Package 'MESS'

September 13, 2025

Type Package

Title Miscellaneous Esoteric Statistical Scripts

Version 0.6.0

Date 2025-09-10

Maintainer Claus Thorn Ekstrøm <claus@rprimer.dk>

Depends R (>= 4.1)

Imports MASS, Matrix, Rcpp, clipr, geepack, geeM, ggplot2, ggformula, glmnet, kinship2, methods, mvtnorm, parallel

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Suggests knitr, lme4, magrittr, rmarkdown, testthat

Description A mixed collection of useful and semi-useful diverse statistical functions, some of which may even be referenced in The R Primer book. See Ekstrøm, C. T. (2016). The R Primer. 2nd edition. Chapman & Hall.

URL https://github.com/ekstroem/MESS

BugReports https://github.com/ekstroem/MESS/issues

Encoding UTF-8

ByteCompile true

License GPL-2

RoxygenNote 7.3.3

NeedsCompilation yes

Author Claus Thorn Ekstrøm [aut, cre], Niels Aske Lundtorp Olsen [ctb]

Repository CRAN

Date/Publication 2025-09-13 04:30:02 UTC

Contents

adaptive.weights . . . 4
add_torows . . . 5
age . . . 6
auc . . . 7
bdstat . . . 8
bees . . . 9
bin . . . 10
categorize . . . 11
clipit . . . 12
clotting . . . 12
cmd . . . 13
col.alpha . . . 14
col.shade . . . 15
col.tint . . . 16
colCumSum . . . 16
common.shared . . . 17
conditional_rowMeans . . . 18
cumsumbinning . . . 19
dCor . . . 20
dCov . . . 20
drop1.geeglm . . . 21
drop1.geem . . . 22
earthquakes . . . 23
expand_table . . . 24
extended.shared . . . 24
fac2num . . . 26
feature.test . . . 26
filldown . . . 28
founder.shared . . . 29
geekin . . . 30
gkgamma . . . 32
greenland . . . 33
happiness . . . 34
ht . . . 35
hwe_frequencies . . . 36
icecreamads . . . 37
ks_cumtest . . . 38
kwdata . . . 39
lifeexpect . . . 39
loadRData . . . 40
lower.tri.vector . . . 41
matched . . . 42
maximum_subarray . . . 42
mfastLmCpp . . . 43
monte_carlo_chisq_test . . . 44
nh4 . . . 45

ordered.clusters . . . 46
pairwise.cor.test . . . 46
pairwise_combination_indices . . . 47
pairwise_distance_correlation . . . 48
pairwise_Schur_product . . . 49
panel.hist . . . 50
panel.r2 . . . 51
picea . . . 52
plr . . . 52
power_binom_test . . . 53
power_mcnemar_test . . . 54
power_prop_test . . . 56
power_t_test . . . 58
prepost.test . . . 59
qdiag . . . 61
QIC.geeglm . . . 61
qpcr . . . 63
quadform . . . 64
rainman . . . 64
repmat . . . 66
residualplot.default . . . 67
residual_plot . . . 69
rmvt.pedigree . . . 72
rmvtnorm.pedigree . . . 73
rnorm_perfect . . . 74
rootonorm . . . 75
round_percent . . . 77
rud . . . 78
scorefct . . . 79
screen_variables . . . 79
segregate.genes . . . 80
sinv . . . 81
smokehealth . . . 82
soccer . . . 83
superroot2 . . . 83
tracemp . . . 84
usd . . . 85
wallyplot.default . . . 86
write.xml . . . 88

Index **90**

`adaptive.weights` *Compute weights for use with adaptive lasso.*

Description

Fast computation of weights needed for adaptive lasso based on Gaussian family data.

Usage

```
adaptive.weights(x, y, nu = 1, weight.method = c("multivariate", "univariate"))
```

Arguments

`x`	input matrix, of dimension nobs x nvars; each row is an observation vector.
`y`	response variable.
`nu`	non-negative tuning parameter
`weight.method`	Should the weights be computed for multivariate regression model (only possible when the number of observations is larger than the number of parameters) or by individual marginal/"univariate" regression coefficients.

Details

The weights returned are 1/abs(beta_hat)^nu where the beta-parameters are estimated from the corresponding linear model (either multivariate or univariate).

Value

Returns a list with two elements:

`weights`	the computed weights
`nu`	the value of nu used for the computations

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Xou, H (2006). The Adaptive Lasso and Its Oracle Properties. JASA, Vol. 101.

See Also

`glmnet`

Examples

```
set.seed(1)
x <- matrix(rnorm(50000), nrow=50)
y <- rnorm(50, mean=x[,1])
weights <- adaptive.weights(x, y)

if (requireNamespace("glmnet", quietly = TRUE)) {
    res <- glmnet::glmnet(x, y, penalty.factor=weights$weights)
    head(res)
}
```

`add_torows` *Fast addition of vector to each row of matrix*

Description

Fast addition of vector to each row of a matrix. This corresponds to t(t(x) + v)

Usage

```
add_torows(x, v)
```

Arguments

x	A matrix with dimensions n*k.
v	A vector of length k.

Value

A matrix of dimension n*k where v is added to each row of x

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
A <- matrix(1:12, ncol=3)
B <- c(1, 2, 3)

add_torows(A, B)
```

age *Compute the age of a person from two dates.*

Description

Compute the age in years of an individual based on the birth date and another (subsequent) date

Usage

```
age(from, to)
```

Arguments

from	a vector of dates (birth dates)
to	a vector of current dates

Details

Returns the full number of years that a person is old on a given date.

Value

A vector of ages (in years)

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

as.POSIXlt

Examples

```
born <- c("1971-08-18", "2000-02-28", "2001-12-20")
check <- c("2016-08-28")
age(born, check)
```

auc *Compute the area under the curve for two vectors.*

Description

Compute the area under the curve using linear or natural spline interpolation for two vectors where one corresponds to the x values and the other corresponds to the y values.

Usage

```
auc(
  x,
  y,
  from = min(x, na.rm = TRUE),
  to = max(x, na.rm = TRUE),
  type = c("linear", "spline"),
  absolutearea = FALSE,
  subdivisions = 100,
  ...
)
```

Arguments

x	a numeric vector of unique x values.
y	a numeric vector of y values of the same length as x.
from	The value from where to start calculating the area under the curve. Defaults to the smallest x value.
to	The value from where to end the calculation of the area under the curve. Defaults to the greatest x value.
type	The type of interpolation. Defaults to "linear" for area under the curve for linear interpolation. The value "spline" results in the area under the natural cubic spline interpolation.
absolutearea	A logical value that determines if negative areas should be added to the total area under the curve. By default the auc function subtracts areas that have negative y values. Set absolutearea=TRUE to _add_ the absolute value of the negative areas to the total area.
subdivisions	an integer telling how many subdivisions should be used for integrate (for non-linear approximations)
...	additional arguments passed on to approx (for linear approximations). In particular rule can be set to determine how values outside the range of x is handled.

Details

For linear interpolation the auc function computes the area under the curve using the composite trapezoid rule. For area under a spline interpolation, auc uses the splinefun function in combination with the integrate to calculate a numerical integral. The auc function can handle unsorted time values, missing observations, ties for the time values, and integrating over part of the area or even outside the area.

Value

The value of the area under the curve.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

approx, splinefun, integrate

Examples

```
x <- 1:4
y <- c(0, 1, 1, 5)
auc(x, y)

# AUC from 0 to max(x) where we allow for extrapolation
auc(x, y, from=0, rule=2)

# Use value 0 to the left
auc(x, y, from=0, rule=2, yleft=0)

# Use 1/2 to the left
auc(x, y, from=0, rule=2, yleft=.5)

# Use 1/2 to the left with spline interpolation
auc(x, y, from=0, rule=2, yleft=.5)
```

---

bdstat *Danish live births and deaths*

Description

Monthly live births and deaths in Denmark from January 1901 to March 2013.

Format

A data frame with 1356 observations on the following 4 variables.

year a numeric vector giving the month

month a numeric vector giving the year

births a numeric vector. The number of births for the given month and year

dead a numeric vector. The number of deaths for the given month and year

Source

Data were obtained from the StatBank from Danmarks Statistik, see http://www.statbank.dk.

Examples

```
data(bdstat)

plot(bdstat$year + bdstat$month/13, bdstat$birth, type="l")

# Create a table of births
# Remove year 2013 as it is incomplete
btable <- xtabs(births ~ year + month, data=bdstat, subset=(year<2013))

# Compute yearly birth frequencies per month
btable.freq <- prop.table(btable, margin=1)
```

---

bees *Bee data. Number of different types of bees caught.*

Description

Number of different types of bees caught in plates of different colours. There are four locations and within each location there are three replicates consisting of three plates of the three different colours (yellow, white and blue). Data are collected at 5 different dates over the summer season. Only data from one date available until data has been published.

Format

A data frame with 72 observations on the following 7 variables.

Locality a factor with levels `Havreholm` `Kragevig` `Saltrup` `Svaerdborg`. Four different localities in Denmark.

Replicate a factor with levels `A` `B` `C`

Color a factor with levels `Blue` `White` `Yellow`. Colour of plates

Time a factor with levels `july1` `july14` `june17` `june3` `june6`. Data collected at different dates in summer season. Only one day is present in the current data frame until the full data has been released.

Type a factor with levels `Bumblebees` `Solitary`. Type of bee.

Number a numeric vector. The response variable with number of bees catched.

id a numeric vector. The id of the clusters (each containg three plates).

Source

Data were kindly provided by Casper Ingerslev Henriksen, Department of Agricultural Sciences, KU-LIFE. Added by Torben Martinussen <tma@life.ku.dk>

Examples

```
data(bees)
model <- glm(Number ~ Locality + Type*Color,
             family=poisson, data=bees)
```

---

bin *Fast binning of numeric vector into equidistant bins*

Description

Fast binning of numeric vector into equidistant bins

Usage

```
bin(x, width, origin = 0, missinglast = FALSE)
```

Arguments

`x`	A matrix of regressor variables. Must have the same number of rows as the length of y.
`width`	The width of the bins
`origin`	The starting point for the bins. Any number smaller than origin will be disregarded
`missinglast`	Boolean. Should the missing observations be added as a separate element at the end of the returned count vector.

Details

Missing values (NA, Inf, NaN) are added at the end of the vector as the last bin returned if missinglast is set to TRUE

Value

An list with elements counts (the frequencies), origin (the origin), width (the width), missing (the number of missings), and last_bin_is_missing (boolean) telling whether the missinglast is true or not.

Author(s)

Hadley Wickham (from SO: https://stackoverflow.com/questions/13661065/superimpose-histogram-fits-in-one-plot-ggplot) - adapted here by Claus Ekstrøm <claus@rprimer.dk>

Examples

```
set.seed(1)
x <- sample(10, 20, replace = TRUE)
bin(x, 15)
```

---

categorize *A table function to use with magrittr pipes*

Description

Accepts a data frame as input and computes a contingency table for direct use in combination with the magrittr package.

Usage

```
categorize(.data, ...)
```

Arguments

.data	A data frame
...	A formula (as in xtabs) or one or more objects which can be interpreted as factors (including character strings), or a list (or data frame) whose components can be so interpreted.

Details

categorize is a wrapper to xtabs or table such that a data frame can be given as the first argument.

Value

A table (possibly as an xtabs class if a model formula was used)

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
esoph |> categorize(alcgp, agegp)
esoph |> categorize(~ alcgp + agegp)
```

clipit *Copy an object as R-code to the clipboard*

Description

Copies an R object to the clipboard so it can be pasted in elsewhere.

Usage

```
clipit(x)
```

Arguments

x	object to copy

Details

Returns nothing but will place the object in the clipboard

Value

Nothing but will put the R object into the clipboard as a side effect

Author(s)

Jonas LindeLøv posted on twitter. Copied shamelessly by Claus Ekstrom <claus@rprimer.dk>

Examples

```
## Not run:
clipit(mtcars$mpg)

## End(Not run)
```

clotting *Blood clotting for 158 rats*

Description

Blood clotting activity (PCA) is measured for 158 Norway rats from two locations just before (baseline) and four days after injection of an anticoagulant (bromadiolone). Normally this would cause reduced blood clotting after 4 days compared to the baseline, but these rats are known to possess anticoagulent resistence to varying extent. The purpose is to relate anticoagulent resistence to gender and location and perhaps weight. Dose of injection is, however, admistered according to weight and gender.

Format

A data frame with 158 observations on the following 6 variables.

rat a numeric vector

locality a factor with levels `Loc1` `Loc2`

sex a factor with levels `F` `M`

weight a numeric vector

PCA0 a numeric vector with percent blood clotting activity at baseline

PCA4 a numeric vector with percent blood clotting activity on day 4

Source

Ann-Charlotte Heiberg, project at The Royal Veterinary and Agricultural University, 1999.
Added by Ib M. Skovgaard <ims@life.ku.dk>

Examples

```
data(clotting)
dim(clotting)
head(clotting)
day0= transform(clotting, day=0, pca=PCA0)
day4= transform(clotting, day=4, pca=PCA4)
day.both= rbind(day0,day4)
m1= lm(pca ~ rat + day*locality + day*sex, data=day.both)
anova(m1)
summary(m1)
m2= lm(pca ~ rat + day, data=day.both)
anova(m2)
## Log transformation suggested.
## Random effect of rat.
## maybe str(clotting) ; plot(clotting) ...
```

---

cmd *Correlation matrix distance*

Description

Computes the correlation matrix distance between two correlation matrices

Usage

```
cmd(x, y)
```

Arguments

x	First correlation matrix
y	Second correlation matrix

Value

Returns the correlation matrix distance, which is a value between 0 and 1. The correlation matrix distance becomes zero for equal correlation matrices and unity if they differ to a maximum extent.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Herdin, M., and Czink, N., and Ozcelik, H., and Bonek, E. (2005). *Correlation matrix distance, a meaningful measure for evaluation of non-stationary mimo channels*. IEEE VTC.

Examples

```
m1 <- matrix(rep(1, 16), 4)
m2 <- matrix(c(1, 0, .5, .5, 0, 1, .5, .5, .5, .5, 1, .5, .5, .5, .5, 1), 4)
m3 <- matrix(c(1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1), 4)
cmd(m1, m1)
cmd(m1, m2)
cmd(m2, m3)
```

---

col.alpha *Add and set alpha channel for RGB color*

Description

Add and set alpha channel

Usage

```
col.alpha(col, alpha = 1)
```

Arguments

col	a vector of RGB color(s)
alpha	numeric value between 0 and 1. Zero results fully transparent and 1 means full opacity

Details

This function adds and set an alpha channel to a RGB color

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Ekstrom, CT (2011) *The R Primer*.

Examples

```
newcol <- col.alpha("blue", .5)
```

`col.shade` *Shade an RGB color*

Description

Shades an RBG color

Usage

```
col.shade(col, shade = 0.5)
```

Arguments

`col`	a vector of RGB color(s)
`shade`	numeric value between 0 and 1. Zero means no change and 1 results in black

Details

This function shades an RGB color and returns the shaded RGB color (with alpha channel added)

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Ekstrom, CT (2011) *The R Primer*.

Examples

```
newcol <- col.shade("blue")
```

col.tint *Tint an RGB color*

Description

Tints an RBG color

Usage

```
col.tint(col, tint = 0.5)
```

Arguments

col	a vector of RGB color(s)
tint	numeric value between 0 and 1. Zero results in white and 1 means no change

Details

This function tints an RGB color and returns the tinted RGB color (with alpha channel added)

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Ekstrom, CT (2011) *The R Primer*.

Examples

```
newcol <- col.tint("blue")
```

colCumSum *Apply cumsum to each column of matrix*

Description

Fast computation of apply(m, 2, cumsum)

Usage

```
colCumSum(m)
```

Arguments

m	A matrix

Value

A matrix the same size as m with the column-wise cumulative sums.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
# Generate a 100 by 10000 matrix
x <- matrix(rnorm(100*10000), nrow=100)
result <- colCumSum(x)
```

---

common.shared *Compute a common shared environment matrix*

Description

Compute the common shared environment matrix for a set of related subjects. The function is generic, and can accept a pedigree, or pedigreeList as the first argument.

Usage

```
common.shared(id, ...)

## S3 method for class 'pedigreeList'
common.shared(id, ...)

## S3 method for class 'pedigree'
common.shared(id, ...)
```

Arguments

id	either a pedigree object or pedigreeList object
...	Any number of optional arguments. Not used at the moment

Details

When called with a pedigreeList, i.e., with multiple families, the routine will create a block-diagonal-symmetric 'bdsmatrix' object. Since the [i,j] value of the result is 0 for any two unrelated individuals i and j and a 'bdsmatix' utilizes sparse representation, the resulting object is often orders of magnitude smaller than an ordinary matrix. When called with a single pedigree and ordinary matrix is returned.

Value

a matrix of shared environment coefficients

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

pedigree, kinship,

Examples

```
library(kinship2)
test1 <- data.frame(id  =c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14),
                    mom =c(0, 0, 0, 0, 2, 2, 4, 4, 6,  2,  0,  0, 12, 13),
                    dad =c(0, 0, 0, 0, 1, 1, 3, 3, 3,  7,  0,  0, 11, 10),
                    sex =c(1, 2, 1, 2, 1, 2, 1, 2, 1,  1,  1,  2,  2,  2))
tped <- with(test1, pedigree(id, dad, mom, sex))
common.shared(tped)
```

---

conditional_rowMeans *Form row means conditional on number of non-missing*

Description

Form row means for multiple vectors, numeric arrays (or data frames) conditional on the number of non-missing observations. NA is returned unless a minimum number of observations is observed.

Usage

```
conditional_rowMeans(..., minobs = 1L)
```

Arguments

...	a series of numeric vectors, arrays, or data frames that have can be combined with cbind
minobs	an integer stating the minimum number of non-NA observations necessary to compute the row mean. Defaults to 1.

Value

A numeric vector containing the row sums or NA if not enough non-NA observations are present

Examples

```
conditional_rowMeans(1:5, c(1:4, NA), c(1:3, NA, NA))
conditional_rowMeans(1:5, c(1:4, NA), c(1:3, NA, NA), minobs=0)
conditional_rowMeans(1:5, c(1:4, NA), c(1:3, NA, NA), minobs=2)
```

cumsumbinning *Binning based on cumulative sum with reset above threshold*

Description

Fast binning of cumulative vector sum with new groups when the sum passes a threshold or the group size becomes too large

Usage

```
cumsumbinning(x, threshold, cutwhenpassed = FALSE, maxgroupsize = NULL)
```

Arguments

x	A matrix of regressor variables. Must have the same number of rows as the length of y.
threshold	The value of the threshold that the cumulative group sum must not cross OR the threshold that each group sum must pass (when the argument cuwhatpassed is set to TRUE).
cutwhenpassed	A boolean. Should the threshold be the upper limit of the group sum (the default) or the value that each group sum needs to pass (when set to TRUE).
maxgroupsize	An integer that defines the maximum number of elements in each group. NAs count as part of each group but do not add to the group sum. NULL (the default) corresponds to no group size limits.

Details

Missing values (NA, Inf, NaN) are completely disregarded and pairwise complete cases are used f

Value

An integer vector giving the group indices

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
set.seed(1)
x <- sample(10, 20, replace = TRUE)
cumsumbinning(x, 15)
cumsumbinning(x, 15, 3)

x <- c(3, 4, 5, 12, 1, 5, 3)
cumsumbinning(x, 10)
cumsumbinning(x, 10, cutwhenpassed=TRUE)
```

dCor *Fast distance correlation matrix*

Description

Fast computation of the distance correation matrix between two matrices with the same number of rows. Note that this is not the same as the correlation matrix distance that can be computed with the cmd function.

Usage

```
dCor(x, y)
```

Arguments

x	A matrix with dimensions n*k.
y	A matrix with dimensions n*l.

Value

A number between 0 and 1 representing the distance covariance between x and y

Author(s)

Claus Ekstrom <claus@rprimer.dk>

dCov *Fast distance covariance matrix*

Description

Fast computation of the distance covariance between two matrices with the same number of rows.

Usage

```
dCov(x, y)
```

Arguments

x	A matrix with dimensions n*k.
y	A matrix with dimensions n*l.

Value

A number representing the distance covariance between x and y

Author(s)

Claus Ekstrom <claus@rprimer.dk>

`drop1.geeglm` *Drop All Possible Single Terms to a geeglm Model Using Wald or Score Test*

Description

Compute all the single terms in the scope argument that can dropped from the model, and compute a table of the corresponding Wald test statistics.

Usage

```
## S3 method for class 'geeglm'
drop1(
  object,
  scope,
  test = c("Wald", "none", "score", "sasscore"),
  method = c("robust", "naive", "sandwich"),
  ...
)
```

Arguments

`object`	a fitted object of class geese.
`scope`	a formula giving the terms to be considered for adding or dropping.
`test`	the type of test to include.
`method`	Indicates which method is used for computing the standard error. `robust` is the default and corresponds to the modified sandwich estimator. `naive` is the classical naive cariance estimate. `sandwich` is an alias for `robust`.
`...`	other arguments. Not currently used

Value

An object of class "anova" summarizing the differences in fit between the models.

Author(s)

Claus Ekstrom <claus@ekstroem.dk>

See Also

drop1, geeglm, geese

Examples

```
library(geepack)
data(ohio)
fit <- geeglm(resp ~ age + smoke + age:smoke, id=id, data=ohio,
             family=binomial, corstr="exch", scale.fix=TRUE)
drop1(fit)
```

`drop1.geem` *Drop All Possible Single Terms to a geem Model Using Wald or Score Test*

Description

Compute all the single terms in the scope argument that can dropped from the model, and compute a table of the corresponding Wald test statistics.

Usage

```
## S3 method for class 'geem'
drop1(
  object,
  scope,
  test = c("Wald", "none", "score", "sasscore"),
  method = c("robust", "naive", "sandwich"),
  ...
)
```

Arguments

`object`	a fitted object of class geese.
`scope`	a formula giving the terms to be considered for adding or dropping.
`test`	the type of test to include.
`method`	Indicates which method is used for computing the standard error. `robust` is the default and corresponds to the modified sandwich estimator. `naive` is the classical naive cariance estimate. `sandwich` is an alias for `robust`.
`...`	other arguments. Not currently used

Value

An object of class "anova" summarizing the differences in fit between the models.

Author(s)

Claus Ekstrom <claus@ekstroem.dk>

See Also

drop1, geem

Examples

```
library(geeM)
library(geepack)
data(ohio)
## Not run:
fit <- geem(resp ~ age + smoke + age:smoke, id=id, data=ohio,
            family="binomial", corstr="exch", scale.fix=TRUE)
drop1(fit)

## End(Not run)
```

---

earthquakes *Earthquakes in 2015*

Description

Information on earthquakes worldwide in 2015 with a magnitude greater than 3 on the Richter scale. The variables are just a subset of the variables available at the source

Format

A data frame with 19777 observations on the following 22 variables.

time a factor with time of the earthquake

`latitude` a numeric vector giving the decimal degrees latitude. Negative values for southern latitudes

`longitude` a numeric vector giving the decimal degrees longitude. Negative values for western longitudes

`depth` Depth of the event in kilometers

`mag` The magnitude for the event

`place` a factor giving a textual description of named geographic region near to the event.

`type` a factor with levels `earthquake` `mining` `explosion` `rock` `burst`

Source

https://www.usgs.gov/programs/earthquake-hazards

Examples

```
data(earthquakes)
with(earthquakes, place[which.max(mag)])
```

expand_table *Expand table or matrix to data frame*

Description

Expands a contingency table to a data frame where each observation in the table becomes a single observation in the data frame with corresponding information for each for each combination of the table dimensions.

Usage

```
expand_table(x)
```

Arguments

x	A table or matrix

Value

A data frame with the table or matrix expanded

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
expand_table(diag(3))
m <- matrix(c(2, 1, 3, 0, 0, 2), 3)
expand_table(m)
result <- expand_table(UCBAdmissions)
head(result)

# Combine into table again
xtabs(~Admit + Gender + Dept, data=result)
```

extended.shared *Compute a common shared environment matrix*

Description

Compute the common shared environment matrix for a set of related subjects. The function is generic, and can accept a pedigree, or pedigreeList as the first argument.

Usage

```
extended.shared(id, rho = 1, theta = 1, ...)

## S3 method for class 'pedigreeList'
extended.shared(id, rho = 1, theta = 1, ...)

## S3 method for class 'pedigree'
extended.shared(id, rho = 1, theta = 1, ...)
```

Arguments

id	either a pedigree object or pedigreeList object
rho	The correlation between spouses
theta	The partial path coefficient from parents to offspring
...	Any number of optional arguments. Not used at the moment

Details

When called with a pedigreeList, i.e., with multiple families, the routine will create a block-diagonal-symmetric 'bdsmatrix' object. Since the [i,j] value of the result is 0 for any two unrelated individuals i and j and a 'bdsmatix' utilizes sparse representation, the resulting object is often orders of magnitude smaller than an ordinary matrix. When called with a single pedigree and ordinary matrix is returned.

Value

a matrix of shared environment coefficients

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

pedigree, kinship,

Examples

```
library(kinship2)
test1 <- data.frame(id  =c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14),
                    mom =c(0, 0, 0, 0, 0, 2, 2, 4, 0,  6,  8,  0, 10, 11),
                    dad =c(0, 0, 0, 0, 0, 1, 1, 3, 0,  5,  7,  0,  9, 12),
                    sex =c(1, 2, 1, 2, 1, 2, 1, 2, 1,  2,  2,  1,  2,  2))

tped <- with(test1, pedigree(id, dad, mom, sex))
extended.shared(tped)
```

fac2num *Convert factor to numeric vector*

Description

Converts the factor labels to numeric values and returns the factor as a numeric vector

Usage

```
fac2num(x)
```

Arguments

x	A factor

Details

Returns a vector of numeric values. Elements in the input factor that cannot be converted to numeric will produce NA.

Value

Returns a numeric vector of the same length as x

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
f <- factor(c(1,2,1,3,2,1,2,3,1))
fac2num(f)
```

feature.test *Inference for features identified by the Lasso*

Description

Performs randomization tests of features identified by the Lasso

Usage

```
feature.test(
  x,
  y,
  B = 100,
  type.measure = "deviance",
  s = "lambda.min",
  keeplambda = FALSE,
  olsestimates = TRUE,
  penalty.factor = rep(1, nvars),
  alpha = 1,
  control = list(trace = FALSE, maxcores = 24),
  ...
)
```

Arguments

x	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	quantitative response variable of length nobs
B	The number of randomizations used in the computations
type.measure	loss to use for cross-validation. See cv.glmnet for more information
s	Value of the penalty parameter 'lambda' at which predictions are required. Default is the entire sequence used to create the model. See coef.glmnet for more information
keeplambda	If set to TRUE then the estimated lambda from cross validation from the original dataset is kept and used for evaluation in the subsequent randomization datasets. This reduces computation time substantially as it is not necessary to perform cross validation for each randomization. If set to a value then that value is used for the value of lambda. Defaults to FALSE
olsestimates	Logical. Should the test statistic be based on OLS estimates from the model based on the variables selected by the lasso. Defaults to TRUE. If set to FALSE then the coefficients from the lasso is used as test statistics.
penalty.factor	a vector of weights used for adaptive lasso. See glmnet for more information.
alpha	The elasticnet mixing parameter. See glmnet for more information.
control	A list of options that control the algorithm. Currently trace is a logical and if set to TRUE then the function produces more output. maxcores sets the maximum number of cores to use with the parallel package
...	Other arguments passed to glmnet

Value

Returns a list of 7 variables:

p.full	The p-value for the test of the full set of variables selected by the lasso (based on the OLS estimates)

ols.selected	A vector of the indices of the non-zero variables selected by glmnet sorted from (numerically) highest to lowest based on their ols test statistic.
p.maxols	The p-value for the maximum of the OLS test statistics
lasso.selected	A vector of the indices of the non-zero variables selected by glmnet sorted from (numerically) highest to lowest based on their absolute lasso coefficients.
p.maxlasso	The p-value for the maximum of the lasso test statistics
lambda.orig	The value of lambda used in the computations
B	The number of permutations used

Author(s)

Claus Ekstrom <ekstrom@sund.ku.dk> and Kasper Brink-Jensen <kbrink@life.ku.dk>

References

Brink-Jensen, K and Ekstrom, CT 2014. *Inference for feature selection using the Lasso with high-dimensional data*. https://arxiv.org/abs/1403.4296

See Also

glmnet

Examples

```
# Simulate some data
x <- matrix(rnorm(30*100), nrow=30)
y <- rnorm(30, mean=1*x[,1])

# Make inference for features
## Not run:
feature.test(x, y)

## End(Not run)
```

---

filldown *Fill down NA with the last observed observation*

Description

Fill down missing values with the latest non-missing value

Usage

```
filldown(x)
```

Arguments

x	A vector

Value

A vector or list with the NA's replaced by the last observed value.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
a <- c(1:5, "Howdy", NA, NA, 2:3, NA)
filldown(a)
filldown(c(NA, NA, NA, 3:5))
```

---

founder.shared *Compute a common shared environment matrix*

Description

Compute the common shared environment matrix for a set of related subjects. The function is generic, and can accept a pedigree, or pedigreeList as the first argument.

Usage

```
founder.shared(id, ...)

## S3 method for class 'pedigreeList'
founder.shared(id, ...)

## S3 method for class 'pedigree'
founder.shared(id, ...)
```

Arguments

id	either a pedigree object or pedigreeList object
...	Any number of optional arguments. Not used at the moment

Details

When called with a pedigreeList, i.e., with multiple families, the routine will create a block-diagonal-symmetric 'bdsmatrix' object. Since the [i,j] value of the result is 0 for any two unrelated individuals i and j and a 'bdsmatix' utilizes sparse representation, the resulting object is often orders of magnitude smaller than an ordinary matrix. When called with a single pedigree and ordinary matrix is returned.

Value

a matrix of shared environment coefficients

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

pedigree, kinship,

Examples

```
library(kinship2)
test1 <- data.frame(id  =c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14),
                    mom =c(0, 0, 0, 0, 2, 2, 4, 4, 6,  2,  0,  0, 12, 13),
                    dad =c(0, 0, 0, 0, 1, 1, 3, 3, 3,  7,  0,  0, 11, 10),
                    sex =c(1, 2, 1, 2, 1, 2, 1, 2, 1,  1,  1,  2,  2,  2))
tped <- with(test1, pedigree(id, dad, mom, sex))
founder.shared(tped)
```

---

geekin *Fit a generalized estimating equation (GEE) model with fixed additive correlation structure*

Description

The geekin function fits generalized estimating equations but where the correlation structure is given as linear function of (scaled) fixed correlation structures.

Usage

```
geekin(
  formula,
  family = gaussian,
  data,
  weights,
  subset,
  id,
  na.action,
  control = geepack::geese.control(...),
  varlist,
  ...
)
```

Arguments

formula	See corresponding documentation to glm.
family	See corresponding documentation to glm.
data	See corresponding documentation to glm.
weights	See corresponding documentation to glm.
subset	See corresponding documentation to glm.
id	a vector which identifies the clusters. The length of id should be the same as the number of observations. Data must be sorted so that observations on a cluster are contiguous rows for all entities in the formula. If not the function will give an error
na.action	See corresponding documentation to glm.
control	See corresponding documentation to glm.
varlist	a list containing one or more matrix or bdsmatrix objects that represent the correlation structures
...	further arguments passed to or from other methods.

Details

The geekin function is essentially a wrapper function to geeglm. Through the varlist argument, it allows for correlation structures of the form

R = sum_i=1^k alpha_i R_i

where alpha_i are(nuisance) scale parameters that are used to scale the off-diagonal elements of the individual correlation matrices, R_i.

Value

Returns an object of type geeglm.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

lmekin, geeglm

Examples

```
# Get dataset
library(kinship2)
library(mvtnorm)
data(minnbreast)

breastpeda <- with(minnbreast[order(minnbreast$famid), ], pedigree(id,
                  fatherid, motherid, sex,
```

```
                    status=(cancer& !is.na(cancer)), affected=proband,
                    famid=famid))

set.seed(10)

nfam <- 6
breastped <- breastpeda[1:nfam]

 # Simulate a response

# Make dataset for lme4
df <- lapply(1:nfam, function(xx) {
            as.data.frame(breastped[xx])
            })

mydata <- do.call(rbind, df)
mydata$famid <- rep(1:nfam, times=unlist(lapply(df, nrow)))

y <- lapply(1:nfam, function(xx) {
            x <- breastped[xx]
            rmvtnorm.pedigree(1, x, h2=0.3, c2=0)
            })
yy <- unlist(y)

library(geepack)

geekin(yy ~ 1, id=mydata$famid, varlist=list(2*kinship(breastped)))

# lmekin(yy ~ 1 + (1|id), data=mydata, varlist=list(2*kinship(breastped)),method="REML")
```

---

gkgamma *Goodman-Kruskal's gamma statistic for a two-dimensional table*

Description

Compute Goodman-Kruskal's gamma statistic for a two-dimensional table of ordered categories

Usage

```
gkgamma(x, conf.level = 0.95)
```

Arguments

x	A matrix or table representing the two-dimensional ordered contingency table of observations
conf.level	Level of confidence interval

Value

A list with class `htest` containing the following components:

`statistic`	the value the test statistic for testing no association
`p.value`	the p-value for the test
`estimate`	the value the gamma estimate
`conf.int`	the confidence interval for the gamma estimate
`method`	a character string indicating the type of test performed
`data.name`	a character string indicating the name of the data input
`observed`	the observed counts
`s0`	the SE used when computing the test statistics
`s1`	the SE used when computing the confidence interval

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Goodman, Leo A. and Kruskal, William H. (1954). "Measures of Association for Cross Classifications". Journal of the American Statistical Association 49 (268): 732-764.

See Also

`chisq.test`

Examples

```
# Data from the Glostrup study comparing smoking to overall health in males
smoke <- matrix(c(16, 15, 13, 10, 1, 73, 75, 59, 81, 29, 6, 6, 7, 17, 3, 1, 0, 1, 3, 1), ncol=4)
colnames(smoke) <- c("VGood", "Good", "Fair", "Bad") # General health status
rownames(smoke) <- c("Never", "No more", "1-14", "15-24", "25+")  # Smoke amount
gkgamma(smoke)
chisq.test(smoke)
```

---

greenland *Average yearly summer air temperature for Tasiilaq, Greenland*

Description

Average yearly summer (June, July, August) air temperature for Tasiilaq, Greenland

Format

A data frame with 51 observations on the following 2 variables.

year year

airtemp average air temperature (degrees Celcius)

Source

Data provided by Sebastian Mernild.
Originally obtained from http://www.dmi.dk/dmi/index/gronland/vejrarkiv-gl.htm.
Added by Claus Ekstrom <ekstrom@life.ku.dk>

References

Aktuelt Naturvidenskab september 2010.
http://aktuelnaturvidenskab.dk/fileadmin/an/nr-4/an4_2010gletscher.pdf

Examples

```
data(greenland)
model <- lm(airtemp ~ year, data=greenland)
plot(greenland$year, greenland$airtemp, xlab="Year", ylab="Air temperature")
abline(model, col="red")
```

---

happiness *Happiness score and tax rates for 148 countries*

Description

Dataset on subjective happiness, tax rates, population sizes, continent, and major religion for 148 countries

Format

A data frame with 148 observations on the following 6 variables.

country a factor with 148 levels that contain the country names

happy a numeric vector with the average subject happiness score (on a scale from 0-10)

tax a numeric vector showing the tax revenue as percentage of GDP

religion a factor with levels `Buddhist` `Christian` `Hindu` `Muslim` `None` or `Other`

continent a factor with levels `AF`, `AS`, `EU`, `NA`, `OC`, `SA`, corresponding to the continents Africa, Asia, Europe, North America, Ocenaia, South American, respectively

population a numeric vector showing the population (in millions)

Source

Data collected by Ellen Ekstroem.
Population sizes are from Wikipedia per August 2nd, 2012 https://en.wikipedia.org/wiki/List_of_countries_by_population
Major religions are from Wikipedia per August 2nd, 2012 https://en.wikipedia.org/wiki/Religions_by_country
Tax rates are from Wikipedia per August 2nd, 2012 https://en.wikipedia.org/wiki/List_of_countries_by_tax_revenue_as_percentage_of_GDP
Average happiness scores are from "Veenhoven, R. Average happiness in 148 nations 2000-2009, World Database of Happiness, Erasmus University Rotterdam, The Netherlands". Assessed on August 2nd, 2012 at: https://worlddatabaseofhappiness-archive.eur.nl/hap_nat/findingreports/RankReport_AverageHappiness.php

Examples

```
data(happiness)
with(happiness, symbols(tax, happy, circles=sqrt(population)/8, inches=FALSE, bg=continent))

#
# Make a prettier image with transparent colors
#

newcols <- rgb(t(col2rgb(palette())),
               alpha=100, maxColorValue=255)

with(happiness, symbols(tax, happy, circles=sqrt(population)/8,
                inches=FALSE, bg=newcols[continent],
                xlab="Tax (% of GDP)", ylab="Happiness"))

#
# Simple analysis
#
res <- lm(happy ~ religion + population + tax:continent, data=happiness)
summary(res)
```

---

ht *Show the head and tail of an object*

Description

Show both the head and tail of an R object

Usage

```
ht(x, n = 6L, m = n, returnList = FALSE, ...)
```

Arguments

x	The object to show
n	The number of elements to list for the head
m	The number of elements to list for the tail
returnList	Logical. Should the result be returned as a list
...	additional arguments passed to functions (not used at the moment)

Details

This function does no error checking and it is up to the user to ensure that the input is indeed symmetric, positive-definite, and a matrix.

Value

NULL unless returnList is set to TRUE in which case a list is returned

Author(s)

Claus Ekstrom, <claus@rprimer.dk>.

Examples

```
ht(trees)
ht(diag(20))
ht(1:20)
ht(1:20, returnList=TRUE)
```

---

hwe_frequencies *Fast estimation of allele and genotype frequencies under Hardy-Weinberg equilibrium*

Description

Alleles are assumed to be numerated from 1 and up with no missing label. Thus if the largest value in either allele1 or allele2 is K then we assume that there can be at least K possible alleles. Genotypes are sorted such the the smallest allele comes first, i.e., 2x1 -> 1x2, and 2x3 -> 2x3

Usage

```
hwe_frequencies(allele1, allele2, min_alleles = 0L)
```

Arguments

allele1	An integer vector (starting with values 1 upwards) of first alleles
allele2	An integer vector (starting with values 1 upwards) of second alleles
min_alleles	A minimum number of unique alleles available

Value

A list with three variables: allele_freq for estimated allele frequencies, genotype_freq for estimated genotype_frequencies (under HWE assumption), obs_genotype is the frequency of the genotypes, available_genotypes is the number of available genotypes used for the estimation, and unique_alleles is the number of unique alleles (matches the length of allele_freq)

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
al1 <- sample(1:5, size=1000, replace=TRUE, prob=c(.4, .2, .2, .1, .1))
al2 <- sample(1:5, size=1000, replace=TRUE, prob=c(.4, .2, .2, .1, .1))
hwe_frequencies(al1, al2)
```

---

icecreamads *Ice cream consumption and advertising*

Description

The impact of advertizing impact, temperature, and price on ice cream consumption

Format

A data frame with 30 observations on the following 4 variables.

Price a numeric vector character vector giving the standardized price

Temperature temperature in degrees Fahrenheit

Consumption a factor with levels `1_low` `2_medium` `3_high`

Advertise a factor with levels `posters` `radio` `television`

Source

Unknown origin

Examples

```
data("icecreamads")
```

`ks_cumtest` *Kolmogorov-Smirnov goodness of fit test for cumulative discrete data*

Description

Kolmogorov-Smirnov goodness of fit test for cumulative discrete data.

Usage

```
ks_cumtest(x, B = 10000L, prob = NULL)
```

Arguments

`x`	A vector representing the contingency table.
`B`	The number of simulations used to compute the p-value.
`prob`	A positive vector of the same length as x representing the distribution under the null hypothesis. It will be scaled to sum to 1. If NULL (the default) then a uniform distribution is assumed.

Details

The name of the function might change in the future so keep that in mind!

Simulation is done by random sampling from the null hypothesis.

Value

A list of class "htest" giving the simulation results.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
x <- 1:6
ks_cumtest(x)
```

kwdata *Non-parametric Kruskal Wallis data example*

Description

Artificial dataset to show that the p-value obtained for the Kruskal Wallis is only valid _after_ the distributional form has been checked to be the same for all groups.

Format

An artificial data frame with 18 observations in each of three groups.

x measurements for group 1

y measurements for group 2

z measurements for group 3

Source

Data example found on the internet

Examples

```
data(kwdata)
newdata <- stack(kwdata)
kruskal.test(values ~ ind, newdata)
```

lifeexpect *Estimated life expectancy for Danish newborns*

Description

The estimated life expectancy for newborn Danes split according to gender.

Format

A data frame with 70 observations on the following 3 variables.

`year` a character vectorgiving the calendar interval on which the estimation was based.

`male` a numeric vectorLife expectancy for males (in years).

`female` a numeric vectorLife expectancy for females (in years)

`myear` a numeric vectorThe midpoint of the year interval

Source

Data collected from Danmarks Statistik. See https://www.dst.dk/en for more information.

Examples

```
data(lifeexpect)
plot(lifeexpect$myear, lifeexpect$male)
```

`loadRData` *Load and extract object from RData file*

Description

Loads and extracts an object from an RData file

Usage

```
loadRData(filename)
```

Arguments

`filename`	The path to the RData file

Details

Returns an R object

Value

An R object

Author(s)

ricardo (from GitHub)

See Also

`load`

Examples

```
## Not run:
  d <- loadRDData("~/blah/ricardo.RData")

## End(Not run)
```

`lower.tri.vector` *Split Matrix by Clusters and Return Lower Triangular Parts as Vector*

Description

Split a matrix into block diagonal sub matrices according to clusters and combine the lower triangular parts into a vector

Usage

```
lower.tri.vector(x, cluster = rep(1, nrow(x)), diag = FALSE)
```

Arguments

`x`	a square matrix
`cluster`	numeric or factor. Is used to identify the sub-matrices of `x` from which the lower triangular parts are extracted. Defaults to the full matrix.
`diag`	logical. Should the diagonal be included?

Value

Returns a numeric vector containing the elements of the lower triangular sub matrices.

Author(s)

Claus Ekstrom <claus@ekstroem.dk>

See Also

`lower.tri`

Examples

```
m <- matrix(1:64, ncol=8)
cluster <- c(1, 1, 1, 1, 2, 2, 3, 3)
lower.tri.vector(m, cluster)
```

matched *Flu hospitalization*

Description

Researchers in a Midwestern county tracked flu cases requiring hospitalization in those residents aged 65 and older during a two-month period one winter. They matched each case with 2 controls by sex and age (150 cases, 300 controls). They used medical records to determine whether cases and controls had received a flu vaccine shot and whether they had underlying lung disease. They wanted to know whether flu vaccination prevents hospitalization for flu (severe cases of flu). Underlying lung disease is a potential confounder.

Format

A data frame with 450 observations on the following 4 variables.

`id` a numeric vector

`iscase` a factor with levels `Control` `Case`

`vaccine` a factor with levels `Not` `Vaccinated`

`lung` a factor with levels `None` `Disease`

Source

Modified from: Stokes, Davis, Koch (2000). "Categorical Data Analysis Using the SAS System," Chapter 10.

Examples

```
data(matched)
```

maximum_subarray *Fast computation of maximum sum subarray*

Description

Fast computation of the maximum subarray sum of a vector using Kadane's algorithm. The implementation handles purely negative numbers.

Usage

```
maximum_subarray(x)
```

Arguments

`x`	A vector

Value

A list with three elements: sum (the maximum subarray sum), start (the starting index of the subarray) and end (the ending index of the subarray)

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
maximum_subarray(1:4)

maximum_subarray(c(-2, 1, -3, 4, -1, 2, 1, -5, 4))

maximum_subarray(rnorm(100000))
```

---

mfastLmCpp *Fast marginal simple regresion analyses*

Description

Fast computation of simple regression slopes for each predictor represented by a column in a matrix

Usage

```
mfastLmCpp(y, x, addintercept = TRUE)
```

Arguments

y	A vector of outcomes.
x	A matrix of regressor variables. Must have the same number of rows as the length of y.
addintercept	A logical that determines if the intercept should be included in all analyses (TRUE) or not (FALSE)

Details

No error checking is done

Value

A data frame with three variables: coefficients, stderr, and tstat that gives the slope estimate, the corresponding standard error, and their ratio for each column in x.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
## Not run:
  // Generate 100000 predictors and 100 observations
  x <- matrix(rnorm(100*100000), nrow=100)
  y <- rnorm(100, mean=x[,1])
  mfastLmCpp(y, x)


## End(Not run)
```

`monte_carlo_chisq_test`

Two-sided table test with fixed margins

Description

Monte Carlo test in a two-way contingency table with the total number of observations fixed, row margin fixed, or both margins fixed.

Usage

```
monte_carlo_chisq_test(x, margin = c("N", "rows", "both"), B = 100000L)
```

Arguments

`x`	A matrix representing the contingency table.
`margin`	A string that determines which margin is fixed: Either "N" for the total number of observations (the default), "rows" for fixed row sums, and "both" for simultaneously fixed row and column sums.
`B`	The number of simulations used to compute the p-value.

Details

Simulation is done by random sampling from the set of all tables with given marginal(s), and works only if the relevant marginal(s) are strictly positive. Continuity correction is never used, and the statistic is quoted without it.

Value

A list of class "htest" giving the simulation results.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
m <- matrix(c(12, 4, 8, 6), 2)
chisq.test(m)
chisq.test(m, correct=FALSE)
monte_carlo_chisq_test(m)

fisher.test(m)
monte_carlo_chisq_test(m, margin="both")

m2 <- matrix(c(9, 3, 3, 7), 2)
monte_carlo_chisq_test(m, margin="N")
monte_carlo_chisq_test(m, margin="both")
```

---

nh4 *Ammonia nitrogen found in river*

Description

Monthly levels of ammonia nitrogen in a river over two years

Format

A data frame with 120 observations on the following 3 variables.

nh4 The ammonia nitrogen levels (mg/l). A value of zero corresponds to a censoring, but it really is censored at <0.01

cens A logical vector indicating if the value was censored

year The year

Source

Found on the internet and partly simulated

Examples

```
data(nh4)
```

ordered.clusters *Check if unique elements of a vector appear in contiguous clusters*

Description

ordered.clusters determines if identical elements of a vector appear in contiguous clusters, and returns TRUE if the do and FALSE otherwise.

Usage

```
ordered.clusters(id)
```

Arguments

id	a vector

Value

The function returns TRUE if the elements appear in contiguous clusters and FALSE otherwise

Author(s)

Claus Ekstrom <claus@ekstroem.dk> with suggestions from Peter Dalgaard.

See Also

duplicated

Examples

```
x <- c(1, 1, 1, 2, 2, 3, 4, 1, 5, 5, 5)
ordered.clusters(x)
ordered.clusters(sort(x))
ordered.clusters(x[order(x)])
```

pairwise.cor.test *Pairwise Tests for Association/Correlation Between Paired Samples*

Description

Calculate pairwise correlations between group levels with corrections for multiple testing.

Usage

```
pairwise.cor.test(
  x,
  g,
  p.adjust.method = p.adjust.methods,
  method = c("pearson", "kendall", "spearman"),
  ...
)
```

Arguments

x	response vector.
g	grouping vector or factor.
p.adjust.method	method for adjusting p values (see p.adjust). Can be abbreviated.
method	string argument to set the method to compute the correlation. Possibilities are "pearson" (the default), "kendall", and "spearman"
...	additional arguments passed to cor.test.

Details

Note that correlation tests require that the two vectors examined are of the same length. Thus, if the grouping defines groups of varying lengths then the specific correlation is not computed and a NA is returned instead. The adjusted p values are only based on the actual correlation that are computed. Extra arguments that are passed on to cor.test may or may not be sensible in this context.

Value

Object of class pairwise.htest

Examples

```
attach(airquality)
Month <- factor(Month, labels = month.abb[5:9])
pairwise.cor.test(Ozone, Month)
pairwise.cor.test(Ozone, Month, p.adj = "bonf")
detach()
```

---

pairwise_combination_indices

Compute all pairwise combinations of indices

Description

Fast computation of indices of all pairwise element of a vector of length n.

Usage

```
pairwise_combination_indices(n, self = FALSE)
```

Arguments

n	A number giving the number of elements to create all pairwise indices from
self	A logical that determines whether a column should also be multiplied by itself.

Details

Note that the output order of columns corresponds to the order of the columns in x. First column 1 is multiplied with each of the other columns, then column 2 with the remaining columns etc.

Value

A matrix with n*(n+1)/2 rows (if self=TRUE) or n*(n-1)/2 rows (if self=FALSE, the default) and two columns gicing all possible combinations of indices.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
pairwise_combination_indices(3)
pairwise_combination_indices(4, self=TRUE)
```

---

pairwise_distance_correlation

Compute pairwise distance correlation metrics of each column to a vector

---

Description

Fast computation of pairwise distance correlations.

Usage

```
pairwise_distance_correlation(x, y)
```

Arguments

x	A matrix. The number of rows should match the length of the vector y
y	A vector

Details

Note: To get the same result as from the energy package you need to take the square root of the results here.

Value

A vector with the same length as the number of columns in x. Each element contains the pairwise distance correlation to y.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
y <- rnorm(100)
x <- matrix(rnorm(100 * 10), ncol = 10)
pairwise_distance_correlation(x, y)
```

---

pairwise_Schur_product

Compute Schur products (element-wise) of all pairwise combinations of columns in matrix

---

Description

Fast computation of all pairwise element-wise column products of a matrix.

Usage

```
pairwise_Schur_product(x, self = FALSE)
```

Arguments

x	A matrix with dimensions r*c.
self	A logical that determines whether a column should also be multiplied by itself.

Details

Note that the output order of columns corresponds to the order of the columns in x. First column 1 is multiplied with each of the other columns, then column 2 with the remaining columns etc.

Value

A matrix with the same number of rows as x and a number of columns corresponding to c choose 2 (+ c if self is TRUE), where c is the number of columns of x.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
X <- cbind(rep(1, 4), 1:4, 4:1)
pairwise_Schur_product(X)
pairwise_Schur_product(X, self=TRUE)
```

---

panel.hist *Panel plot of histogram and density curve*

Description

Prints the histogram and corresponding density curve

Usage

```
panel.hist(x, col.bar = "gray", ...)
```

Arguments

x	a numeric vector of x values
col.bar	the color of the bars
...	options passed to hist

Details

This function prints a combined histogram and density curve for use with the pairs function

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Ekstrom, CT (2011) *The R Primer*.

Examples

```
pairs(~ Ozone + Temp + Wind + Solar.R, data=airquality,
      lower.panel=panel.smooth, diag.panel=panel.hist,
      upper.panel=panel.r2)
```

panel.r2 *Panel plot of R2 values for pairs*

Description

Prints the R2 with text size depending on the size of R2

Usage

```
panel.r2(x, y, digits = 2, cex.cor, ...)
```

Arguments

x	a numeric vector of x values
y	a numeric vector of y values
digits	a numeric value giving the number of digits to present
cex.cor	scaling fator for the size of text
...	extra options (not used at the moment)

Details

This function is a slight modification of the panel.cor function defined on the pairs help page. It calculated and prints the squared correlation, R2, with text size depending on the proportion of explained variation.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Ekstrom, CT (2011) *The R Primer*.

Examples

```
pairs(~ Ozone + Temp + Wind + Solar.R, data=airquality,
      lower.panel=panel.smooth, upper.panel=panel.r2)
```

picea *Ozone concentration damage to picea spruce*

Description

Damage scores (ordinal scale) for Picea Sitchesis shoots at two dates, at four temperatures, and 4 ozone Levels

Format

An artificial data frame with 18 observations in each of three groups.

date a character vector giving the date

temp temperature in degrees Celcius

conc Ozone concentration at 4 different levels

damage the damage score from 0-4, higher is more damage

count The number of occurrences of this group

Source

P.W. Lucas, D.A. Cottam, L.J. Sheppard, B.J. Francis (1988). "Growth Responses and Delayed Winter Hardening in Sitka Spruce Following Summer Exposure to Ozone," New Phytologist, Vol. 108, pp. 495-504.

Examples

```
data(picea)
```

plr *Fast computation of several simple linear regressions*

Description

Fast computation of several simple linear regression, where the outcome is analyzed with several marginal analyses, or where several outcome are analyzed separately, or a combination of both.

Usage

```
plr(y, x, addintercept = TRUE)

## S3 method for class 'numeric'
plr(y, x, addintercept = TRUE)

## S3 method for class 'matrix'
plr(y, x, addintercept = TRUE)
```

Arguments

y	either a vector (of length N) or a matrix (with N rows)
x	a matrix with N rows
addintercept	boolean. Should the intercept be included in the model by default (TRUE)

Value

a data frame (if Y is a vector) or list of data frames (if Y is a matrix)

Author(s)

Claus Ekstrom <ekstrom@sund.ku.dk>

See Also

mfastLmCpp

Examples

```
N <- 1000  # Number of observations
Nx <- 20   # Number of independent variables
Ny <- 80   # Number of dependent variables

# Simulate outcomes that are all standard Gaussians
Y <- matrix(rnorm(N*Ny), ncol=Ny)
X <- matrix(rnorm(N*Nx), ncol=Nx)

plr(Y, X)
```

---

power_binom_test *Power Calculations for Exact Test of a simple null hypothesis in a Bernoulli experiment*

Description

Compute power of test, or determine parameters to obtain target power.

Usage

```
power_binom_test(
  n = NULL,
  p0 = NULL,
  pa = NULL,
  sig.level = 0.05,
  power = NULL,
  alternative = c("two.sided", "less", "greater")
)
```

Arguments

n	Number of observations
p0	Probability under the null
pa	Probability under the alternative
sig.level	Significance level (Type I error probability)
power	Power of test (1 minus Type II error probability)
alternative	One- or two-sided test

Details

The procedure uses uniroot to find the root of a discontinuous function so some errors may pop up due to the given setup that causes the root-finding procedure to fail. Also, since exact binomial tests are used we have discontinuities in the function that we use to find the root of but despite this the function is usually quite stable.

Value

Object of class power.htest, a list of the arguments (including the computed one) augmented with method and note elements.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

binom.test

Examples

```
power_binom_test(n = 50, p0 = .50, pa = .75)      ## => power = 0.971
power_binom_test(p0 = .50, pa = .75, power = .90) ## =>     n = 41
power_binom_test(n = 50, p0 = .25, power = .90, alternative="less")  ## => pa = 0.0954
```

---

power_mcnemar_test *Power Calculations for Exact and Asymptotic McNemar Test in a 2 by 2 table*

Description

Compute power of test, or determine parameters to obtain target power for matched case-control studies.

Usage

```
power_mcnemar_test(
  n = NULL,
  paid = NULL,
  psi = NULL,
  sig.level = 0.05,
  power = NULL,
  alternative = c("two.sided", "one.sided"),
  method = c("normal", "exact", "cond.exact")
)
```

Arguments

n	Number of observations (number of pairs)
paid	The probability that a case patient is not exposed and that the corresponding control patient was exposed (specifying p_12 in the 2 x 2 table). It is assumed that this is the _smaller_ of the two discordant probabilities.
psi	The relative probability that a control patient is not exposed and that the corresponding case patient was exposed compared to the probability that a case patient is not exposed and that the corresponding control patient was exposed (i.e., p_21 / p_12 in the 2x2 table). Also called the discordant proportion ratio. psi must be larger than or equal to 1 since paid was the smaller of the two discordant probabilities.
sig.level	Significance level (Type I error probability)
power	Power of test (1 minus Type II error probability)
alternative	One- or two-sided test
method	Power calculations based on exact or asymptotic test. The default (normal) corresponds to an approximative test, "exact" is the unconditional exact test, while "cond.exact" is a conditional exact test (given fixed n). The "exact" method is very slow for large values of n so it is most useful for fixed (and moderately-sized) n.

Value

Object of class power.htest, a list of the arguments (including the computed one) augmented with method and note elements.

Note

uniroot is used to solve power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Duffy, S (1984). Asymptotic and Exact Power for the McNemar Test and its Analogue with R Controls per Case

Fagerland MW, 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.

See Also

mcnemar.test

Examples

```
# Assume that pi_12 is 0.125 and we wish to detect an OR of 2.
# This implies that pi_12=0.25, and with alpha=0.05, and a power of 90% you get
power_mcnemar_test(n=NULL, paid=.125, psi=2, power=.9)

power_mcnemar_test(n=NULL, paid=.1, psi=2, power=.8, method="normal")
power_mcnemar_test(n=NULL, paid=.1, psi=2, power=.8)
```

---

power_prop_test *Power Calculations for Two-Sample Test for Proportions with unequal sample size*

Description

Compute power of test, or determine parameters to obtain target power for equal and unequal sample sizes.

Usage

```
power_prop_test(
  n = NULL,
  p1 = NULL,
  p2 = NULL,
  sig.level = 0.05,
  power = NULL,
  ratio = 1,
  alternative = c("two.sided", "one.sided"),
  tol = .Machine$double.eps^0.25
)
```

Arguments

n	Number of observations (in group 1)
p1	Probability in one group
p2	Probability in other group
sig.level	Significance level (Type I error probability)
power	Power of test (1 minus Type II error probability)
ratio	The ratio n2/n1 between the larger group and the smaller group. Should be a value equal to or greater than 1 since n2 is the larger group. Defaults to 1 (equal group sizes)
alternative	String. Can be one- or two-sided test. Can be abbreviated.
tol	Numerical tolerance used in root finding, the default providing (at least) four significant digits

Details

Exactly one of the parameters n, delta, power, sd, sig.level, ratio sd.ratio must be passed as NULL, and that parameter is determined from the others. Notice that the last two have non-NULL defaults so NULL must be explicitly passed if you want to compute them.

Value

Object of class power.htest, a list of the arguments (including the computed one) augmented with method and note elements.

Note

uniroot is used to solve power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

power.prop.test, power_t_test, power.t.test

Examples

```
power_prop_test(n=NULL, p1=.65, p2=.85, power=.8, ratio=2)
```

power_t_test *Power calculations for one and two sample t tests with unequal sample size*

Description

Compute power of test, or determine parameters to obtain target power for equal and unequal sample sizes.

Usage

```
power_t_test(
  n = NULL,
  delta = NULL,
  sd = 1,
  sig.level = 0.05,
  power = NULL,
  ratio = 1,
  sd.ratio = 1,
  type = c("two.sample", "one.sample", "paired"),
  alternative = c("two.sided", "one.sided"),
  df.method = c("welch", "classical"),
  strict = TRUE
)
```

Arguments

n	Number of observations (in the smallest group if two groups)
delta	True difference in means
sd	Standard deviation
sig.level	Significance level (Type I error probability)
power	Power of test (1 minus Type II error probability)
ratio	The ratio n2/n1 between the larger group and the smaller group. Should be a value equal to or greater than 1 since n2 is the larger group. Defaults to 1 (equal group sizes). If ratio is set to NULL (i.e., find the ratio) then the ratio might be smaller than 1 depending on the desired power and ratio of the sd's.
sd.ratio	The ratio sd2/sd1 between the standard deviations in the larger group and the smaller group. Defaults to 1 (equal standard deviations in the two groups)
type	Type of t test
alternative	One- or two-sided test
df.method	Method for calculating the degrees of default. Possibilities are welch (the default) or classical.
strict	Use strict interpretation in two-sided case. Defaults to TRUE unlike the standard power.t.test function.

Details

Exactly one of the parameters n, delta, power, sd, sig.level, ratio sd.ratio must be passed as NULL, and that parameter is determined from the others. Notice that the last two have non-NULL defaults so NULL must be explicitly passed if you want to compute them.

The default strict = TRUE ensures that the power will include the probability of rejection in the opposite direction of the true effect, in the two-sided case. Without this the power will be half the significance level if the true difference is zero.

Value

Object of class power.htest, a list of the arguments (including the computed one) augmented with method and note elements.

Note

uniroot is used to solve power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

power.t.test, power_prop_test, power.prop.test

Examples

```
# Sampling with a ratio of 1:4
power_t_test(delta=300, sd=450, power=.8, ratio=4)

# Equal group sizes but different sd's
# The sd in the second group is twice the sd in the second group
power_t_test(delta=300, sd=450, power=.8, sd.ratio=2)

# Fixed group one size to 50 individuals, but looking for the number of individuals in the
# second group. Different sd's with twice the sd in the larger group
power_t_test(n=50, delta=300, sd=450, power=.8, ratio=NULL, sd.ratio=2)
```

---

prepost.test *Pretest-posttest RCT for quantitative observations with possible missing values*

Description

In a typical pretest-posttest RCT, subjects are randomized to two treatments, and response is measured at baseline, prior to intervention with the randomized treatment (pretest), and at prespecified follow-up time (posttest). Interest focuses on the effect of treatments on the change between mean baseline and follow-up response. Missing posttest response for some subjects is routine, and disregarding missing cases can lead to invalid inference.

Usage

```
prepost.test(baseline, post, treatment, conf.level = 0.95, delta = "estimate")
```

Arguments

baseline	A vector of quantitative baseline measurements
post	A vector of quantitative post-test measurements with same length as baseline. May contain missing values
treatment	A vector of 0s and 1s corresponding to treatment indicator. 1 = treated, Same length as baseline
conf.level	confidence level of the interval
delta	A numeric between 0 and 1 OR the string "estimate" (the default). The proportion of observation treated.

Author(s)

Claus Ekstrom <ekstrom@sund.ku.dk>

References

Marie Davidian, Anastasios A. Tsiatis and Selene Leon (2005). "Semiparametric Estimation of Treatment Effect in a Pretest-Posttest Study with Missing Data". Statistical Science 20, 261-301.

See Also

chisq.test

Examples

```
# From Altman
expo = c(rep(1,9),rep(0,7))
bp1w = c(137,120,141,137,140,144,134,123,142,139,134,136,151,147,137,149)
bp_base = c(147,129,158,164,134,155,151,141,153,133,129,152,161,154,141,156)
diff = bp1w-bp_base
prepost.test(bp_base, bp1w, expo)
```

qdiag *Fast extraction of matrix diagonal*

Description

Fast extraction of matrix diagonal

Usage

```
qdiag(x)
```

Arguments

x	The matrix to extract the diagonal from

Details

Note this function can only be used for extraction

Value

A vector with the diagonal elements

Author(s)

Claus Ekstrom <claus@rprimer.dk>

QIC.geeglm *Quasi Information Criterion*

Description

Function for calculating the quasi-likelihood under the independence model information criterion (QIC), quasi-likelihood, correlation information criterion (CIC), and corrected QIC for one or several fitted geeglm model object from the geepack package.

Usage

```
## S3 method for class 'geeglm'
QIC(object, tol = .Machine$double.eps, ...)

## S3 method for class 'ordgee'
QIC(object, tol = .Machine$double.eps, ...)

## S3 method for class 'geekin'
QIC(object, tol = .Machine$double.eps, ...)

QIC(object, tol = .Machine$double.eps, ...)
```

Arguments

object	a fitted GEE model from the geepack package. Currently only works on geeglm objects
tol	the tolerance used for matrix inversion
...	optionally more fitted geeglm model objects

Details

QIC is used to select a correlation structure. The QICu is used to compare models that have the same working correlation matrix and the same quasi-likelihood form but different mean specifications. CIC has been suggested as a more robust alternative to QIC when the model for the mean may not fit the data very well and when models with different correlation structures are compared.

Models with smaller values of QIC, CIC, QICu, or QICC are preferred.

If the MASS package is loaded then the ginv function is used for matrix inversion. Otherwise the standard solve function is used.

Value

A vector or matrix with the QIC, QICu, quasi likelihood, CIC, the number of mean effect parameters, and the corrected QIC for each GEE object

Author(s)

Claus Ekstrom <claus@rprimer.dk>, Brian McLoone <bmcloone@pdx.edu>, and Steven Orzack <orzack@freshpond.org>

References

Pan, W. (2001). *Akaike's information criterion in generalized estimating equations*. Biometrics, 57, 120-125.
Hardin, J.W. and Hilbe, J.M. (2012). *Generalized Estimating Equations, 2nd Edition*, Chapman and Hall/CRC: New York.
Hin, L.-Y. and Wang, Y-G. (2009). *Working-correlation-structure identification in generalized estimating equations*, Statistics in Medicine 28: 642-658.
Thall, P.F. and Vail, S.C. (1990). *Some Covariance Models for Longitudinal Count Data with Overdispersion*. Biometrics, 46, 657-671.

See Also

geeglm

Examples

```
library(geepack)
data(ohio)
fit <- geeglm(resp ~ age + smoke + age:smoke, id=id, data=ohio,
              family=binomial, corstr="exch", scale.fix=TRUE)
QIC(fit)
```

qpcr *Gene expression from real-time quantitative PCR*

Description

Gene expression levels from real-time quantitative polymerase chain reaction (qPCR) experiments on two different plant lines. Each line was used for 7 experiments each with 45 cycles.

Format

A data frame with 630 observations on the following 4 variables.

`flour`	numeric	Fluorescence level
`line`	factor	Plant lines `rnt` (mutant) and `wt` (wildtype)
`cycle`	numeric	Cycle number for the experiment
`transcript`	factor	Transcript used for the different runs

Source

Data provided by Kirsten Jorgensen <kij@life.ku.dk>.
Added by Claus Ekstrom <ekstrom@life.ku.dk>

References

Morant, M. et al. (2010). Metabolomic, Transcriptional, Hormonal and Signaling Cross-Talk in Superroot2. *Molecular Plant*. 3, p.192–211.

Examples

```
data(qpcr)

#
# Analyze a single run for the wt line, transcript 1
#
run1 <- subset(qpcr, transcript==1 & line=="wt")

model <- nls(flour ~ fmax/(1+exp(-(cycle-c)/b))+fb,
             start=list(c=25, b=1, fmax=100, fb=0), data=run1)

print(model)

plot(run1$cycle, run1$flour, xlab="Cycle", ylab="Fluorescence")
lines(run1$cycle, predict(model))
```

quadform *Fast quadratic form computation*

Description

Fast computation of a quadratic form $t(x) * M * x$.

Usage

```
quadform(x, M, invertM = FALSE, transposex = FALSE)
```

Arguments

x	A matrix with dimensions n*k.
M	A matrix with dimenions n*n. If it is to be inverted then the matrix should be symmetric and positive difinite (no check is done for this)
invertM	A logical. If set to TRUE then M will be inverted before computations (defaults to FALSE)
transposex	A logical. Should the matrix be transposed before computations (defaults to FALSE).

Value

A matrix with dimensions k * k giving the quadratic form

Author(s)

Claus Ekstrom <claus@rprimer.dk>

rainman *Perception of points in a swarm*

Description

Five raters were asked to guess the number of points in a swarm for 10 different figures (which - unknown to the raters - were each repeated three times).

Format

A data frame with 30 observations on the following 6 variables.

SAND The true number of points in the swarm. Each picture is replicated thrice

ME Ratings from judge 1

TM Ratings from judge 2

AJ Ratings from judge 3

BM Ratings from judge 4

LO Ratings from judge 5

Details

The raters har approximately 10 seconds to judge each picture, and the thought it was 30 different pictures. Before starting the experiment they were shown 6 (unrelated) pictures and were told the number of points in each of those pictures. The SAND column contains the picture id and the true number of points in the swarm.

Source

Collected by Claus Ekstrom.

Examples

```
data(rainman)
long <- data.frame(stack(rainman[,2:6]), figure=factor(rep(rainman$SAND,5)))
figind <- interaction(long$figure,long$ind)
# Use a linear random effect model from the
# lme4 package if available
if(require(lme4)) {
  model <- lmer(values ~ (1|ind) + (1|figure) + (1|figind), data=long)
}

#
# Point swarms were generated by the following program
#
## Not run:
set.seed(2) # Original
npoints <- sample(4:30)*4
nplots <- 10
pdf(file="swarms.pdf", onefile=TRUE)

s1 <- sample(npoints[1:nplots])
print(s1)
for (i in 1:nplots) {
  n <- s1[i]
  set.seed(n)
  x <- runif(n)
  y <- runif(n)
  plot(x,y, xlim=c(-.15, 1.15), ylim=c(-.15, 1.15), pch=20, axes=FALSE,
       xlab="", ylab="")
}
s1 <- sample(npoints[1:nplots])
print(s1)
for (i in 1:nplots) {
  n <- s1[i]
  set.seed(n)
  x <- runif(n)
  y <- runif(n)
  plot(y,x, xlim=c(-.15, 1.15), ylim=c(-.15, 1.15), pch=20, axes=FALSE,
       xlab="", ylab="")
}
s1 <- sample(npoints[1:nplots])
print(s1)
```

```
for (i in 1:nplots) {
  n <- s1[i]
  set.seed(n)
  x <- runif(n)
  y <- runif(n)
  plot(-x,y, xlim=c(-1.15, .15), ylim=c(-.15, 1.15), pch=20, axes=FALSE,
       xlab="", ylab="")
}
dev.off()

## End(Not run)
```

---

repmat *Fast replication of a matrix*

Description

Fast generation of a matrix by replicating a matrix row- and column-wise in a block-like fashion

Usage

```
repmat(x, nrow = 1L, ncol = 1L)
```

Arguments

x	A matrix with dimensions r*c.
nrow	An integer giving the number of times the matrix is replicated row-wise
ncol	An integer giving the number of times the matrix is replicated column-wise

Value

A matrix with dimensions (r*nrow) x (c*ncol)

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
m <- matrix(1:6, ncol=3)
repmat(m, 2)     # Stack two copies of m on top of each other
repmat(m, 2, 3)  # Replicate m with two copies on top and three copies side-by-side
```

`residualplot.default` *Plots a standardized residual*

Description

Plots a standardized residual plot from an lm or glm object and provides additional graphics to help evaluate the variance homogeneity and mean.

Usage

```
## Default S3 method:
residualplot(
  x,
  y = NULL,
  candy = TRUE,
  bandwidth = 0.3,
  xlab = "Fitted values",
  ylab = "Std.res.",
  col.sd = "blue",
  col.alpha = 0.3,
  ylim = NA,
  ...
)

## S3 method for class 'lm'
residualplot(
  x,
  y,
  candy = TRUE,
  bandwidth = 0.3,
  xlab = "Fitted values",
  ylab = "Stud.res.",
  col.sd = "blue",
  col.alpha = 0.3,
  ...
)

## S3 method for class 'glm'
residualplot(
  x,
  y,
  candy = TRUE,
  bandwidth = 0.4,
  xlab = "Fitted values",
  ylab = "Std. dev. res.",
  col.sd = "blue",
  col.alpha = 0.3,
```

```
  ...
)

residualplot(
  x,
  y = NULL,
  candy = TRUE,
  bandwidth = 0.3,
  xlab = "Fitted values",
  ylab = "Std.res.",
  col.sd = "blue",
  col.alpha = 0.3,
  ylim = NA,
  ...
)
```

Arguments

x	lm object or a numeric vector
y	numeric vector for the y axis values
candy	logical. Should a lowess curve and local standard deviation of the residual be added to the plot. Defaults to TRUE
bandwidth	The width of the window used to calculate the local smoothed version of the mean and the variance. Value should be between 0 and 1 and determines the percentage of the window width used
xlab	x axis label
ylab	y axis label
col.sd	color for the background residual deviation
col.alpha	number between 0 and 1 determining the transprency of the standard deviation plotting color
ylim	pair of observations that set the minimum and maximum of the y axis. If set to NA (the default) then the limits are computed from the data.
...	Other arguments passed to the plot function

Details

The y axis shows the studentized residuals (for lm objects) or standardized deviance residuals (for glm objects). The x axis shows the linear predictor, i.e., the predicted values for lm objects.

The blue area is a smoothed estimate of 1.96*SD of the standardized residuals in a window around the predicted value. The blue area should largely be rectangular if the standardized residuals have more or less the same variance.

The dashed line shows the smoothed mean of the standardized residuals and should generally follow the horizontal line through (0,0).

Solid circles correspond to standardized residuals outside the range from [-1.96; 1.96] while open circles are inside that interval. Roughly 5

Value

Produces a standardized residual plot

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

rstandard, predict

Examples

```
# Linear regression example
data(trees)
model <- lm(Volume ~ Girth + Height, data=trees)
residualplot(model)
model2 <- lm(Volume ~ Girth + I(Girth^2) + Height, data=trees)
residualplot(model2)
```

`residual_plot` *Plots a standardized residual*

Description

Plots a standardized residual plot from an lm or glm object and provides additional graphics to help evaluate the variance homogeneity and mean.

Usage

```
residual_plot(
  x,
  y = NULL,
  candy = TRUE,
  bandwidth = 0.3,
  xlab = "Fitted values",
  ylab = "Std.res.",
  col.sd = "blue",
  alpha = 0.1,
  ylim = NA,
  ...
)

## Default S3 method:
residual_plot(
  x,
  y = NULL,
```

```
  candy = TRUE,
  bandwidth = 0.3,
  xlab = "Fitted values",
  ylab = "Std.res.",
  col.sd = "blue",
  alpha = 0.1,
  ylim = NA,
  ...
)

## S3 method for class 'lm'
residual_plot(
  x,
  y,
  candy = TRUE,
  bandwidth = 0.3,
  xlab = "Fitted values",
  ylab = "Stud.res.",
  col.sd = "blue",
  alpha = 0.1,
  ...
)

## S3 method for class 'glm'
residual_plot(
  x,
  y,
  candy = TRUE,
  bandwidth = 0.4,
  xlab = "Fitted values",
  ylab = "Std. dev. res.",
  col.sd = "blue",
  alpha = 0.1,
  ...
)
```

Arguments

x	lm object or a numeric vector
y	numeric vector for the y axis values
candy	logical. Should a lowess curve and local standard deviation of the residual be added to the plot. Defaults to TRUE
bandwidth	The width of the window used to calculate the local smoothed version of the mean and the variance. Value should be between 0 and 1 and determines the percentage of the window width used
xlab	x axis label
ylab	y axis label

col.sd	color for the background residual deviation
alpha	number between 0 and 1 determining the transprency of the standard deviation plotting color
ylim	pair of observations that set the minimum and maximum of the y axis. If set to NA (the default) then the limits are computed from the data.
...	Other arguments passed to the plot function

Details

The y axis shows the studentized residuals (for lm objects) or standardized deviance residuals (for glm objects). The x axis shows the linear predictor, i.e., the predicted values for lm objects.

The blue area is a smoothed estimate of 1.96*SD of the standardized residuals in a window around the predicted value. The blue area should largely be rectangular if the standardized residuals have more or less the same variance.

The dashed line shows the smoothed mean of the standardized residuals and should generally follow the horizontal line through (0,0).

Solid circles correspond to standardized residuals outside the range from [-1.96; 1.96] while open circles are inside that interval. Roughly 5

Value

Produces a standardized residual plot

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

rstandard, predict

Examples

```
# Linear regression example
data(trees)
model <- lm(Volume ~ Girth + Height, data=trees)
residual_plot(model)
model2 <- lm(Volume ~ Girth + I(Girth^2) + Height, data=trees)
residual_plot(model2)

# Add extra information about points by adding geom_text to the object produced

m <- lm(mpg ~ hp + factor(vs), data=mtcars)
residual_plot(m) + ggplot2::geom_point(ggplot2::aes(color=factor(cyl)), data=mtcars)
```

`rmvt.pedigree` *Simulate residual multivariate t-distributed data from a polygenic model*

Description

Simulates residual multivariate t-distributed response data from a pedigree where the additive genetic, dominance genetic, and shared environmental effects are taken into account.

Usage

```
rmvt.pedigree(n = 1, pedigree, h2 = 0, c2 = 0, d2 = 0, df = 1)
```

Arguments

`n`	numeric. The number of simulations to generate
`pedigree`	a `pedigree` object
`h2`	numeric. The heritability
`c2`	numeric. The environmentability
`d2`	numeric. The dominance deviance effect
`df`	numeric. The degrees of freedom for the t distribution

Details

The three parameters should have a sum: h2+c2+d2 that is less than 1. The total variance is set to 1, and the mean is zero.

Value

Returns a matrix with the simulated values with n columns (one for each simulation) and each row matches the corresponding individual from the pedigree

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

`pedigree`, `kinship`,

Examples

```
library(kinship2)
library(mvtnorm)
mydata <- data.frame(id=1:5,
                     dadid=c(NA, NA, 1, 1, 1),
                     momid=c(NA, NA, 2, 2, 2),
                     sex=c("male", "female", "male", "male", "male"),
                     famid=c(1,1,1,1,1))
relation <- data.frame(id1=c(3), id2=c(4), famid=c(1), code=c(1))
ped <- pedigree(id=mydata$id, dadid=mydata$dadid, momid=mydata$momid,
                sex=mydata$sex, relation=relation)
rmvt.pedigree(2, ped, h2=.25, df=4)
```

---

rmvtnorm.pedigree *Simulate residual multivariate Gaussian data from a polygenic model*

Description

Simulates residual multivariate Gaussian response data from a pedigree where the additive genetic, dominance genetic, and shared environmental effects are taken into account.

Usage

```
rmvtnorm.pedigree(n = 1, pedigree, h2 = 0, c2 = 0, d2 = 0)
```

Arguments

n	numeric. The number of simulations to generate
pedigree	a pedigree object
h2	numeric. The heritability
c2	numeric. The environmentability
d2	numeric. The dominance deviance effect

Details

The three parameters should have a sum: h2+c2+d2 that is less than 1. The total variance is set to 1, and the mean is zero.

Value

Returns a matrix with the simulated values with n columns (one for each simulation) and each row matches the corresponding individual from the pedigree

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

pedigree, kinship,

Examples

```
library(kinship2)
library(mvtnorm)
mydata <- data.frame(id=1:5,
                     dadid=c(NA, NA, 1, 1, 1),
                     momid=c(NA, NA, 2, 2, 2),
                     sex=c("male", "female", "male", "male", "male"),
                     famid=c(1,1,1,1,1))
relation <- data.frame(id1=c(3), id2=c(4), famid=c(1), code=c(1))
ped <- pedigree(id=mydata$id, dadid=mydata$dadid, momid=mydata$momid,
                sex=mydata$sex, relation=relation)
rmvtnorm.pedigree(2, ped, h2=.25)
```

---

rnorm_perfect *Simulate values from a perfect normal distribution*

Description

Random generation for a perfect normal distribution with mean equal to mean and standard deviation equal to sd.

Usage

```
rnorm_perfect(n, mean = 0, sd = 1)
```

Arguments

n	number of observations. If length(n) > 1, the length is taken to be the number required.
mean	number of mean.
sd	number of standard deviation.

Details

The function will return the same set of quantiles for fixed n. In that sense there is not much randomness going on, and the function is mostly useful for illustrative purposes.

Value

Returns a vector of values from a perfect normal distribution

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
rnorm_perfect(30, mean=10, sd=2)
```

---

rootonorm *Hanging rootogram for normal distribution*

Description

Create a hanging rootogram for a quantitative numeric vector and compare it to a Gaussian distribution.

Usage

```
rootonorm(
  x,
  breaks = "Sturges",
  type = c("hanging", "deviation"),
  scale = c("sqrt", "raw"),
  zeroline = TRUE,
  linecol = "red",
  rectcol = "lightgrey",
  xlab = xname,
  ylab = "Sqrt(frequency)",
  yaxt = "n",
  ylim = NULL,
  mu = mean(x),
  s = sd(x),
  gap = 0.1,
  ...
)
```

Arguments

x	a numeric vector of values for which the rootogram is desired
breaks	Either the character string 'Sturges' to use Sturges' algorithm to decide the number of breaks or a positive integer that sets the number of breaks.
type	if "hanging" then a hanging rootogram is plotted, and if "deviation" then deviations from zero are plotted.
scale	The type of transformation. Defaults to "sqrt" which takes square roots of the frequencies. "raw" yields untransformed frequencies.
zeroline	logical; if TRUE a horizontal line is added at zero.
linecol	The color of the density line for the normal distribution. The default is to make a red density line.

rectcol	a colour to be used to fill the bars. The default of lightgray yields lightgray bars.
xlab, ylab	plot labels. The xlab and ylab refer to the x and y axes respectively
yaxt	Should y axis text be printed. Defaults to n.
ylim	the range of y values with sensible defaults.
mu	the mean of the Gaussian distribution. Defaults to the sample mean of x.
s	the standard deivation of the Gaussian distribution. Defaults to the sample std.dev. of x.
gap	The distance between the rectangles in the histogram.
...	further arguments and graphical parameters passed to plot.

Details

The mean and standard deviation of the Gaussian distribution are calculated from the observed data unless the mu and s arguments are given.

Value

Returns a vector of counts of each bar. This may be changed in the future. The plot is the primary output of the function.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Tukey, J. W. 1972. *Some Graphic and Semigraphic Displays*. In *Statistical Papers in Honor of George W. Snedecor*, p. 293-316.

Examples

```
oldpar <- par()
par(mfrow=c(2,2))
rootonorm(rnorm(200))
rootonorm(rnorm(200), type="deviation", scale="raw")
rootonorm(rnorm(200), mu=1)
rootonorm(rexp(200), mu=1)
par(oldpar)
```

round_percent *Round vector of number to percentages*

Description

Rounds a vector of numeric values to percentages ensuring that they add up to 100

Usage

```
round_percent(x, decimals = 0L, ties = c("random", "last"))
```

Arguments

x	A numeric vector with non-negative values.
decimals	An integer giving the number of decimals that are used
ties	A string that is either 'random' (the default) or 'last'. Determines how to break ties. Random is random, last prefers to break ties at the last position

Details

Returns a vector of numeric values.

Value

Returns a numeric vector of the same length as x

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
f <- c(1,2,1,3,2,1,2,3,1)
round_percent(f)
```

rud *Simulate randomized urn design*

Description

Simulates a randomized treatment based on an urn model.

Usage

```
rud(
  n,
  alpha = c(1, 1),
  beta = 1,
  labels = seq(1, length(alpha)),
  data.frame = FALSE,
  startid = 1
)
```

Arguments

n	the number of individuals to randomize
alpha	a non-negative integer vector of weights for each treatment group. The length of the vector corresponds to the number of treatment groups.
beta	a non-negative integer of weights added to the groups that were not given treatment
labels	a vector of treatment labels. Must be the same length as the length of alpha.
data.frame	A logical that determines if the function should return a vector of group indices (the default, if FALSE) or a data frame (if TRUE).
startid	margin paramaters; vector of length 4 (see par)

Details

The urn model can be described as follows: For k different treatments, the urn design is initiated with a number of balls in an urn corresponding to the start weight (the alpha argument), where each treatment has a specific colour. Whenever a patient arrives, a random ball is drawn from the urn and the colour decides the treatment for the patient. For each of the treatments that weren't chosen we add beta balls of the corresponding colour(s) to the urn to update the probabilities for the next patient.

Value

A vector with group indices. If the argument data.frame=TRUE is used then a data frame with three variables is returned: id, group, and treatment (the group label).

Examples

```
rud(5)
rud(5, alpha=c(1,1,10), beta=5)
```

`scorefct` *Internal functions for the MESS package*

Description

Internal functions for the MESS package

Usage

```
scorefct(o, beta = NULL, testidx = NULL, sas = FALSE)
```

Arguments

`o`	input geepack object from a geeglm fit.
`beta`	The estimated parameters. If set to `NULL` then the parameter estimates are extracted from the model fit object o.
`testidx`	Indices of the beta parameters that should be tested equal to zero
`sas`	Logical. Should the SAS version of the score test be computed. Defaults to `FALSE`.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

`screen_variables` *Screen variable before penalized regression*

Description

Expands a contingency table to a data frame where each observation in the table becomes a single observation in the data frame with corresponding information for each for each combination of the table dimensions.

Usage

```
screen_variables(x, y, lambda = 0.1, method = c("global-strong", "global-DPP"))
```

Arguments

x	A table or matrix
y	A vector of outcomes
lambda	a vector of positive values used for the penalization parameter.
method	a string giving the method used for screening. Two possibilities are "global-strong" and "global-DPP"

Details

Note that no standardization is done (not necessary?)

Value

A list with three elements: lambda which contains the lambda values, selected which contains the indices of the selected variables, and method a string listing the method used.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Hastie, Tibshirani and Wainwright (2015). "Statistical Learning with Sparsity". CRC Press.

Examples

```
x <- matrix(rnorm(50*100), nrow=50)
y <- rnorm(50, mean=x[,1])
screen_variables(x, y, lambda=c(.1, 1, 2))
```

---

segregate.genes *Segregate genes through a pedigree*

Description

Segregate di-allelic genes down through the generations of a pedigree. It is assumed that the founders are independent and that the genes are in Hardy Weinberg equilibrium in the population.

Usage

```
segregate.genes(pedigree, maf)
```

Arguments

pedigree	a pedigree object
maf	a vector of minor allele frequencies for each diallelic gene to segregate through the pedigree

Value

Returns a data frame. Each row matches the order of the individuals in the pedigree and each column corresponds to each of the segregated genes. The data frame contains values 0, 1, or 2 corresponding to the number of copies of the minor allele frequency allele that person has.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

pedigree, kinship,

Examples

```
library(kinship2)
mydata <- data.frame(id=1:5,
                     dadid=c(NA, NA, 1, 1, 1),
                     momid=c(NA, NA, 2, 2, 2),
                     sex=c("male", "female", "male", "male", "male"),
                     famid=c(1,1,1,1,1))
relation <- data.frame(id1=c(3), id2=c(4), famid=c(1), code=c(1))
ped <- pedigree(id=mydata$id, dadid=mydata$dadid, momid=mydata$momid,
                sex=mydata$sex, relation=relation)
segregate.genes(ped, c(.1, .3, .5))
```

---

sinv *Invert a symmetric positive-definite matrix*

Description

Inverts a symmetric positive-definite matrix without requiring the Matrix package.

Usage

```
sinv(obj)
```

Arguments

obj	The symmetric positive-definite matrix

Details

This function does no error checking and it is up to the user to ensure that the input is indeed symmetric, positive-definite, and a matrix.

Value

A matrix of the same size as the input object

Author(s)

Claus Ekstrom, <claus@rprimer.dk>.

Examples

```
m <- matrix(c(1, 0, .5, .5, 0, 1, .5, .5, .5, .5, 1, .5, .5, .5, .5, 1), 4)
sinv(m)
```

---

smokehealth *Effect of smoking on self reported health*

Description

Effect of smoking at 45 years of age on self reported health five years later. Data are on a sample of males from the Glostrup survey.

Format

A table with daily smoking categories for the rows and self reported health five years later as the columns.

Source

Data example found on the internet but originates from Svend Kreiner

Examples

```
data(smokehealth)
m <- smokehealth
m[,3] <- m[,3]+ m[,4]
m[4,] <- m[4,] + m[5,]
m <- m[1:4,1:3]
gkgamma(m)
chisq.test(m)
```

soccer *Danish national soccer players*

Description

Players on the Danish national soccer team. The dataset consists of all players who have been picked to play on the men's senior A-team, their position, date-of-birth, goals and matches.

Format

A data frame with 805 observations on the following 5 variables.

name a factor with names of the players

DoB a Date. The date-of-birth of the player

position a factor with levels `Forward` `Defender` `Midfielder` `Goalkeeper`

matches a numeric vector. The number of A matches played by the player

goals a numeric vector. The number of goals scored by the player in A matches

Source

Data collected from the player database of DBU on March 21st, 2014. See https://www.dbu.dk for more information.

Examples

```
data(soccer)

birthmonth <- as.numeric(format(soccer$DoB, "%m"))
birthyear <- as.numeric(format(soccer$DoB, "%Y"))
```

superroot2 *Gene expression data from two-color dye-swap experiment*

Description

Gene expression levels from two-color dye-swap experiment on 6 microarrays. Arrays 1 and 2 represent the first biological sample (ie, the first dye swap), 3 and 4 the second, and arrays 5 and 6 the third.

Format

A data frame with 258000 observations on the following 5 variables.

color a factor with levels `green` `red` representing the dye used for the gene expression

array a factor with levels `1` `2` `3` `4` `5` `6` corresponding to the 6 arrays

gene a factor with 21500 levels representing the genes on the arrays

plant a factor with levels `rnt` `wt` for the two types of plants: runts and wild type

signal a numeric vector with the gene expression level (normalized but not log transformed)

Source

Data provided by Soren Bak <bak@life.ku.dk>.
Added by Claus Ekstrom <ekstrom@sund.ku.dk>

References

Morant, M. et al. (2010). Metabolomic, Transcriptional, Hormonal and Signaling Cross-Talk in Superroot2. *Molecular Plant*. 3, p.192–211.

Examples

```
data(superroot2)
# Select one gene
g1 <- superroot2[superroot2$gene=="AT2G24000.1",]
model <- lm(log(signal) ~ plant + color + array, data=g1)
summary(model)
```

`tracemp` *Fast computation of trace of matrix product*

Description

Fast computation of the trace of the matrix product trace(t(A)

Usage

```
tracemp(A, B)
```

Arguments

`A`	A matrix with dimensions n*k.
`B`	A matrix with dimenions n*k.

Value

The trace of the matrix product

Author(s)

Claus Ekstrom <claus@rprimer.dk>

Examples

```
A <- matrix(1:12, ncol=3)
tracemp(A, A)
```

---

usd *Unbiased standard deviation*

Description

This function computes the unbiased standard deviation of the values in x. If na.rm is TRUE then missing values are removed before computation proceeds.

Usage

```
usd(x, na.rm = FALSE)
```

Arguments

x	a numeric vector or an R object but not a factor coercible to numeric by as.double(x)
na.rm	logical. Should missing values be removed?

Details

Like var this uses denominator n - 1. The standard deviation of a length-one or zero-length vector is NA.

Value

A scalar

Examples

```
sd(1:5)
usd(1:5)
```

`wallyplot.default` *Plots a Wally plot*

Description

Produces a 3x3 grid of residual- or qq-plots plots from a lm object. One of the nine subfigures is the true residual plot/qqplot while the remaining are plots that fulfill the assumptions of the linear model

Usage

```
## Default S3 method:
wallyplot(
  x,
  y = x,
  FUN = residualplot,
  hide = TRUE,
  simulateFunction = rnorm,
  model = NULL,
  ...
)

## S3 method for class 'lm'
wallyplot(
  x,
  y = x,
  FUN = residualplot,
  hide = TRUE,
  simulateFunction = lmsimresiduals,
  ...
)

wallyplot(
  x,
  y = x,
  FUN = residualplot,
  hide = TRUE,
  simulateFunction = rnorm,
  ...
)
```

Arguments

x	a numeric vector of x values, or an lm object.
y	a numeric vector of y values of the same length as x or a n * 9 matrix of y values - one column for each of the nine plots to make. The first column is the one corresponding to the results from the dataset

FUN	a function that accepts an x, y and ... argument and produces a graphical model validation plots from the x and y values.
hide	logical; if TRUE (the default) then the identity of the true residual plot is hidden until the user presses a key. If FALSE then the true residual plot is shown in the center.
simulateFunction	The function used to produce y values under the null hypothesis. Defaults to rnorm
model	Optional input to simulateFunction
...	Other arguments passed to the plot function FUN

Details

Users who look at residual plots or qqnorm plots for the first time often feel they lack the experience to determine if the residual plot is okay or if the model assumptions are indeed violated. One way to convey "experience" is to plot a series of graphical model validation plots simulated under the model assumption together with the corresponding plot from the real data and see if the user can pinpoint one of them that looks like an odd-one-out. If the proper plot from the real data does not stand out then the assumptions are not likely to be violated.

The Wallyplot produces a 3x3 grid of plots from a lm object or from a set of pairs of x and y values. One of the nine subfigures is the true plot while the remaining are plots that fulfill the assumptions of the linear model. After the user interactively hits a key the correct residual plot (correponding to the provided data) is shown.

The plotting function can be set using the FUN argument which should be a function that accepts x, y and ... arguments and plots the desired figure. When y is a single vector the same length as x then the function simulateFunction is used to generate the remaining y values corresponding the situations under the null.

For a description of the features of the default residual plot see the help page for residualplot.

Author(s)

Claus Ekstrom <claus@rprimer.dk>

References

Ekstrom, CT (2014) *Teaching 'Instant Experience' with Graphical Model Validation Techniques*. Teaching Statistics (36), p 23-26

Examples

```
## Not run:
data(trees)
res <- lm(Volume ~ Height + Girth, data=trees)
wallyplot(res)


# Create a grid of QQ-plot figures
# Define function to plot a qq plot with an identity line
```

```
qqnorm.wally <- function(x, y, ...) { qqnorm(y, ...) ; abline(a=0, b=1) }
wallyplot(res, FUN=qqnorm.wally, main="")

# Define function to simulate components+residuals for Girth
cprsimulate <- function(n) {rnorm(n)+trees$Girth}
# Create the cpr plotting function
cprplot <- function(x, y, ...) {plot(x, y, pch=20, ...) ;
                                lines(lowess(x, y), lty=3)}
# Create the Wallyplot
wallyplot(trees$Girth, trees$Girth+rstudent(res), FUN=cprplot,
          simulateFunction=cprsimulate, xlab="Girth")

## End(Not run)
```

`write.xml` *Write a data frame in XML format*

Description

Writes the data frame to a file in the XML format.

Usage

```
write.xml(data, file = NULL, collapse = TRUE)
```

Arguments

`data`	the data frame object to save
`file`	the file name to be written to.
`collapse`	logical. Should the output file be collapsed to make it fill less? (Defaults to TRUE)

Details

This function does not require the **XML** package to be installed to function properly.

Value

None

Author(s)

Claus Ekstrom, <claus@rprimer.dk> based on previous work by Duncan Temple Lang.

Examples

```
## Not run:
data(trees)
write.xml(trees, file="mydata.xml")

## End(Not run)
```

Index

∗ **~htests**
feature.test, 26
∗ **datagen**
age, 6
auc, 7
clipit, 12
common.shared, 17
extended.shared, 24
founder.shared, 29
plr, 52
rmvt.pedigree, 72
rmvtnorm.pedigree, 73
segregate.genes, 80
∗ **datasets**
bdstat, 8
bees, 9
clotting, 12
earthquakes, 23
greenland, 33
happiness, 34
icecreamads, 37
kwdata, 39
lifeexpect, 39
matched, 42
nh4, 45
picea, 52
qpcr, 63
rainman, 64
smokehealth, 82
soccer, 83
superroot2, 83
∗ **file**
sinv, 81
write.xml, 88
∗ **hplot**
residual_plot, 69
residualplot.default, 67
rnorm_perfect, 74
rootonorm, 75
∗ **htest**
drop1.geeglm, 21
drop1.geem, 22
gkgamma, 32
power_binom_test, 53
power_mcnemar_test, 54
power_prop_test, 56
power_t_test, 58
prepost.test, 59
QIC.geeglm, 61
∗ **iplot**
col.alpha, 14
col.shade, 15
col.tint, 16
panel.hist, 50
panel.r2, 51
wallyplot.default, 86
∗ **manip**
adaptive.weights, 4
categorize, 11
expand_table, 24
fac2num, 26
loadRData, 40
lower.tri.vector, 41
monte_carlo_chisq_test, 44
round_percent, 77
scorefct, 79
screen_variables, 79
∗ **models**
geekin, 30
∗ **print**
ht, 35
∗ **univar**
cmd, 13
∗ **utilities**
ordered.clusters, 46

adaptive.weights, 4
add_torows, 5
age, 6

approx, *8*
as.POSIXlt, *6*
auc, 7

bdstat, 8
bees, 9
bin, 10
binom.test, *54*

categorize, 11
chisq.test, *33*, *60*
clipit, 12
clotting, 12
cmd, 13
col.alpha, 14
col.shade, 15
col.tint, 16
colCumSum, 16
common.shared, 17
conditional_rowMeans, 18
cor.test, *47*
cumsumbinning, 19

dCor, 20
dCov, 20
drop1, *21*, *23*
drop1.geeglm, 21
drop1.geem, 22
duplicated, *46*

earthquakes, 23
expand_table, 24
extended.shared, 24

fac2num, 26
feature.test, 26
filldown, 28
founder.shared, 29

geekin, 30
ginv, *62*
gkgamma, 32
greenland, 33

happiness, 34
ht, 35
hwe_frequencies, 36

icecreamads, 37
integrate, *8*

ks_cumtest, 38
kwdata, 39

lifeexpect, 39
load, *40*
loadRData, 40
lower.tri, *41*
lower.tri.vector, 41

matched, 42
maximum_subarray, 42
mcnemar.test, *56*
mfastLmCpp, 43
monte_carlo_chisq_test, 44

nh4, 45

ordered.clusters, 46

p.adjust, *47*
pairwise.cor.test, 46
pairwise_combination_indices, 47
pairwise_distance_correlation, 48
pairwise_Schur_product, 49
panel.hist, 50
panel.r2, 51
par, *78*
picea, 52
plr, 52
power.prop.test, *57*, *59*
power.t.test, *57*, *59*
power_binom_test, 53
power_mcnemar_test, 54
power_prop_test, 56, *59*
power_t_test, *57*, 58
predict, *69*, *71*
prepost.test, 59
print.geekin *(*geekin*)*, 30

qdiag, 61
QIC *(*QIC.geeglm*)*, 61
QIC.geeglm, 61
qpcr, 63
quadform, 64

rainman, 64
repmat, 66
residual_plot, 69
residualplot, *87*
residualplot *(*residualplot.default*)*, 67

residualplot.default, 67
rmvt.pedigree, 72
rmvtnorm.pedigree, 73
rnorm_perfect, 74
rootogram *(*rootonorm*)*, 75
rootonorm, 75
round_percent, 77
rstandard, *69*, *71*
rud, 78

scorefct, 79
screen_variables, 79
segregate.genes, 80
sinv, 81
smokehealth, 82
soccer, 83
solve, *62*
splinefun, *8*
superroot2, 83

tracemp, 84

usd, 85

wallyplot *(*wallyplot.default*)*, 86
wallyplot.default, 86
write.xml, 88