Package ‘MESS’

February 19, 2015

Type Package
Title Miscellaneous Esoteric Statistical Scripts
Version 0.3-2
Date 2014-10-19
Author Claus Ekstrom <claus@rprimer.dk>
Maintainer Claus Ekstrom <claus@rprimer.dk>
Depends geepack, glmnet, Matrix
Imports kinship2, parallel, mvtnorm
Suggests MASS, lme4
Description A mixed collection of useful and semi-useful functions, some of which may even be referenced in The R Primer book.
ByteCompile true
License GPL-2
NeedsCompilation no
Repository CRAN
Date/Publication 2014-10-21 20:02:36

R topics documented:

MESS-package ................................................................. 2
adaptive.weights ......................................................... 3
auc ................................................................. 4
bdstat ............................................................. 5
bees ........................................................... 6
clotting ............................................................ 7
cmd ............................................................... 8
common.shared .................................................... 9
desctable .......................................................... 10
drop1.geeglm .......................................................... 11
feature.test ......................................................... 12
geekin ............................................................. 13
Collection of miscellaneous useful and semi-useful functions and add-on functions that enhances a number of existing packages and provides in particular in relation to statistical genetics.

Details

Package: MESS
Type: Package
Version: 1.0
Date: 2012-03-29
License: GPL-2

Author(s)

Claus Ekstrom <claus@rprimer.dk>
Maintainer: Claus Ekstrom <claus@rprimer.dk>

References

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


See Also

glmnet
Examples

```r
library(glmnet)
set.seed(1)
x <- matrix(rnorm(50000), nrow=50)
y <- rnorm(50, mean=x[,1])
weights <- adaptive.weights(x, y)
glmnet(x, y, penalty.factor=weights$weights)
```

---

**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

```r
auc(x, y, from = min(x), to = max(x), type = c("linear", "spline"), ...)
```

### Arguments

- **x**: a numeric vector of 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 smallest y 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.
- **...**: additional arguments passed on to approx. 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.
bdstat

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.

Usage
data(bdstat)

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**

```r
data(bdstat)

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

# Create 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.

**Usage**

data(bees)

**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).
clotting

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)

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 resistance to varying extent. The purpose is to relate anticoagulent resistance to gender and location and perhaps weight. Dose of injection is, however, administered according to weight and gender.

Usage

data(clotting)

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

```r
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**

```r
cmd(m1, m2)
```

**Arguments**

- `m1`: First correlation matrix
- `m2`: Second correlation matrix

**Details**

Returns a value between 0 and 1. The correlation matrix distance becomes zero if the correlation matrices are equal up to a scaling factor and one if they differ to a maximum extent.

**Value**

Returns the correlation matrix distance.

**Author(s)**

Claus Ekstrom <claus@rprimer.dk>

**References**

common.shared

Examples

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

### 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

```r
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 `bdsmatrix` 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, 2, 2, 4, 4, 6, 2, 0, 0, 12, 13),
dad = c(0, 0, 0, 1, 1, 3, 3, 7, 0, 0, 11, 10),
sex = c(0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1))
tped <- with(test1, pedigree(id, dad, mom, sex))
common.shared(tped)

---

desctable Create a descriptive table of summaries

Description

Creates a descriptive table of summary statistics

Usage

desctable(object, group = NULL, proportions = FALSE, ...)

Arguments

- **object**: an object for which a descriptive summary table is desired
- **group**: A factor the same length as object (or the elements of object if that is a data frame) that is used to split up the summary for each level of group. If set to NULL (the default) then every observation is assumed to be in one group
- **proportions**: Logical. Should proportions be listed for categorical data (defaults to FALSE)
- **...**: any other arguments passed of functions

Value

A list with length the same as object with items

- **output**: Output for use with the descriptive table
- **p**: p-value for testing equality across groups (set to NULL if groups=NULL)

Author(s)

Claus Ekstrom <claus@rprimer.dk>

See Also

summary

Examples

desctable(ToothGrowth, ToothGrowth$supp)
**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**

```r
## 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`: 
- `test`: 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 variance estimate. sandwich is an alias for robust.
- `method`: 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**

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

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)
```

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, 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

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

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

```r
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


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("VGGood", "Good", "Fair", "Bad") # General health status
rownames(smoke) <- c("Never", "No more", "1-14", "15-24", "25+") # Smoke amount
kgamma(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

Usage

data(greenland)
happiness

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.
Added by Claus Ekstrom <ekstrom@life.ku.dk>

References

Aktuelt Naturvidenskab september 2010.

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

Usage

data(happiness)

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)
lower.tri.vector

**Source**

Data collected by Ellen Ekstroem.

**Examples**

```r
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)
```

---

**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**

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

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

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

---

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

```r
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.
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>
power.binom.test

See Also

binom.test

Examples

### Should be **DIRECTLY** executable !! ----
### => Define data, use random,
###-- do help(data-index) for the standard data sets.

### The function is currently defined as

```
function(n = NULL, p0 = NULL, pa = NULL, sig.level = 0.05, power = NULL,
  alternative = c("two.sided", "less", "greater"))
{
  if (sum(sapply(list(n, p0, pa, power, sig.level), is.null)) != 1)
    stop("exactly one of 'n', 'p0', 'pa', 'power', and 'sig.level' must be NULL")
  if (!is.null(sig.level) && !is.numeric(sig.level) || any(0 >
    sig.level | sig.level > 1))
    stop("'sig.level' must be numeric in [0, 1]")
  alternative <- match.arg(alternative)
  pfun <- function(n, p0, pa, sig.level, alternative) {
    n <- ceiling(n)
    power <- switch(alternative, less = {
      pbinom(qbinom(1 - sig.level, size = n, prob = p0,
        lower.tail = FALSE) - 1, size = n, prob = pa)
    },
      greater = {
        pbinom(qbinom(1 - sig.level, size = n, prob = p0),
          size = n, prob = pa, lower.tail = FALSE)
    },
      two.sided = {
        lx <- qbinom(sig.level, size = n, prob = p0)
        ux <- qbinom(sig.level, size = n, prob = p0, lower.tail = FALSE)
        x <- c(seq(0, lx), seq(ux, n))
        d <- dbinom(x, size = n, prob = p0)
        ordd <- order(d)
        cs <- cumsum(sort(d))
        xval <- which.min(cs < sig.level) - 1
        ssh <- d[ordd[xval]]
        relErr <- 1 + 1e-07
        m <- n * p0
        if (xval == 0) return(0)
        if (x[ordd[xval]] < m) {
          i <- seq.int(from = ux, to = n)
          y <- sum(dbinom(i, n, p0) <= ssh * relErr)
          pbinom(x[ordd[xval]], size = n, prob = pa) +
            pbinom(n - y, size = n, prob = pa, lower.tail = FALSE)
        } else {
          i <- seq.int(from = 0, to = lx)
          y <- sum(dbinom(i, n, p0) <= ssh * relErr)
          pbinom(y - 1, size = n, prob = pa) + pbinom(x[ordd[xval]] -
            1, n, pa, lower.tail = FALSE)
        }
    })
```
power

p.body <- Vectorize(pfun)
ppp <- body(p.body)
qqq <- quote({
  do.call("mapply", c(FUN = pfun, list(n, p0, pa, sig.level,
    alternative), SIMPLIFY = TRUE, USE.NAMES = TRUE))
})
if (is.null(power))
  power <- eval(qqq)
else if (is.null(n)) {
  ans <- uniroot(function(n) eval(qqq) - power, c(2, 1e+06))
  n <- ans$root + (ans$f.root < 0)
}
else if (is.null(p0))
  p0 <- uniroot(function(p0) eval(p.body) - power, c(1e-07,
    1 - 1e-07)$root
else if (is.null(pa))
  pa <- uniroot(function(pa) eval(p.body) - power, c(1e-07,
    1 - 1e-07)$root
else if (is.null(sig.level))
  sig.level <- uniroot(function(sig.level) eval(p.body) -
    power, c(1e-10, 1 - 1e-10)$root
else stop("internal error", domain = NA)
NOTE <- "One-sample exact binomial power calculation"
structure(list(n = n, p0 = p0, pa = pa, sig.level = sig.level,
  power = power, alternative = alternative, note = NOTE,
  method = METHOD), class = "power.htest")

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"))

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

psi
The odds ratio for exposure in case and control individuals

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

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

See Also
mcnemar.test

Examples
```r
power.mcnemar.test(n=NULL, paid=.1, psi=2, power=.8, method="normal")
power.mcnemar.test(n=NULL, paid=.1, psi=2, power=.8)
```
QIC

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

```r
## S3 method for class 'geeglm'
QIC(object, ...)
## S3 method for class 'geekin'
QIC(object, ...)
## S3 method for class 'ordgee'
QIC(object, ...)
```

Arguments

- `object`: a fitted GEE model from the geepack package. Currently only works on geeglm objects
- `...`: 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>
qpcr

References


See Also

ggeglm

Examples

library(geepack)
data(ohio)
fit <- ggeglm(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.

Usage

data(qpcr)

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


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))

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).

Usage

data(rainman)

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
L0 Ratings from judge 5
Details

The raters had 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
#
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(-1.5, 1.15), ylim=c(-1.15, 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(-1.5, 1.15), ylim=c(-1.15, 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(-1.15, 1.15), pch=20, axes=FALSE,
       xlab="", ylab="")
}
dev.off()

residualplot

Plots a standardized residual

Description

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

Usage

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

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 transparency of the standard deviation plotting color  
... Other arguments passed to the plot function
Details

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

The brown area is a smoothed estimate of $1.96*SD$ of the standardized residuals in a window around the predicted value. The brown 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 $(0,0)$.

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)

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: \( h^2 + c^2 + d^2 \) 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

```r
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=0.25, df=4)
```
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))
rootonorm <- data.frame(id=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)

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


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)

---

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))

---

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.

Usage
data(soccer)

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


Examples

data(soccer)

birthmonth <- as.numeric(format(soccer$DoB, "%m"))
bbirthyear <- as.numeric(format(soccer$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 (i.e., the first dye swap), 3 and 4 the second, and arrays 5 and 6 the third.

Usage

data(superroot2)

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@life.ku.dk>

References

Examples

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

---

**wallyplot**

*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**

```r
## Default S3 method:
wallyplot(x, y = x, FUN = residualplot, hide = TRUE,
          simulateFunction = rnorm, ...)
## S3 method for class 'lm'
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`.
- `...`: 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 (corresponding 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.

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

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

# Define function to simulate components+residuals for Girth
cppsimulate <- 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=cppsimulate, xlab="Girth")
Index

*Topic \textasciitildehtests
feature.test, 12
*Topic datagen
  auc, 4
  common.shared, 9
  rmvt.pedigree, 29
  rmvtnorm.pedigree, 31
  segregate.genes, 33
*Topic datasets
  bdstat, 5
  bees, 6
  clotting, 7
  greenland, 16
  happiness, 17
  qpcr, 25
  rainman, 26
  soccer, 34
  superroot2, 35
*Topic hplot
  residualplot, 28
  rootonorm, 32
*Topic htest
  drop1.geeglm, 11
  gkgamma, 15
  power.binom.test, 20
  power.mcnemar.test, 22
  QIC, 24
*Topic iplot
  wallplot, 36
*Topic manip
  adaptive.weights, 3
  lower.tri.vector, 18
*Topic models
  geekin, 13
*Topic package
  \textsc{mess}-package, 2
*Topic print
  desctable, 10
*Topic univar
  cmd, 8
*Topic \textit{utilities}
  ordered.clusters, 19
  adaptive.weights, 3
  approx, 5
  auc, 4
  bdstat, 5
  bees, 6
  binom.test, 21
  chisq.test, 16
  clotting, 7
  cmd, 8
  common.shared, 9
  desctable, 10
  drop1, 11
  drop1.geeglm, 11
  duplicated, 20
  feature.test, 12
  geekin, 13
  ginv, 24
  gkgamma, 15
  greenland, 16
  happiness, 17
  integrate, 5
  lower.tri, 19
  lower.tri.vector, 18
  mcnemar.test, 23
  \textsc{mess}(\textsc{mess}-package), 2
  \textsc{mess}-package, 2
  ordered.clusters, 19
INDEX

power.binom.test, 20
power.mcnemar.test, 22
predict, 29
print.geekin(geekin), 13
QIC, 24
qpcr, 25
rainman, 26
residualplot, 28
rmvt.pedigree, 29
rmvtnorm.pedigree, 31
rootogram(rootonorm), 32
rootonorm, 32
rstandard, 29
segregate.genes, 33
soccer, 34
solve, 24
splinefun, 5
summary, 10
superroot2, 35
wallyplot, 36