fit *(*glmnet_fit*)*, 7
print.cv *(*cv*)*, 3
print.glmnet_fit *(*glmnet_fit*)*, 7

roc.test, *19*

set.seed, *13*, *19*
simple.accuracy, *5*, 16, *19*, *20*, *22*
simple.accuracy.test, *18*, 18
simple.calibration, *5*, *18*, 21
surv2binary, 23