Package 'errors'

November 4, 2026

Type Package

Title Uncertainty Propagation for R Vectors

Version 0.4.5

Description Support for measurement errors in R vectors, matrices and arrays: automatic uncertainty propagation and reporting.
Documentation about 'errors' is provided in the paper by Ucar, Pebesma & Azcorra (2018, <doi:10.32614/RJ-2018-075>), included in this package as a vignette; see 'citation(``errors'')' for details.

License MIT + file LICENSE

Encoding UTF-8

URL https://r-quantities.github.io/errors/, https://github.com/r-quantities/errors

BugReports https://github.com/r-quantities/errors/issues

LazyData true

Depends R (>= 3.0.0)

Suggests dplyr (>= 1.0.0), vctrs (>= 0.5.0), pillar, ggplot2 (>= 3.5.0), testthat, vdiffr, knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Iñaki Ucar [aut, cph, cre] (ORCID: <https://orcid.org/0000-0001-6403-5550>),
Lionel Henry [ctb],
RStudio [cph] (Copyright for code written by RStudio employees.)

Maintainer Iñaki Ucar <iucar@fedoraproject.org>

Repository CRAN

Date/Publication 2026-11-03 23:00:29 UTC

Contents

errors-package . . . 2
as.data.frame.errors . . . 4
as.list.errors . . . 4
as.matrix.errors . . . 5
c.errors . . . 5
cbind.errors . . . 6
correl . . . 7
datasets . . . 8
diff.errors . . . 9
drop_errors . . . 10
errors . . . 10
Extract.errors . . . 12
format.errors . . . 12
geom_errors . . . 13
groupGeneric.errors . . . 16
mean.errors . . . 17
plot.errors . . . 18
print.errors . . . 19
rep.errors . . . 19
scale_errors . . . 20
t.errors . . . 21

Index **22**

`errors-package` **errors**: *Uncertainty Propagation for R Vectors*

Description

Support for measurement errors in R vectors, matrices and arrays: automatic uncertainty propagation and reporting.

Details

Every measurement has an unknown error associated. Uncertainty is the acknowledgement of that error: we are aware that our representation of reality may differ from reality itself. This package provides support for measurement errors in R vectors, matrices and arrays. Uncertainty metadata is associated to quantity values (see `errors`), and this uncertainty is automatically propagated when you operate with `errors` objects (see `groupGeneric.errors`), or with `errors` and numeric objects (then numeric values are automatically coerced to `errors` objects with no uncertainty).

Correlations between measurements are also supported. In particular, any operation (e.g., `z <- x + y`) results in a correlation between output and input variables (i.e., `z` is correlated to `x` and `y`, even if there was no correlation between `x` and `y`). And in general, the user can establish correlations between any pair of variables (see `correl`).

This package treats uncertainty as coming from Gaussian and linear sources (note that, even for non-Gaussian non-linear sources, this is a reasonable assumption for averages of many measurements),

and propagates them using the first-order Taylor series method for propagation of uncertainty. Although the above assumptions are valid in a wide range of applications in science and engineering, the practitioner should evaluate whether they apply for each particular case.

Author(s)

Iñaki Ucar

References

Iñaki Ucar, Edzer Pebesma and Arturo Azcorra (2018). Measurement Errors in R. *The R Journal*, 10(2), 549-557. doi:10.32614/RJ2018075

See Also

`datasets` for a description of the datasets used in the examples below.

Examples

```
## Simultaneous resistance and reactance measurements

# Obtain mean values and uncertainty from measured values
V   <- mean(set_errors(GUM.H.2$V))
I   <- mean(set_errors(GUM.H.2$I))
phi <- mean(set_errors(GUM.H.2$phi))

# Set correlations between variables
correl(V, I)   <- with(GUM.H.2, cor(V, I))
correl(V, phi) <- with(GUM.H.2, cor(V, phi))
correl(I, phi) <- with(GUM.H.2, cor(I, phi))

# Computation of resistance, reactance and impedance values
(R <- (V / I) * cos(phi))
(X <- (V / I) * sin(phi))
(Z <- (V / I))

# Correlations between derived quantities
correl(R, X)
correl(R, Z)
correl(X, Z)

## Calibration of a thermometer

# Least-squares fit for a reference temperature of 20 degC
fit <- lm(bk ~ I(tk - 20), data = GUM.H.3)

# Extract coefficients and set correlation using the covariance matrix
y1 <- set_errors(coef(fit)[1], sqrt(vcov(fit)[1, 1]))
y2 <- set_errors(coef(fit)[2], sqrt(vcov(fit)[2, 2]))
covar(y1, y2) <- vcov(fit)[1, 2]

# Predicted correction for 30 degC
```

```
(b.30 <- y1 + y2 * set_errors(30 - 20))
```

`as.data.frame.errors` *Coerce to a Data Frame*

Description

S3 method for `errors` objects (see `as.data.frame`).

Usage

```
## S3 method for class 'errors'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	any R object.
`row.names`	`NULL` or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	logical. If `TRUE`, setting row names and converting column names (to syntactic names: see `make.names`) is optional. Note that all of R's **base** package `as.data.frame()` methods use `optional` only for column names treatment, basically with the meaning of `data.frame(*, check.names = !optional)`. See also the `make.names` argument of the `matrix` method.
`...`	additional arguments to be passed to or from methods.

Examples

```
x <- set_errors(1:3, 0.1)
y <- set_errors(4:6, 0.2)
(z <- cbind(x, y))
as.data.frame(z)
```

`as.list.errors` *Coerce to a List*

Description

S3 method for `errors` objects (see `as.list`).

Usage

```
## S3 method for class 'errors'
as.list(x, ...)
```

Arguments

x	object to be coerced or tested.
...	objects, possibly named.

Examples

```
x <- set_errors(1:3, 0.1)
as.list(x)
```

---

as.matrix.errors *Coerce to a Matrix*

Description

S3 method for errors objects (see as.matrix).

Usage

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

Arguments

x	an R object.
...	additional arguments to be passed to or from methods.

Examples

```
as.matrix(set_errors(1:3, 0.1))
```

---

c.errors *Combine Values into a Vector or List*

Description

S3 method for errors objects (see c).

Usage

```
## S3 method for class 'errors'
c(..., recursive = FALSE)
```

Arguments

`...`	objects to be concatenated. All `NULL` entries are dropped before method dispatch unless at the very beginning of the argument list.
`recursive`	logical. If `recursive = TRUE`, the function recursively descends through lists (and pairlists) combining all their elements into a vector.

Examples

```
c(set_errors(1, 0.2), set_errors(7:9, 0.1), 3)
```

`cbind.errors` *Combine* R *Objects by Rows or Columns*

Description

S3 methods for `errors` objects (see `cbind`).

Usage

```
## S3 method for class 'errors'
cbind(..., deparse.level = 1)

## S3 method for class 'errors'
rbind(..., deparse.level = 1)
```

Arguments

`...`	(generalized) vectors or matrices. These can be given as named arguments. Other R objects may be coerced as appropriate, or S4 methods may be used: see sections 'Details' and 'Value'. (For the `"data.frame"` method of `cbind` these can be further arguments to `data.frame` such as `stringsAsFactors`.)
`deparse.level`	integer controlling the construction of labels in the case of non-matrix-like arguments (for the default method): `deparse.level = 0` constructs no labels; the default `deparse.level = 1` typically and `deparse.level = 2` always construct labels from the argument names, see the 'Value' section below.

See Also

`c.errors`

Examples

```
x <- set_errors(1, 0.1)
y <- set_errors(1:3, 0.2)
(m <- cbind(x, y)) # the '1' (= shorter vector) is recycled
(m <- cbind(m, 8:10)[, c(1, 3, 2)]) # insert a column
cbind(y, diag(3)) # vector is subset -> warning
cbind(0, rbind(x, y))
```

`correl` *Handle Correlations Between* `errors` *Objects*

Description

Set or retrieve correlations or covariances between `errors` objects. See the details section below.

Usage

```
correl(x, y)

correl(x, y) <- value

set_correl(x, y, value)

covar(x, y)

covar(x, y) <- value

set_covar(x, y, value)
```

Arguments

`x`	an object of class `errors`.
`y`	an object of class `errors`.
`value`	a numeric vector of length 1 or the same length as `x`.

Details

The uncertainties associated to `errors` objects are supposed to be independent by default. If there is some known correlation, it can be defined using these methods, and it will be used for the propagation of the uncertainty by the mathematical and arithmetic operations.

The `correl` method sets or retrieves correlations, i.e., a value (or vector of values) between `-1` and `1` (see base `cor` on how to compute correlations). A covariance is just a correlation value multiplied by the standard deviations (i.e., the standard uncertainty) of both variables. It can be defined using the `covar` method (see base `cov` on how to compute covariances). These methods are equivalent; in fact, `correl` calls `covar` internally.

Every errors object has a unique ID, and pairwise correlations are stored in an internal hash table. All the functions or methods that modify somehow the dimensions of errors objects (i.e., subsets, binds, concatenations, summaries...) generate new objects with new IDs, and correlations are not, and cannot be, propagated. Only mathematical and arithmetic operations propagate correlations, where appropriate, following the Taylor series method.

Value

correl and covar return a vector of correlations and covariances respectively (or NULL). set_correl and set_covar, which are pipe-friendly versions of the setters, return the x object.

Examples

```
x <- set_errors(1:5, 0.1)
y <- set_errors(1:5, 0.1)

# Self-correlation is of course 1, and cannot be changed
correl(x, x)
## Not run:
correl(x, x) <- 0.5
## End(Not run)

# Cross-correlation can be set, but must be a value between -1 and 1
correl(x, y)
## Not run:
correl(x, y) <- 1.5
## End(Not run)
correl(x, y) <- runif(length(x))
correl(x, y)
covar(x, y)
```

---

datasets *Datasets from the Guide to the Expression of Uncertainty in Measurement (GUM)*

Description

Datasets found in Annex H of the GUM (see reference below).

Usage

```
GUM.H.2

GUM.H.3
```

Format

GUM.H.2, from Section 2 of Annex H (Table H.2), provides simultaneous resistance and reactance measurements. It is a data frame with 5 rows and 3 variables:

V Voltage amplitude, in Volts.

I Current amplitude, in Amperes.

phi Phase-shift angle of the voltage relative to the current, in radians.

GUM.H.3, from Section 3 of Annex H (Table H.6), provides thermometer readings and observed corrections to obtain a linear calibration curve for some reference temperature. It is a data frame with 11 rows and 2 variables:

tk Thermometer reading, in Celsius degrees.

bk Observed correction, in Celsius degrees.

An object of class data.frame with 11 rows and 2 columns.

Source

BIPM, IEC, IFCC, ILAC, IUPAC, IUPAP, ISO, and OIML (2008). Evaluation of Measurement Data – Guide to the Expression of Uncertainty in Measurement, 1st edn. JCGM 100:2008. *Joint Committee for Guides in Metrology.* https://www.bipm.org/en/committees/jc/jcgm/publications

See Also

See errors-package for examples.

---

diff.errors *Lagged Differences*

Description

S3 method for errors objects (see diff).

Usage

```
## S3 method for class 'errors'
diff(x, lag = 1L, differences = 1L, ...)
```

Arguments

x	a numeric vector or matrix containing the values to be differenced.
lag	an integer indicating which lag to use.
differences	an integer indicating the order of the difference.
...	further arguments to be passed to or from methods.

Examples

```
diff(set_errors(1:10, 0.1), 2)
diff(set_errors(1:10, 0.1), 2, 2)
x <- cumsum(cumsum(set_errors(1:10, 0.1)))
diff(x, lag = 2)
diff(x, differences = 2)
```

`drop_errors` *Drop Uncertainty*

Description

Drop Uncertainty

Usage

```
drop_errors(x)

## S3 method for class 'data.frame'
drop_errors(x)
```

Arguments

`x`	an `errors` object.

Value

the numeric without any `errors` attributes, while preserving other attributes like dimensions or other classes.

Note

Equivalent to `errors(x) <- NULL` or `set_errors(x, NULL)`.

`errors` *Handle Uncertainty on a Numeric Vector*

Description

Set or retrieve uncertainty to/from numeric vectors.

Usage

```
errors(x)

errors_max(x)

errors_min(x)

errors(x) <- value

set_errors(x, value = 0)

as.errors(x, value = 0)
```

Arguments

x	a numeric object, or object of class errors.
value	a numeric vector of length 1 or the same length as x.

Details

`errors<-` sets the uncertainty values (and converts x into an object of class errors). set_errors is a pipe-friendly version of `errors<-` and returns an object of class errors. as.errors is an alias for set_errors.

See correl on how to handle correlations between pairs of variables.

Value

errors returns a vector of uncertainty. errors_max (errors_min) returns the values plus (minus) the uncertainty.

See Also

groupGeneric.errors, mean.errors, Extract.errors, c, rep, cbind.errors, format.errors, print.errors, plot.errors, as.data.frame.errors, as.matrix.errors, t.

Examples

```
x = 1:3
class(x)
x
errors(x) <- 0.1
class(x)
x

(x <- set_errors(x, seq(0.1, 0.3, 0.1)))
errors_max(x)
errors_min(x)
```

Extract.errors *Extract or Replace Parts of an Object*

Description

S3 operators to extract or replace parts of `errors` objects.

Usage

```
## S3 method for class 'errors'
x[...]

## S3 method for class 'errors'
x[[...]]

## S3 replacement method for class 'errors'
x[...] <- value

## S3 replacement method for class 'errors'
x[[...]] <- value
```

Arguments

`x`	object from which to extract element(s) or in which to replace element(s).
`...`	additional arguments to be passed to base methods (see `Extract`).
`value`	typically an array-like R object of a similar class as `x`.

Examples

```
x <- set_errors(1:3, 0.1)
y <- set_errors(4:6, 0.2)
(z <- rbind(x, y))
z[2, 2]
z[2, 2] <- -1
errors(z[[1, 2]]) <- 0.8
z[, 2]
```

format.errors *Encode* `errors`

Description

Format an `errors` object for pretty printing.

Usage

```
## S3 method for class 'errors'
format(x, digits = NULL, scientific = FALSE,
  notation = getOption("errors.notation", "parenthesis"),
  decimals = getOption("errors.decimals", FALSE), ...)
```

Arguments

x	an errors object.
digits	how many significant digits are to be used for uncertainties. The default, NULL, uses getOption("errors.digits", 1). Use digits="pdg" to choose an appropriate number of digits for each value according to the Particle Data Group rounding rule (see references).
scientific	logical specifying whether the elements should be encoded in scientific format.
notation	error notation; "parenthesis" and "plus-minus" are supported through the "errors.notation" option.
decimals	logical specifying whether the uncertainty should be formatted with a decimal point even when the "parenthesis" notation is used. Otherwise (by default), the "parenthesis" notation scales the uncertainty to match the least significant digit of the value.
...	ignored.

References

K. Nakamura et al. (Particle Data Group), J. Phys. G 37, 075021 (2010)

Examples

```
x <- set_errors(1:3*100, 1:3*100 * 0.05)
format(x)
format(x, digits=2)
format(x, digits=2, decimals=TRUE)
format(x, scientific=TRUE)
format(x, notation="plus-minus")

x <- set_errors(c(0.827, 0.827), c(0.119, 0.367))
format(x, notation="plus-minus", digits="pdg")
```

---

geom_errors *Errorbars for* errors *objects*

Description

Automatic errorbars for variables with uncertainty.

Usage

```
geom_errors(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., na.rm = FALSE, orientation = NA,
  show.legend = NA, inherit.aes = TRUE)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth

= 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept.
- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

na.rm — If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

orientation — The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Aesthetics

geom_errors() understands the following aesthetics (required aesthetics are in bold):

- x ***or*** y
- alpha
- colour
- group
- linetype
- linewidth

Examples

```
if (requireNamespace("ggplot2", quietly=TRUE)) {
```

```
iris.e <- iris
iris.e[1:4] <- lapply(iris.e[1:4], function(x) set_errors(x, x*0.02))

library(ggplot2)

ggplot(iris.e) + aes(Sepal.Length, Sepal.Width, color=Species) +
  geom_point() + geom_errors()

ggplot(iris.e) + aes(Sepal.Length, Sepal.Width, color=Species) +
  geom_point() + geom_errors(width=0.05, height=0.05, linewidth=0.2)

}
```

`groupGeneric.errors` *S3 Group Generic Functions*

Description

`Math`, `Ops` and `Summary` group generic methods for `errors` objects with support for automatic uncertainty propagation (see `groupGeneric` for a comprehensive list of available methods).

Usage

```
## S3 method for class 'errors'
Math(x, ...)

## S3 method for class 'errors'
Ops(e1, e2)

## S3 method for class 'errors'
Summary(..., na.rm = FALSE)
```

Arguments

`x`, `e1`, `e2`	objects.
`...`	further arguments passed to methods.
`na.rm`	logical: should missing values be removed?

Details

`Math`**:** The `sign` method returns a numeric value without uncertainty. `floor`, `ceiling`, `trunc`, `round` and `signif` add the rounding error to the original uncertainty. `lgamma`, `gamma`, `digamma` and `trigamma` are not implemented. The rest of the methods propagate the uncertainty as expected from the first-order Taylor series method.

`Ops`**:** Boolean operators drop the uncertainty and operate on the numeric values unless the option `errors.compare.probabilistic` is set to `TRUE`. In such case the comparison operators return

a numeric value between 0 and 1, representing the probability that the comparison is true, assuming normal distribution of the errors. The rest of the operators propagate the uncertainty as expected from the first-order Taylor series method. Any numeric operand is automatically coerced to `errors` with no uncertainty.

`Summary`: The methods `all` and `any` are not supported for `errors` objects and fail with an informative message. `min`, `max` (and `range`) return the minimum or (and) maximum value minus/plus its uncertainty. `sum` and `prod` propagate the uncertainty as expected from the first-order Taylor series method.

Examples

```
x <- set_errors(1:3, 0.1)
exp(x)
log(x)
cumsum(x)
cumprod(x)

y <- set_errors(1:3 + 0.1, c(0, 0.1, 0.2))
x / sqrt(y) + y * sin(x)

# numeric values are automatically coerced to errors
x^2

# boolean operators drop uncertainty
y > x
# unless probabilistic comparisons are enabled
options(errors.compare.probabilistic = TRUE)
y > x
options(errors.compare.probabilistic = NULL)

c(min(x), max(x))
range(x)
sum(y)
prod(y)
```

---

mean.errors *Arithmetic Mean and Median Value*

Description

S3 methods for `errors` objects.

Usage

```
## S3 method for class 'errors'
mean(x, trim = 0, na.rm = FALSE, ...)
```

```
## S3 method for class 'errors'
weighted.mean(x, ..., na.rm = FALSE)

## S3 method for class 'errors'
median(x, na.rm = FALSE, ...)
```

Arguments

x	an errors object.
trim	the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm	a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to of from other methods.

Details

The mean and weighted.mean methods set the uncertainty as the maximum of the standard deviation of the mean and the (weighted) mean of the uncertainty.

The median method sets the uncertainty as 1.253 * errors(mean(x)), which is derived from the asymptotic variance formula of the median. Note that this value is valid only if the sample is big enough.

Value

An errors object.

---

plot.errors *Scatterplot with Error Bars*

Description

S3 method for errors objects which automatically prints the error bars.

Usage

```
## S3 method for class 'errors'
plot(x, y, ...)
```

Arguments

x, y	the x and y arguments provide the x and y coordinates for the plot. Any reasonable way of defining the coordinates is acceptable. See the function xy.coords for details. If supplied separately, they must be of the same length.
...	additional arguments (see plot).

Examples

```
cars <- as.matrix(cars)
cars <- as.data.frame(set_errors(cars, cars * 0.05))
plot(cars$speed)
plot(cars)
```

---

print.errors *Print Values*

Description

S3 method for errors objects.

Usage

```
## S3 method for class 'errors'
print(x, ...)
```

Arguments

x	an errors object.
...	further arguments passed to or from other methods.

Examples

```
x <- set_errors(1:10, 1:10 * 0.05)
print(x)
print(x[1:3])
print(x[1])
print(x[1], digits=2)
print(x[1], notation="plus-minus")
```

---

rep.errors *Replicate Elements of Vectors and Lists*

Description

S3 method for errors objects (see rep).

Usage

```
## S3 method for class 'errors'
rep(x, ...)
```

Arguments

x	a vector (of any mode including a list) or a factor or (for rep only) a POSIXct or POSIXlt or Date object; or an S4 object containing such an object.
...	further arguments to be passed to or from other methods. For the internal default method these can include:
	times an integer-valued vector giving the (non-negative) number of times to repeat each element if of length length(x) * each, or to repeat the whole vector if of length 1. Negative or NA values are an error. A double vector is accepted, other inputs being coerced to an integer or double vector.
	length.out non-negative integer. The desired length of the output vector. Other inputs will be coerced to a double vector and the first element taken. Ignored if NA or invalid.
	each non-negative integer. Each element of x is repeated each times. Other inputs will be coerced to an integer or double vector and the first element taken. Treated as 1 if NA or invalid.

Examples

```
rep(set_errors(1, 0.1), 4)
```

---

scale_errors *Continuous scales for* errors *objects*

Description

Default scales for the errors class.

Usage

```
scale_x_errors(...)

scale_y_errors(...)

scale_colour_errors(...)

scale_color_errors(...)

scale_fill_errors(...)

scale_alpha_errors(...)

scale_size_errors(...)

scale_size_area_errors(...)
```

```
scale_radius_errors(...)

scale_linewidth_errors(...)
```

Arguments

`...`	arguments passed on to the corresponding continuous scale (see the manual page for each `scale_{type}` for details).

`t.errors` *Matrix Transpose*

Description

S3 method for `errors` objects (see `t`).

Usage

```
## S3 method for class 'errors'
t(x)
```

Arguments

`x`	a matrix or data frame, typically.

Examples

```
a <- matrix(1:30, 5, 6)
errors(a) <- 1:30
t(a)
```

Index

∗ **datasets**
    datasets, 8
[.errors *(*Extract.errors*)*, 12
[<-.errors *(*Extract.errors*)*, 12
[[.errors *(*Extract.errors*)*, 12
[[<-.errors *(*Extract.errors*)*, 12

aes(), *14*
annotation_borders(), *15*
as.data.frame, *4*
as.data.frame.errors, 4, *11*
as.errors *(*errors*)*, 10
as.list, *4*
as.list.errors, 4
as.matrix, *5*
as.matrix.errors, 5, *11*

c, *5*, *11*
c.errors, 5, *6*
cbind, *6*
cbind.errors, 6, *11*
cor, *7*
correl, *2*, 7, *11*
correl<- *(*correl*)*, 7
cov, *7*
covar *(*correl*)*, 7
covar<- *(*correl*)*, 7

data.frame, *4*, *6*
datasets, *3*, 8
diff, *9*
diff.errors, 9
drop_errors, 10

errors, *2*, 10
errors-package, 2
errors<- *(*errors*)*, 10
errors_max *(*errors*)*, 10
errors_min *(*errors*)*, 10
Extract, *12*
Extract.errors, *11*, 12

format.errors, *11*, 12
fortify(), *14*

geom_errors, 13
ggplot(), *14*
groupGeneric, *16*
groupGeneric.errors, *2*, *11*, 16
GUM.H.2 *(*datasets*)*, 8
GUM.H.3 *(*datasets*)*, 8

key glyphs, *15*

layer position, *14*
layer stat, *14*
layer(), *14*, *15*
list, *20*

make.names, *4*
Math.errors *(*groupGeneric.errors*)*, 16
mean.errors, *11*, 17
median.errors *(*mean.errors*)*, 17

NULL, *6*

Ops.errors *(*groupGeneric.errors*)*, 16

plot, *18*
plot.errors, *11*, 18
print.errors, *11*, 19

rbind.errors *(*cbind.errors*)*, 6
rep, *11*, *19*
rep.errors, 19

scale_alpha_errors *(*scale_errors*)*, 20
scale_color_errors *(*scale_errors*)*, 20
scale_colour_errors *(*scale_errors*)*, 20
scale_errors, 20
scale_fill_errors *(*scale_errors*)*, 20

scale_linewidth_errors (scale_errors), 20
scale_radius_errors (scale_errors), 20
scale_size_area_errors (scale_errors), 20
scale_size_errors (scale_errors), 20
scale_x_errors (scale_errors), 20
scale_y_errors (scale_errors), 20
set_correl (correl), 7
set_covar (correl), 7
set_errors (errors), 10
Summary.errors (groupGeneric.errors), 16

t, *11*, *21*
t.errors, 21

weighted.mean.errors (mean.errors), 17

xy.coords, *18*