Package ‘cg’

May 8, 2026

Type Package

Version 1.0-4

Title Compare Groups, Analytically and Graphically

Maintainer Bill Pikounis <`cg@billpikounis.net`>

Description Comprehensive data analysis software, and the name ``cg'' stands for ``compare groups.'' Its genesis and evolution are driven by common needs to compare administrations, conditions, etc. in medicine research and development. The current version provides comparisons of unpaired samples, i.e. a linear model with one factor of at least two levels. It also provides comparisons of two paired samples. Good data graphs, modern statistical methods, and useful displays of results are emphasized.

License GPL-2 | GPL-3

LazyLoad yes

LazyData true

Depends R (>= 3.2.3), Hmisc (>= 3.17-1)

Imports VGAM (>= 1.0-0), methods, graphics, stats, utils, grid, MASS, lattice, survival, multcomp, nlme, rms

NeedsCompilation no

Author Bill Pikounis [aut, cre, cph],
John Oleynick [aut],
Eva Ye [ctb]

Repository CRAN

Date/Publication 2025-04-13 16:00:11 UTC

Contents

---

cg-package *Compare Groups, Analytically and Graphically*

Description

cg is comprehensive data analysis software, and stands for "compare groups." Its genesis and evolution are driven by common needs to compare administrations, conditions, etc. in medicine research and development. The current version provides comparisons of unpaired samples, i.e. a linear model with one factor of at least two levels. It also provides comparisons of two paired samples. Good data graphs, modern statistical methods, and useful displays of results are emphasized.

Details

Package: cg
Type: Package
Version: 1.0-4
Date: 2025-04-12
License: GPL (>= 2)
LazyLoad: yes
LazyData: yes
Depends: R (>= 3.2.3), Hmisc (>= 3.17-1) Imports: VGAM (>= 1.0-0), methods, graphics, stats, utils, grid, MASS, lattice, survival, multcomp, nlme, rms

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Maintainer: Bill Pikounis <cg@billpikounis.net>

References

Pikounis, B. and Oleynick, J. (2013). "The cg Package for Comparison of Groups", *Journal of Statistical Software*, Volume 52, Issue 1, 1-27, `https://www.jstatsoft.org/v52/i01/`.

Examples

```
#### One Factor data

data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
## Exploratory methods
pointGraph(canine.data)

boxplot(canine.data)
```

```
descriptiveTable(canine.data)

## Fits and Comparisons
canine.fit <- fit(canine.data)

canine.comps0 <- comparisonsTable(canine.fit)

errorBarGraph(canine.fit)

canine.comps1 <- comparisonsTable(canine.fit,  mcadjust=TRUE,
                                   type="allgroupstocontrol", refgrp="CC")

comparisonsGraph(canine.comps1)

grpSummaryTable(canine.fit)

## Diagnostics
varianceGraph(canine.fit)

qqGraph(canine.fit)

# downweightedTable(canine.fit, cutoff=0.95)

## Sample Size calculations
canine.samplesize <- samplesizeTable(canine.fit, direction="increasing",
                                     mmdvec=c(10, 25, 50, 75, 100))

samplesizeGraph(canine.samplesize)

## Censored Data Set
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
pointGraph(gmcsfcens.data)

boxplot(gmcsfcens.data)

descriptiveTable(gmcsfcens.data)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

## Paired Samples

data(anorexiaFT)

anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
```

```
                                              logscale=TRUE)

## Exploratory methods
descriptiveTable(anorexiaFT.data)

profileGraph(anorexiaFT.data)

diffGraph(anorexiaFT.data)

## Fits and Comparisons
anorexiaFT.fit <- fit(anorexiaFT.data)

comparisonsTable(anorexiaFT.fit)
```

`anorexiaFT` *AnorexiaFT Data Set in the cg package*

Description

A data frame used to illustrate the **cg** package. It has a Paired Samples / Paired Difference layout.

Usage

```
data(anorexiaFT)
```

Format

A 17-by-2 data frame from an study of young female anorexia patients. It is a subset of the `anorexia` data frame included in the **MASS** package. Out of the three groups there, only the factor level `FT` group makes up the `anorexiaFT` data frame. The endpoint is Weight in pounds (lbs), `FT` represents "Family Therapy" treament, and the paired levels are

`PreWt`	Patient weight before treatment
`PostWt`	Patient weight after treatment

See `anorexia` in the **MASS** package for additional description.

Details

The `anorexiaFT` data set that comes with the **cg** package is in `groupcolumns` format for the `prepareCGPairedDifferenceData` call. Each column represents a group, and each row represents an individual patient, or experimental unit. Each observation in a row that spans the two group columns are paired individual response or outcome values.

The purpose of the study was to evaluate effects of multiple treatments on weight as a marker for anorexia, and to compare their relative effectiveness.

See `anorexia` in the **MASS** package for references and more details.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

See Also

`anorexia`, `prepareCGPairedDifferenceData`

Examples

```
data(anorexiaFT)
str(anorexiaFT)
```

`boxplot.cgOneFactorData`

Box Plot Graph of Groups from a cgOneFactorData object

Description

Create graph of boxplots of groups in a `cgOneFactorData` object.

Usage

```
## S4 method for signature 'cgOneFactorData'
boxplot(x, ...)
```

Arguments

`x`	A `cgOneFactorData` object.
`...`	Additional arguments, both *optional*. Two are currently valid:

`logscale` A `logical` value, indicating whether or not the boxplot graph should be plotted on the logarithmic scale. If `logscale` is not specified, its value is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `logscale` argument.

`ticklabels` A list of two components:

- `mod` Can be either of these two values,
  - `"replace"` Before graphing the data, remove any automatically generated tickmarks for the y-axis, and create the tickmarks specified in the `marks` component (see below).
  - `"add"` Before graphing the data, add tickmarks specified in the `marks` component to the automatically generated ones.
- `marks` A vector of tickmarks to be placed on the y-axis. Any numeric representations will be coerced to chracter.

Details

For uncensored data, the boxplot for each group produced is a standard boxplot, similar to that produced by **graphics**::`boxplot.default`, but with the median shown as a "+" and the mean shown as a "o". A warning is added to the plot if any of the groups or all of the groups have 5 or fewer observations (in which case a plot from `pointGraph.cgOneFactorData` might be more suitable).

For censored data, Kaplan-Meier estimates are used for the quantiles, as proposed by Gentleman and Crowley (1991). The **survival**::`survfit` conventions are followed for interpolation of these quantiles. Extreme values that are censored are drawn as open arrow heads rather than open circles. Left-censored values are shown as a shallow "V", which is actually just a rotated downward ">" sign. Similarly, right-censored values are shown as a deeper "^", which is a actually just a rotated upward ">" sign. Individual points are `jitter`ed, and open circles are used for complete observations to alleviate potential overlap and the danger of representing multiple points as a single point. Individual censored values are similarly jittered. With enough censored data observations in a group, certain quantiles may not be estimable, and thus a complete box would not appear.

If `logscale=TRUE`, the tick marks for the y-axis on the left side of the plot show original values, while the ticks mark for the y-axis on the right side of the graph show base 10 log values.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

The heading for the graph is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `analysisname` argument. The label for the y-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `endptname` argument. The number of decimal places printed in the ticks on the y-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `digits` argument.

The minimum and maximum values from the range of the data are respectively labeled in the bottom and top left corners of the graph region.

If group labels along the x-axis seem to overlap in the standard horizontal form, they will be rotated 45 degrees.

Value

`boxplot.cgOneFactorData` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Gentleman, R.C. and Crowley, J. (1991). "Graphical Methods for Censored Data", *Journal of the American Statistical Association*, Volume 86, 678-683.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
boxplot(canine.data)

## Plot the data on the original scale instead of the log scale
boxplot(canine.data, logscale=FALSE)

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
boxplot(gmcsfcens.data)
```

`canine` *Canine Data Set in the cg package*

Description

A data frame used to illustrate the **cg** package. It has a One Factor / One-Way / Unpaired Samples layout.

Usage

```
data(canine)
```

Format

A 5-by-5 data frame with 5 numeric observations from an experiment on the following 5 groups of beagle dogs.

`AE`	castration plus estradiol and androstanediol
`E1`	castration plus low dose estradiol
`E2`	castration plus high dose estradiol
`CC`	castration alone
`NC`	no castration (normal controls)

Details

The `canine` data set that comes with the **cg** package is in `groupcolumns` format for the `prepareCGOneFactorData` call. Each column represents a group, and the observations in that group's column are the individual response or outcome values.

The 5 groups are regarded as levels of one factor in the `prepareCGOneFactorData`, `fit`, and other methods in the **cg** package.

An alternative format of this data set is contained in `canine.listfmt`. See that help file for details, including how it would be read and `prepare`d by **cg**.

The purpose of this experiment was to evaluate the effect of a physiological dose of estradiol on prostate growth in dogs using ultrasound. See the reference below for details. Comparisons amongst all five groups are of interest.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

References

Rhodes, L., Ding, V.D.H., Kemp, R.K., Khan, M.S., Nakhla, A.M., Pikounis, B., Rosner, W., Saunders, H.M. and Feeney, W.P. (2000). "Estradiol causes a dose dependent stimulation of prostate growth in castrate beagle dogs." *The Prostate*, Volume 44, 8-18.

See Also

`canine.listfmt`, `prepareCGOneFactorData`

Examples

```
data(canine)
str(canine)
```

`canine.listfmt` *Canine Data Set in the cg package*

Description

A data frame used to illustrate the **cg** package. It has a One Factor / One-Way / Unpaired Samples layout.

Usage

```
data(canine.listfmt)
```

Format

A 25-by-2 data frame with 5 numeric observations from an experiment on each of the following 5 groups of beagle dogs.

`AE`	castration plus estradiol and androstanediol
`E1`	castration plus low dose estradiol
`E2`	castration plus high dose estradiol
`CC`	castration alone
`NC`	no castration (normal controls)

The above 5 items are the levels of the first column's factor, named `grp`. The second column `size` contains the numeric observations.

Details

The `canine.listfmt` data set that comes with the **cg** package is in `listed` format for the `prepareCGOneFactorData` call.

The 5 groups are regarded as levels of one factor in the `prepareCGOneFactorData`, `fit`, and other methods in the **cg** package.

`canine.listfmt` is an alternative format of the `canine` data set. See that help file for details. Once the data set is `prepare`d into a `cgOneFactorData` object, all the subsequent methods work on the object in the same way.

The purpose of this experiment was to evaluate the effect of a physiological dose of estradiol on prostate growth in dogs using ultrasound. See the reference below for details. Comparisons amongst all five groups are of interest.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

References

Rhodes, L., Ding, V.D.H., Kemp, R.K., Khan, M.S., Nakhla, A.M., Pikounis, B., Rosner, W., Saunders, H.M. and Feeney, W.P. (2000). "Estradiol causes a dose dependent stimulation of prostate growth in castrate beagle dogs." *The Prostate*, Volume 44, 8-18.

See Also

`canine`, `prepareCGOneFactorData`

Examples

```
data(canine.listfmt)
str(canine.listfmt)

## Analogous to prepareCGOneFactorData call on canine data frame format,
## subsequent methods will work for canine.listfmt.data:
```

```
canine.listfmt.data <- prepareCGOneFactorData(canine.listfmt, format="listed",
                                              analysisname="Canine",
                                              endptname="Prostate Volume",
                                              endptunits=expression(plain(cm)^3),
                                              digits=1, logscale=TRUE,
                                              refgrp="CC")
## as they do on canine.data:

canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
```

`cgInternalClasses` *cg package Internal Virtual Classes*

Description

cg package Internal Virtual Classes designed for polymorphic slots

Details

The virtual classes

`characterOrExpression`

`characterOrNULL`

`dataframeMatrixOrNULL`

`dataframeOrNULL`

`numericOrNULL`

`olsfit`

`rrfit`

`aftfit`

`uvfit`

are used internally by the cg package, and designed as polymorphic slots.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

`cgInternalUtilities` *cg package Internal Utilities*

Description

cg package Internal Utility functions and objects not intended for user-level calls

Details

The functions

`stndErr` `geoMean` `pctToRatio` `makeZeroScore` `unwind` `unwrap` `graphStampCG` `setupAxisTicks` `setupLog10AxisTicks` `tryAgain` `seeHelpFile` `paragraphWrap` `cgMessage` `factorInSeq` `setupGrpNameTicks` `xTicksCex` `yTicksCex` `rmTicks` `minmaxTicks` `plotGrpNameTicks` `boxplotStamp` `errorBarGraphStamp` `comparisonsGraphStamp` `errorBarGraphApproximateStamp` `trimWhiteSpace` `chopZeroes` `fmtRatioToPercent` `fmtDifference` `fmtRatio` `fmtPercent` `fmtPvalue` `cgDevice` `contrastMatrix` `blockDiag` `rangeExtend` `getNumDigits` `makeCensored` `multcompInform` `multcompDone` `isAllEqual` `makeEndptLabel` `catCharExpr` `residualgrptrend.helper` `fround` `fround.charcens` `chop.matrix` `stripmiss` `makeTickMarks` `scaleVar` `makeContrastVec` `cg.largest.empty` `qminmin` `unpaste` `grpsummary` `samplesize` `samplesizegraph` `boxplotcensoreddata` `descriptive.censoreddata` `pairwisecompsmatrix`

are used internally by the **cg** package. See source code for details.

The `blockDiag` function is adapted from a Ben Bolker function contribution on R-help in 2002.

The `factorInSeq` function is exported since it may be useful for a user. It is a simple wrapper around `factor` with the order of its levels determined by first occurrence of each level in its `x` vector argument.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

`cgLineColors` *Color Choice order for Graph Lines*

Description

A vector named `cgLineColors` that assigns colors to lines for graphing methods in the **cg** package.

Usage

```
cgLineColors
```

Format

A vector containing ten values in this order: `black` `blue` `green` `red` `orange` `brown` `yellow` `darkblue` `darkgreen` `darkgray`

Details

This is a package internal convenience, and not intended for use or modification by the end user.

If more than ten values are needed then recycling will occur.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

`cgValidity` *cg package Validity Checks*

Description

cg package Internal Validity functions not intended for user-level calls. These are used on input arguments for methods and functions.

Details

`validAlpha` `validPower` `validDataFormat` `validBoolean` `validCharacter` `validNumeric` `validNumericOrCensored` `validList` `validAtomicVec` `validArgMatch` `validDotsArg` `validDotsArgs` `getDotsArgName` `parsePartialName` `reportInvalidArg` `validEqualLength` `validArgMatch` `validArgDigits` `validArgModel` `validCGOneFacGroupColDfr` `validCGOneFacListedDfr` `validCensor` `validZeroScore` `validAddConstant` `validAft` `validFitType` `validErrorDf` `validComparisonType` `validEstimates` `validGrpNames` `validN` `validCutoffWt` `validDenDf`

are used internally by the **cg** package.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

`comparisons` *Create a table of comparisons amongst groups*

Description

Create a table of comparisons based on group estimates and variance-covariance matrix. Pairwise or custom specified contrasts are estimated and tested.

Usage

```
comparisons(estimates, varcovmatrix, errordf = Inf, endptscale,
            mcadjust = FALSE, alpha = 0.05, type = "pairwisereflect",
            contrastmatrix = NULL, n, offset = NULL, cnames = "derive",
            analysisname = "", endptname = "", digits = NULL, addpct = FALSE,
            display = "print")
```

Arguments

`estimates`	A named vector of estimates. Each estimate element is a measure of the center of the group. The name of each group must be present in the names attribute of the vector.
`varcovmatrix`	The estimated variance-covariance matrix associated with the `estimates`. Must have the same number of columns and rows as the length of the `estimates` vector.
`errordf`	Can be one of three types of values: `Inf` The default, and will just use standard Gaussian (normal) distribution quantile for the critical points in each comparison; `numeric` A finite positive number that will be used for the degrees of freedom for the t-distribution quantile; `"approx"` Will try to apply a Satterthwaite approximation based on the `variance-covariance` matrix and `n` to estimate the degrees of freedom for each comparison. The variance-covariance matrix will need to be diagonal. Only will be accepted when `mcadjust=FALSE`.
`endptscale`	Must be specified as `"log"` or `"original"`. If `"log"` then the `estimates` vector is assumed to be in the log scale, and calculations will transform it to the original scale for the generated table.
`mcadjust`	Do a multiple comparisons adjustment, based on the simultaneous inference capabilities of the **multcomp** package. See Details below. The default value is `FALSE`. If `mcadjust=TRUE` is specified, there will be a delay, usually just for a few seconds, due to computing time of the critical point in order to conduct the adjusted comparisons.
`alpha`	Significance level, by default set to `0.05` to correspond to 95% confidence.
`type`	Can be one of four values:

	`"pairwisereflect"` The default value, it calculates and lists all possible pairwise comparison permutations, as each pair ordering is included. In other words, Groups A vs. B and B vs. A will be included.
	`"pairwise"` Calculates and lists all possible pairwise comparison combinations. Once a pair such as Groups A vs. B is specified, then the reflective B vs. A is not included. So the number of comparisons presented is half that produced by `"pairwisereflect"`. The ordering of group levels in the `estimates` vector is used to determine which ordering is included and which is not. If all orderings are of interest, such as for `endptscale=="log"`, use the `"pairwisereflect"` value above.
	`"allgroupstocontrol"` Takes the first element group of the `estimate` vector, deems it the "control" group, and constructs pairwise comparisons of all other groups to it.
	`"custom"` Indicates that a custom matrix of comparisons will be constructed, and that matrix needs to be specified in the `contrastmatrix` argument.
`contrastmatrix`	Only relevant if `type="custom"` is specified. In that case, a numeric matrix with the number of rows equal to the number of comparisons of interest is needed. The number of columns must be equal to the number of groups in the `estimate` vector. Each row in the matrix is assumed to represent a contrast of coefficients amongst the groups that defines the comparison of interest.
`n`	Needs to be specified only when `errordf="approx"`. In this case it needs to be a vector of group sample sizes, the same in length as the `estimates` vector.
`offset`	*Optional*, If for example a numeric constant was added to all response values before calculation of the `estimates` as means in the `"log"` scale, this could be used to adjust the estimates and comparisons appropriately. The default is `NULL`.
`cnames`	If the default value of `"derive"` is used, row names are derived for the table that reflect the A vs. B type of comparison items in each row, using the names attrributes of the `estimates` vector and the middle term of `"vs."` Otherwise, this can be explicitly specified and needs to be a character vector of the same length as `estimates` .
`analysisname`	*Optional*, a character text that will be printed along with the results table. The default value is the empty `""`.
`endptname`	*Optional*, a character text that will be printed along with the results table. The default value is the empty `""`.
`digits`	*Optional*, For output display purposes, values will be rounded to this numeric value. Only the integers of 0, 1, 2, 3, and 4 are accepted. No rounding is done during any calculations. The default value is `NULL`, which will examine each individual `estimates` value and choose the one that has the maximum number of digits after any trailing zeroes are ignored. The max number of digits will be 4.
`addpct`	Only relevant if `endptscale=="original"`. An column of percent differences is added for the comparisons, as a descriptive supplement to the original scale differences that are formally estimated.
`display`	One of three valid values:
	`"print"` The default value; It calls a `print` method for the created object, which is a formatted text output of the table(s).

`"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired.

`"show"` Calls the default `showDefault` method, which will just print out the `comparisons` return object.

Details

Only two-sided Wald-type of confidence intervals are possible with this function.

When `mcadjust=TRUE`, a status message of `"Some time may be needed as the critical point " "from the multcomp::summary.glht function call is calculated "` is displayed at the console. This computed critical point is used for all subsequent p-value and confidence interval calculations.

The **multcomp** package provides a unified way to calculate critical points based on the comparisons of interest in a "family." Thus a user does not need to worry about choosing amongst the myriad names of multiple comparison procedures.

Value

Creates a return data frame object that specifies the comparison of the form A vs. B in each row, and with these columns:

`estimate` The difference in group estimates in the comparison: A vs. B. If `endptscale="log"`, this will be back-transformed to a percent difference scale.

`se` The estimated standard error of the difference `estimate`. If `endptscale="log"`, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

`lowerci` The lower 100 * (1-`alpha`) % confidence limit of the difference `estimate`. With the default `alpha=0.05`, this is 95%. If `endptscale="log"`, the confidence limit is first computed in the logarithmic scale of analysis, and then back-transformed to a percent difference scale.

`upperci` The upper 100 * (1-`alpha`) % confidence limit of the difference `estimate`. With the default `alpha=0.05`, this is 95%. If `endptscale="log"`, the confidence limit is first computed in the logarithmic scale of analysis, and then back-transformed to a percent difference scale.

`pval` The computed p-value from the test of the difference `estimate`.

`meanA` **or** `geomeanA` The estimated "mean" for the left hand side "A" of the A vs. B comparison. If `endptscale="log"`, this is a back-transform to the original scale, and therefore is a "geometric" mean, and will be labelled `geomeanA`. Otherwise it is the arithmetic mean and labelled `meanA`.

`seA` The estimated standard error of the `meanA` `estimate`. If `endptscale="log"`, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

`meanB` **or** `geomeanB` The estimated "mean" for the right hand side "B" of the A vs. B comparison. If `endptscale="log"`, this is a back-transform to the original scale, and therefore is a "geometric" mean, and will be labelled `geomeanB`. Otherwise it is the arithmetic mean and labelled `meanB`.

`seB` The estimated standard error of the `meanB` `estimate`. If `endptscale="log"`, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

An additional column `addpct` of percent differences is added if `endptscale=="original"` and `addpct=TRUE`, as a descriptive supplement to the original scale differences that are formally estimated.

Warning

This function was created for internal use in the **cg** package as its use can be seen in the `comparisonsTable` methods code. Therefore any direct use of it needs to be done cautiously.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Hothorn, T., Bretz, F., Westfall, P., Heiberger, R.M., and Schuetzenmeister, A. (2010). The **multcomp** package.

Hothorn, T., Bretz, F., and Westfall, P. (2008). "Simultaneous Inference in General Parametric Models", *Biometrical Journal*, 50, 3, 346-363.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")

canine.fit <- fit(canine.data)


## Easier way: notice the comparisonsTable call

comparisonsTable(canine.fit, model="olsonly")

## Manual way
## Instead of comparisonsTable(canine.fit, model="olsonly")

comparisons(estimates=canine.fit@olsfit$coef,
            varcovmatrix=vcov(canine.fit@olsfit),
            errordf=canine.fit@olsfit$df.residual,
            endptscale="log",
            analysisname="Canine",
            digits=1,
            endptname="Prostate Volume")
```

`comparisonsGraph` *Graph comparisons specified amongst groups*

Description

Generic function to create a Comparisons Graph based on a Comparisons Table created in turn by the **cg** package.

Usage

```
comparisonsGraph(compstable, cgtheme=TRUE, device="single",
                 wraplength=20, cex.comps=0.7, ...)
```

Arguments

`compstable`	A `comparisonsTable` object created by a `comparisonsTable` method from the **cg** package. There is one class of objects that is currently available: `cgOneFactorComparisonsTable`, which is prepared by the `comparisonsTable.cgOneFactorFit` method.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely, `background`, `strip.shingle` and `strip.background` are each set to `"white"`.
`device`	Can be one of three values: `"single"` The default, which will put all graph panels on the same device page. `"multiple"` Relevant only when more than one panel of graphs is possible. In that case, a new graphics device is generated each newly generated single-paneled graph. `"ask"` Relevant only when more than one panel of graphs is possible. In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that user input confirmation is needed before the graphs are drawn.
`wraplength`	On the left hand vertical axis are each A vs. B comparison label from the `compstable` object. An attempt at sensible formatting when a newline is needed is made, but adjustment by this argument may be needed. The default is `20` characters before wrapping to a newline.
`cex.comps`	Similar to `wraplength`, adjustment of this argument parameter can be made to fit the comparison labels on the left hand vertical axis.
`...`	Additional arguments, depending on the specific method written for the `compstable` object. Currently, there is only one such specific method; see `comparisonsGraph.cgOneFactorComparisonsTable` for any additional arguments that can be specified.

Value

The main purpose is the side effect of graphing to the current device. See the specific methods for discussion of any return values.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`comparisonsGraph.cgOneFactorComparisonsTable`

Examples

```
#### One Factor data

data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.comps1 <- comparisonsTable(canine.fit,  mcadjust=TRUE,
                                  type="allgroupstocontrol", refgrp="CC")

comparisonsGraph(canine.comps1)
```

`comparisonsgraph` *Create an graph of comparisons*

Description

Creates a graph to see comparisons based on group estimates and variance-covariance matrix

Usage

```
comparisonsgraph(compstable, difftype, analysisname = "",
                 endptname = "", alpha = 0.05, digits = NULL,
                 titlestamp = TRUE, explanation = TRUE,
                 wraplength = 20, cex.comps = 0.7,
                 ticklabels = NULL, ...)
```

Arguments

`compstable`	A data frame object of form like that created by the `comparisons` function.
`difftype`	Must be specified as one of the following: `"percent"` Presumes the estimated differences in `compstable` are Percent, and thus will space the x-axis logarithmically; `"amount"` Presumes the estimated differences in `compstable` are Simple Amounts, and thus will space the x-axis in untransformed scale; `"simple"` Synonym for `amount`, presumes the estimated differences in `compstable` are Simple Amounts, and thus will space the x-axis in untransformed scale.
`analysisname`	*Optional*, a character text or math-valid expression that will used in the graph title. The default value is the empty `""`.
`endptname`	*Optional*, a character text or math-valid expression that that will be used as the x-axis label of the graph. The default value is the empty `""`.
`alpha`	Significance level, by default set to `0.05`. This is only used for labelling purposes.
`digits`	*Optional*, For output display purposes in the graph, values will be rounded to this numeric value. Only the integers of 0, 1, 2, 3, and 4 are accepted. No rounding is done during any calculations. The default value is `NULL`, which will examine each individual data value and choose the one that has the maximum number of digits after any trailing zeroes are ignored. The max number of digits will be 4.
`explanation`	If `TRUE`, which is the default, add explanatory message to the graph rendering about "Error bars that do not cross the zero line indicate statistically significant difference(s)" along with the confidence level derived from `alpha`.
`titlestamp`	Specify text to the graph in the top of graph area, otherwise a default description of "Comparisons Graph" and `analysisname` will be constructed.
`wraplength`	On the left hand axis are each A vs. B comparison label from the `compstable` data frame. An attempt at sensible formatting when a newline is needed is made, but adjustment by this argument may be needed. The default is `20` characters before wrapping to a newline.
`cex.comps`	Similar to `wraplength`, adjustment of this argument parameter can be made to fit the comparison labels on the left hand axis.
`ticklabels`	*Optional*, before graphing the data, remove any automatically generated tickmarks for the x-axis, and use these tickmarks instead. A vector of tickmarks to be placed on the x-axis. Any numeric representations will be coerced to character.
`...`	Additional arguments. None are currently used.

Details

The minimum and maximum values across all the bar ends are added inside the plot region in blue, flush against the x-axis. In two panel cases, there is a tendency to fall outside the panel area even though right justified is used for the `adj` parameter of functions like `panel.text`.

Value

`comparisonsgraph` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Warning

This function was created for internal use in the **cg** package as its use can be seen in the `comparisonsGraph` methods source code. Therefore any direct use of it needs to be done cautiously.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`comparisons`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)
canine.comps <- comparisonsTable(canine.fit)

## Easier way: notice the camel-case of the comparisonsGraph call

comparisonsGraph(canine.comps, model="olsonly")

## Manual way
## Instead of comparisonsGraph(canine.comps, model="olsonly")

canine.compstable <- comparisons(estimates=canine.fit@olsfit$coef,
                                 varcovmatrix=vcov(canine.fit@olsfit),
                                 errordf=canine.fit@olsfit$df.residual,
                                 endptscale="log",
                                 analysisname="Canine",
                                 digits=1,
                                 endptname="Prostate Volume")

comparisonsgraph(canine.compstable,
                 difftype="percent",
                 analysisname="Canine",
                 digits=1,
```

```
                endptname=expression(paste( plain('Prostate Volume'),
                                      ' (', plain(cm)^3  ,  ')' ))
              )
```

`comparisonsGraph.cgOneFactorComparisonsTable`

Create an graph of the comparisons in a cgOneFactorComparisonsTable object

---

Description

Creates a graph to see comparisons in a cgOneFactorComparisonsTable object

Usage

```
## S4 method for signature 'cgOneFactorComparisonsTable'
comparisonsGraph(compstable, cgtheme=TRUE, device="single",
 wraplength = 20, cex.comps = 0.7, ...)
```

Arguments

`compstable` A `cgOneFactorComparisonsTable` object prepared by the `comparisonsTable.cgOneFactorFit` method.

`cgtheme` When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely, `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.

`device` Can be one of three values:

- `"single"` The default, which will put all graphs on the same device page. For example, when resistant & robust and classical least squares are present and `model="both"` (the default), a 2 x 1 paneled graph will be created.
- `"multiple"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default). In that case, a new graphics device is generated to hold the resistant & robust version, as a single-paneled graph. The classical least squares version is on the previous device.
- `"ask"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default). In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that user input confirmation is needed before the graphs are drawn.

`wraplength` On the left hand axis are each A vs. B comparison label from the `compstable` object. An attempt at sensible formatting when a newline is needed is made, but adjustment by this argument may be needed. The default is `20` characters before wrapping to a newline.

`cex.comps` Similar to `wraplength`, adjustment of this argument parameter can be made to fit the comparison labels on the left hand axis.

`...` Additional arguments. Two are currently valid:

`model` For `cgOneFactorOneFactorComparisonsTable` objects that have classical least squares `lm()` or resistant & robust `rlm()` table slots, the following argument values are possible:

- `"both"` Graphs of Comparisons Tables based on both the ordinary classical least squares and resistant & robust slots are populated. This is the default when both slots are present in the `cgOneFactorComparisonsTable` object specified in the `compstable` argument. If the resistant & robust fit is not available, this value is not relevant.
- `"olsonly"` Only an Comparisons Graph based on the ordinary classical least squares table slot is performed.
- `"rronly"` Only a Comparisons Graph based on the resistant and robust table slot is performed.

For other possible `cgOneFactorComparisonsTable` table slots such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the appropriate comparisons graph will be calculated for these model types.

`ticklabels` A list of two components:

- `mod` Can be either of these two values,
  - `"replace"` Before graphing the data, remove any automatically generated tickmarks for the x-axis, and create the tickmarks specified in the `marks` component below.
  - `"add"` Before graphing the data, add tickmarks specified in the `marks` component to the automatically generated ones.
- `marks` A vector of tickmarks to be placed on the x-axis. Any numeric representations will be coerced to character.

Details

The minimum and maximum values across all the bar ends are added inside the plot region in blue, flush against the x-axis. In two panel cases, there is a tendency to fall outside the panel area even though right justified is used for the `adj` parameter of functions like `panel.text`. The number of decimal places are determined by the `digits` and `endptscale` values in the `compstable@settings` slot.

Value

`comparisonsGraph.cgOneFactorComparisonsTable` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorComparisonsTable`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

## Comparisons Tables
canine.comps0 <- comparisonsTable(canine.fit)

canine.comps1 <- comparisonsTable(canine.fit,  mcadjust=TRUE,
                                   type="allgroupstocontrol", refgrp="CC")


## Comparisons Graphs
comparisonsGraph(canine.comps0)

comparisonsGraph(canine.comps1)

comparisonsGraph(canine.comps1, cex.comps=0.9,
                 ticklabels=list(mod="add", marks=c(300, 700)))
```

`comparisonsGraph.cgPairedDifferenceComparisonsTable`

Create an graph of the comparisons in a cgPairedDifferenceComparisonsTable object

Description

Creates a graph to see comparisons in a cgPairedDifferenceComparisonsTable object

Usage

```
## S4 method for signature 'cgPairedDifferenceComparisonsTable'
comparisonsGraph(compstable,
cgtheme=TRUE, device="single", wraplength = 20, cex.comps = 0.7, ...)
```

Arguments

`compstable`	A `cgPairedDifferenceComparisonsTable` object prepared by the `comparisonsTable.cgPairedDifferenceFit` method.

`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely, `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`device`	Can be one of three values: `"single"` The default, which will put all graphs on the same device page. For example, when resistant & robust and classical least squares are present and `model="both"` (the default), a 2 x 1 paneled graph will be created. `"multiple"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default). In that case, a new graphics device is generated to hold the resistant & robust version, as a single-paneled graph. The classical least squares version is on the previous device. `"ask"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default). In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that user input confirmation is needed before the graphs are drawn.
`wraplength`	On the left hand axis are each A vs. B comparison label from the `compstable` object. An attempt at sensible formatting when a newline is needed is made, but adjustment by this argument may be needed. The default is `20` characters before wrapping to a newline.
`cex.comps`	Similar to `wraplength`, adjustment of this argument parameter can be made to fit the comparison labels on the left hand axis.

`...`	Additional arguments. Two are currently valid:
	`model` For `cgPairedDifferencePairedDifferenceComparisonsTable` objects that have classical least squares `lm()` or resistant & robust `rlm()` table slots, the following argument values are possible:
	  `"both"` Graphs of Comparisons Tables based on both the ordinary classical least squares and resistant & robust slots are populated. This is the default when both slots are present in the `cgPairedDifferenceComparisonsTable` object specified in the `compstable` argument. If the resistant & robust fit is not available, this value is not relevant.
	  `"olsonly"` Only an Comparisons Graph based on the ordinary classical least squares table slot is performed.
	  `"rronly"` Only a Comparisons Graph based on the resistant and robust table slot is performed.
	`ticklabels` A list of two components:
	  `mod` Can be either of these two values,
	    `"replace"` Before graphing the data, remove any automatically generated tickmarks for the x-axis, and create the tickmarks specified in the `marks` component below.
	    `"add"` Before graphing the data, add tickmarks specified in the `marks` component to the automatically generated ones.
	  `marks` A vector of tickmarks to be placed on the x-axis. Any numeric representations will be coerced to character.

Details

The minimum and maximum values across the bar ends are added inside the plot region in blue, flush against the x-axis. The number of decimal places are determined by the `digits` and `endptscale` values in the `compstable@settings` slot.

Value

`comparisonsGraph.cgPairedDifferenceComparisonsTable` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceComparisonsTable`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1,
                                                  logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

anorexiaFT.comps0 <- comparisonsTable(anorexiaFT.fit)
anorexiaFT.comps1 <- comparisonsTable(anorexiaFT.fit,  model="olsonly", display="none")

comparisonsGraph(anorexiaFT.comps0)
comparisonsGraph(anorexiaFT.comps1)
```

`comparisonsTable` *Create a Table of Comparisons amongst Groups*

Description

Create a table of comparisons based on a fit by the **cg** package.

Usage

```
comparisonsTable(fit, type = "pairwisereflect", alpha = 0.05, addpct = FALSE,
 display = "print", ...)
```

Arguments

`fit`	A fit object created with a `fit` method from the **cg** package. See specific methods.
`type`	Can be one of four values: `"pairwisereflect"` The default value; It calculates and lists all possible pairwise comparison permutations, as each pair order is included. In other words, Groups A vs. B and B vs. A will be included. `"pairwise"` Calculates and lists all possible pairwise comparison combinations. Once a pair such as Groups A vs. B is specified, then the reflective B vs. A is not included. So the number of comparisons is half that produced by `"pairwisereflect"`. The ordering of group levels in the `fit` object is used to determine which ordering is included and which is not. If all orderings are of interest, such as for `settings$endptscale=="log"` in the `fit` object, use the `"pairwisereflect"` value above. `"allgroupstocontrol"` Takes the value of `settings$refgrp` in the **cg** `fit` object, deems it the "control" group, and constructs pairwise comparisons of all other groups to it. `"custom"` Indicates that a custom matrix of comparisons will be constructed, and that matrix needs to be specified in the `contrastmatrix` argument of a method.
`alpha`	Significance level, by default set to `0.05`.
`addpct`	Only relevant if `settings$endptscale=="original"` in the fit object. An column of percent differences is added for the comparisons, as a descriptive supplement to the original scale differences that are formally estimated.
`display`	One of three valid values: `"print"` The default value, it calls a `print` method for the created `ComparisonsTable` object, which is a formatted text output of the table(s). `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `ComparisonsTable` object components.
`...`	Additional arguments, depending on the specific method written for the object. See the method-specific documentation for additional details.

Value

A method-specific `comparisonsTable` object is returned. See the specific methods for discussion of return values.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`comparisonsTable.cgOneFactorFit`, `comparisonsTable.cgPairedDifferenceFit`.

Examples

```
#### One Factor data

data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.comps0 <- comparisonsTable(canine.fit)

canine.comps1 <- comparisonsTable(canine.fit,  mcadjust=TRUE,
                                   type="allgroupstocontrol", refgrp="CC")


data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

gmcsfcens.comps <- comparisonsTable(gmcsfcens.fit)

## Paired Difference data

data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

comparisonsTable(anorexiaFT.fit)
```

`comparisonsTable.cgOneFactorFit` *Create a table of comparisons amongst groups with the cgOneFactorFit object*

Description

Create a table of comparisons based on the cgOneFactorFit object. Pairwise or custom specified contrasts are estimated and tested. A cgOneFactorComparisonsTable class object is created.

Usage

```
## S4 method for signature 'cgOneFactorFit'
comparisonsTable(fit, type="pairwisereflect",
 alpha=0.05, addpct=FALSE, display="print", ...)
```

Arguments

`fit` An object of class `cgOneFactorFit`.

`type` Can be one of four values:

- `"pairwisereflect"` The default value, it calculates and lists all possible pairwise comparison permutations, as each pair order is included. In other words, Groups A vs. B and B vs. A will be included.
- `"pairwise"` Calculates and lists all possible pairwise comparison combinations. Once a pair such as Groups A vs. B is specified, then the reflective B vs. A is not included. So the number of comparisons is half that produced by `"pairwisereflect"`. The ordering of group levels in the `fit` object is used to determine which ordering is included and which is not. If all orderings are of interest, such as for `settings$endptscale=="log"` in the fit objects, use the `"pairwisereflect"` value above.
- `"allgroupstocontrol"` Takes the value of `settings$refgrp` in the **cg** `fit` object, deems it the "control" group, and constructs pairwise comparisons of all other groups to it. This setting is required when the `refgrp` argument is specified in the call (see Additional Arguments section below.)
- `"custom"` Indicates the a custom matrix of comparisons will be constructed, and that matrix needs to be specified in the `contrastmatrix` argument.

`alpha` Significance level, by default set to `0.05`.

`addpct` Only relevant if `settings$endptscale=="original"` in the fit object. An column of percent differences is added for the comparisons, as a descriptive supplement to the original scale differences that are formally estimated.

`display` One of three valid values:

- `"print"` The default value; It calls a `print` method for the created `cgOneFactorComparisonsTable` object, which is a formatted text output of the table(s).

- `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired.
- `"show"` Calls the default `showDefault` method, which will just print out the `cgOneFactorComparisonsTable` components.

`...` Additional arguments.

`mcadjust` Do a multiple comparisons adjustment, based on the simultaneous inference capabilities of the **multcomp** package. See Details below. The default value is `FALSE`. If `mcadjust=TRUE` is specified, there will be a delay, usually just for a few seconds, due to computing time of the critical point in order to conduct the adjusted comparisons.

`contrastmatrix` Only relevant if `type="custom"` is specified. In that case, a numeric matrix with the number of rows equal to the number of comparisons of interest. The number of columns must be equal to the number of group means. Each row in the matrix is assumed to represent a contrast of coefficients amongst the groups that defines the comparison of interest.

`refgrp` If left at the default value of `NULL`, it will be set to the `settings$refgrp` value in the **cg** `fit` object. When set, it is deemed the "reference", or "control" group, so that pairwise comparisons of all other groups to it will be constructed when `type="allgroupstocontrol"`. Please note the `type="allgroupstocontrol"` setting is REQUIRED when the `refgrp` argument is specified in the call with a valid non-`NULL` value.

`model` For `cgOneFactorFit` `fit` objects that have classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible:

- `"both"` Comparison tables based on both the ordinary classical least squares and resistant & robust fits are created. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `fit` argument. If the resistant & robust fit is not available, this value is not relevant.
- `"olsonly"` Only a comparison table based on the ordinary classical least squares `olsfit` fit slot is performed.
- `"rronly"` Only a comparison table based on the resistant and robust `rrfit` fit slot is performed.

For other possible `cgOneFactorFit` fit components such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the appropriate comparisons table will be calculated for these model types.

Details

When `mcadjust=TRUE`, a status message of `"Some time may be needed as the critical point"` `"from the multcomp::summary.glht function call is calculated"` is displayed at the console. This computed critical point is used for all subsequent p-value and confidence interval calculations.

The **multcomp** package provides a unified way to calculate critical points based on the comparisons of interest in a "family". Thus a user does not need to worry about choosing amongst the myriad names of multiple comparison procedures.

Value

Creates an object of class `cgOneFactorComparisonsTable`, with the following slots:

`ols.comprs` The table of comparisons based on the `olsfit` component of the `cgOneFactorFit`, unless `model="rronly"` is specified. In that case the slot value is `NULL`. Will not be appropriate in the case where a valid `aftfit` component is present in the `cgOneFactorFit` object. See below for the data frame structure of the table.

`rr.comprs` The table of comparisons based on the `rrfit` component of the `cgOneFactorFit` object, if a valid resistant & robust fit object is present. If `rrfit` is a simple character value of `"No fit was selected."`, or `model="olsonly"` was specified, then the value is `NULL`. See below for the data frame structure of the table.

`aft.comprs` The table of comparisons based on the `aftfit` component of the `cgOneFactorFit` object if a valid accelerated failure time fit object is present. If `aftfit` is a simple character value of `"No fit was selected."`, then the value is `NULL`. See below for the data frame structure of the table.

`uv.comprs` The table of comparisons based on the `uvfit` component of the `cgOneFactorFit` object if a valid unequal variances fit object is present. The error degrees of freedom for each comparison estimate and test is individually estimated with a Satterthwaite approximation. See below for the data frame structure of the table.

`settings` A list of settings carried from the `cgOneFactorFit` `fit` object, and the addition of some specified arguments in the method call above: `alpha`, `mcadjust`, `type`, and `addpct`. These are used for the `print.cgOneFactorComparisonsTable` method, invoked for example when `display="print"`.

The data frame structure of the comparisons table in a `*.comprs` slot consists of `row.names` that specify the comparison of the form A vs. B, and these columns:

`estimate` The difference in group means in the comparison: A vs. B. If `settings$endptscale=="log"` in the `fit` object, this will be back-transformed to a percent difference scale.

`se` The estimated standard error of the difference `estimate`. If `settings$endptscale=="log"` in the `fit` object, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

`lowerci` The lower 100 * (1-`alpha`) % confidence limit of the difference `estimate`. With the default `alpha=0.05`, this is 95%. If `settings$endptscale=="log"` in the `fit` object, the confidence limit is first computed in the logarithmic scale of analysis, and then back-transformed to a percent difference scale.

`upperci` The upper 100 * (1-`alpha`) % confidence limit of the difference `estimate`. With the default `alpha=0.05`, this is 95%. If `settings$endptscale=="log"` in the `fit` object, the confidence limit is first computed in the logarithmic scale of analysis, and then back-transformed to a percent difference scale.

`pval` The computed p-value from the test of the difference `estimate`.

`meanA` **or** `geomeanA` The estimated mean for the left hand side "A" of the A vs. B comparison. If `settings$endptscale=="log"` in the `fit` object, this is a back-transform to the original scale, and therefore is a geometric mean, and will be labelled `geomeanA`. Otherwise it is the arithmetic mean and labelled `meanA`.

`seA` The estimated standard error of the `meanA estimate`. If `settings$endptscale=="log"` in the `fit` object, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

`meanB` **or** `geomeanB` The estimated mean for the right hand side "B" of the A vs. B comparison. If `settings$endptscale=="log"` in the `fit` object, this is a back-transform to the original scale, and therefore is a geometric mean, and will be labelled `geomeanB`. Otherwise it is the arithmetic mean and labelled `meanB`.

`seB` The estimated standard error of the `meanB estimate`. If `settings$endptscale=="log"` in the `fit` object, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

An additional column `addpct` of percent differences is added if `endptscale=="original"` and `addpct=TRUE`, as a descriptive supplement to the original scale differences that are formally estimated.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Hothorn, T., Bretz, F., Westfall, P., Heiberger, R.M., and Schuetzenmeister, A. (2010). The `multcomp` package.

Hothorn, T., Bretz, F., and Westfall, P. (2008). "Simultaneous Inference in General Parametric Models", *Biometrical Journal*, 50, 3, 346-363.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.comps0 <- comparisonsTable(canine.fit)

canine.comps1 <- comparisonsTable(canine.fit,  mcadjust=TRUE,
                                  type="allgroupstocontrol", refgrp="CC")


data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
```

```
                                          logscale=TRUE)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

gmcsfcens.comps <- comparisonsTable(gmcsfcens.fit)
```

---

comparisonsTable.cgPairedDifferenceFit

Create a table of comparisons between two groups with the cgPairedDifferenceFit object

---

Description

Create a table of comparisons based on the cgPairedDifferenceFit object. A cgPairedDifference-ComparisonsTable class object is created.

Usage

```
## S4 method for signature 'cgPairedDifferenceFit'
comparisonsTable(fit, type="pairwisereflect",
 alpha=0.05, addpct=FALSE, display="print",  ...)
```

Arguments

`fit`	An object of class `cgPairedDifferenceFit`.
`type`	Can be one of two values: `"pairwisereflect"` The default value, it calculates and lists both pairwise comparison orders. In other words, Groups A vs. B and B vs. A will be included. `"pairwise"` Calculates and lists only one of the two possible pairwise comparison orders. Ib other words, once A vs. B is specified, then the reflective B vs. A is not included. So the number of comparisons is half that produced by `"pairwisereflect"`. The ordering of group levels in the `fit` object is used to determine which ordering is included and which is not. If both orderings are of interest, such as for `settings$endptscale=="log"` in the fit objects, use the `"pairwisereflect"` value above.
`alpha`	Significance level, by default set to `0.05`.
`addpct`	Only relevant if `settings$endptscale=="original"` in the fit object. An column of percent differences is added for the comparisons, as a descriptive supplement to the original scale differences that are formally estimated.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `cgPairedDifferenceComparisonsTable` object, which is a formatted text output of the table(s).

`"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired.

`"show"` Calls the default showDefault method, which will just print out the `cgPairedDifferenceComparisonsTable` components.

`...` Additional arguments. Only one is currently valid:

`model` For `cgPairedDifferenceFit` `fit` objects that have classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible:

- `"both"` Comparison tables based on both the ordinary classical least squares and resistant & robust fits are created. This is the default when both fits are present in the `cgPairedDifferenceFit` object specified in the `fit` argument. If the resistant & robust fit is not available, this value is not relevant.
- `"olsonly"` Only a comparison table based on the ordinary classical least squares `olsfit` fit slot is performed.
- `"rronly"` Only a comparison table based on the resistant and robust `rrfit` fit slot is performed.

Value

Creates an object of class `cgPairedDifferenceComparisonsTable`, with the following slots:

`ols.comprs` The table of comparisons based on the `olsfit` component of the `cgPairedDifferenceFit`, unless `model="rronly"` is specified. In that case the slot value is `NULL`. See below for the data frame structure of the table.

`rr.comprs` The table of comparisons based on the `rrfit` component of the `cgPairedDifferenceFit` object, if a valid resistant & robust fit object is present. If `rrfit` is a simple character value of `"No fit was selected."`, or `model="olsonly"` was specified, then the value is `NULL`. See below for the data frame structure of the table.

`settings` A list of settings carried from the `cgPairedDifferenceFit` `fit` object, and the addition of some specified arguments in the method call above: `alpha`, `type`, and `addpct`. These are used for the `print.cgPairedDifferenceComparisonsTable` method, invoked for example when `display="print"`.

The data frame structure of the comparisons table in a `*.comprs` slot consists of `row.names` that specify the comparison of the form A vs. B, and these columns:

`estimate` The difference in group means in the comparison: A vs. B. If `settings$endptscale=="log"` in the `fit` object, this will be back-transformed to a percent difference scale.

`se` The estimated standard error of the difference `estimate`. If `settings$endptscale=="log"` in the `fit` object, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

`lowerci` The lower 100 * (1-alpha) % confidence limit of the difference `estimate`. With the default `alpha=0.05`, this is 95%. If `settings$endptscale=="log"` in the `fit` object, the confidence limit is first computed in the logarithmic scale of analysis, and then back-transformed to a percent difference scale.

`upperci` The upper 100 * (1-`alpha`) % confidence limit of the difference `estimate`. With the default `alpha=0.05`, this is 95%. If `settings$endptscale=="log"` in the `fit` object, the confidence limit is first computed in the logarithmic scale of analysis, and then back-transformed to a percent difference scale.

`pval` The computed p-value from the test of the difference `estimate`.

`meanA` **or** `geomeanA` The estimated mean for the left hand side "A" of the A vs. B comparison. If `settings$endptscale=="log"` in the `fit` object, this is a back-transform to the original scale, and therefore is a geometric mean, and will be labelled `geomeanA`. Otherwise it is the arithmetic mean and labelled `meanA`.

`seA` The estimated standard error of the `meanA` `estimate`. If `settings$endptscale=="log"` in the `fit` object, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

`meanB` **or** `geomeanB` The estimated mean for the right hand side "B" of the A vs. B comparison. If `settings$endptscale=="log"` in the `fit` object, this is a back-transform to the original scale, and therefore is a geometric mean, and will be labelled `geomeanB`. Otherwise it is the arithmetic mean and labelled `meanB`.

`seB` The estimated standard error of the `meanB` `estimate`. If `settings$endptscale=="log"` in the `fit` object, this estimate will be based on the Delta method, and will particularly begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

An additional column `addpct` of percent differences is added if `endptscale=="original"` and `addpct=TRUE`, as a descriptive supplement to the original scale differences that are formally estimated. This is only possible for the `model=="ols"` case, since the original arithmetic means are not estimated in the Resistant & Robust `model=="rr"` case.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(anorexiaFT)

## log scale
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1,
                                                  logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)


anorexiaFT.comps <- comparisonsTable(anorexiaFT.fit, display="none")
```

```
print(anorexiaFT.comps)

comparisonsTable(anorexiaFT.fit, model="olsonly")
comparisonsTable(anorexiaFT.fit, model="rronly")


## original scale evaluation
anorexiaFT.orig.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                      analysisname="Anorexia FT",
                                                      endptname="Weight",
                                                      endptunits="lbs",
                                                      expunitname="Patient",
                                                      digits=1,
                                                      logscale=FALSE)
anorexiaFT.orig.fit <- fit(anorexiaFT.orig.data)

comparisonsTable(anorexiaFT.orig.fit)

comparisonsTable(anorexiaFT.orig.fit, addpct=TRUE)
```

`correlationTable` *Compute Correlations*

Description

Create a table of correlations of the data in a **cg** data object.

Usage

```
correlationTable(data, display = "print", ...)
```

Arguments

`data`	A data object created and prepared (see `prepare`) using the **cg** package.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `correlationTable` object, which is a formatted text output of the table. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `correlationTable` components.
`...`	Additional arguments. Currently only one is valid: `logscale` A `logical` value, indicating whether the correlation of the log transformed data should be calculated. If `logscale` is not specified (default), its value is taken from the `data` object's component in the `settings` slot.

Value

A method-specific `correlationTable` object is returned. See the specific methods for discussion of return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`correlationTable.cgPairedDifferenceData`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
correlationTable(anorexiaFT.data)
```

`correlationTable.cgPairedDifferenceData`
Compute Correlations from a cgPairedDifferenceData object

Description

Create a table of correlations of the data in a `cgPairedDifferenceData` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceData'
correlationTable(data, display = "print", ...)
```

Arguments

`data`	A `cgPairedDifferenceData` object, typically created by `prepareCGPairedDifferenceData`.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `correlationTable` object, which is a formatted text output of the table. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the cgPairedDifferenceData object components.
`...`	Additional arguments. Currently only one is valid: `logscale` A `logical` value, indicating whether the correlation of the log transformed data should be calculated. If `logscale` is not specified, its value is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `logscale` argument.

Details

The returned table contains correlations between the paired samples. The Pearson and Spearman methods are applied with the `cor.test` function from the core **stats** package. If the `logscale` option is specified (either explicitly, or implicitly from the `cgPairedDifferenceData` object), then the Pearson calculation on the log transformed data is added.

Value

Creates an object of class `cgPairedDifferenceCorrelationTable`, with the following slots:

`contents` The table of correlations for the paired differences. See below for the data frame structure of the table.

`settings` A list of settings carried from the `cgPairedDifferenceData` data object. These are used for the `print.cgPairedDifferenceCorrelationTable` method, invoked for example when `display="print"`.

The data frame structure of the correlation table in a `contents` slot consists of `row.names` that specify the correlation method: `Pearson`, and `Spearman` if original (i.e. `logscale=FALSE`), and `Pearson Original`, `Pearson Log`, and `Spearman` if `logscale=TRUE`. The header label for the column of calculated correlations is `correlation`.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)

correlationTable(anorexiaFT.data)

## Show only correlations computed on original scale
correlationTable(anorexiaFT.data, logscale=FALSE)
```

`descriptiveTable` *Compute Descriptive Summary Statistics of Groups*

Description

Create a table of quantiles and other summary statistics of the data in a **cg** data object.

Usage

```
descriptiveTable(data, display = "print", ...)
```

Arguments

`data`	A data object created and prepared (see `prepare`) using the **cg** package.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `descriptiveTable` object, which is a formatted text output of the table. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `descriptiveTable` components.
`...`	Additional arguments. Currently only one is valid: `logscale` A `logical` value, indicating whether or not the geometric means and standard errors should be included in the summary. If `logscale` is not specified (default), its value is taken from the `data` object.

Value

A method-specific `descriptiveTable` object is returned. See the specific methods for discussion of return values.

Note

Contact <cg@billpikounis.net> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`descriptiveTable.cgOneFactorData`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
descriptiveTable(canine.data)

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
descriptiveTable(gmcsfcens.data)

## Paired Difference Data

data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)

descriptiveTable(anorexiaFT.data)
```

`descriptiveTable.cgOneFactorData` *Compute Descriptive Summary Statistics of Groups in a cgOneFactorData object*

Description

Create a table of quantiles and other summary statistics of the data in a `cgOneFactorData` object.

Usage

```
## S4 method for signature 'cgOneFactorData'
descriptiveTable(data, display = "print", ...)
```

Arguments

`data`	A `cgOneFactorData` object, typically created by `prepareCGOneFactorData`.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `descriptiveTable` object, which is a formatted text output of the table. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the cgOneFactorDescriptiveTable object components.
`...`	Additional arguments. Currently only one is valid: `logscale` A `logical` value, indicating whether or not the geometric means and standard errors should be included in the summary. If `logscale` is not specified, its value is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `logscale` argument.

Details

The returned table contains quantiles, means, sample sizes, and estimates of variability for each group. If censored data are present, the estimated quantiles accomodate that with the Kaplan-Meier method, following Gentleman and Crowley (1991) . The number of censored / incomplete and number of complete observations are also included when censored data is present in any of the groups. If the `logscale` option is specified (either explicitly, or implicitly from the `cgOneFactorData` object), then the geometric mean and geometric standard error for each group are also included. See the Value section below for details.

Value

Creates an object of class `cgOneFactorDescriptiveTable`, with the following slots:

`contents` The table of descriptive summary statistics for each group. See below for the data frame structure of the table.

`settings` A list of settings carried from the `cgOneFactorData` `data` object. These are used for the `print.cgOneFactorDescriptiveTable` method, invoked for example when `display="print"`.

The data frame structure of the descriptive table in a `contents` slot consists of `row.names` that specify the group, and these columns:

`n` The sample size of the group.

`Min` The minimum value of the group.

`25%ile` The 25th percentile of the group, estimated with the `quantile` function.

`Median` The median value of the group.

`75%ile` The 75th percentile of the group, estimated with the `quantile` function.

`Max` The maximum value of the group.

`Mean` The arithmetic mean value of the group.

`StdDev` The standard deviation value of the group.

`StdErr` The standard error value of the group.

If `logscale=TRUE`, then two additional columns are added:

`GeoMean` The geometric mean value of the group.

`SEGeoMean` The estimated standard error associated withthe geometric mean. This is calculated with the Delta Method, and will particularly lose accuracy in its useful approximation once the standard error in the log scale exceeds 0.50. A warning message is issued when this occurs.

If censored data are present in the `cgOneFactorData` object, then two more columns are added:

`ncensored` The number of censored / incomplete observations.

`ncomplete` The number of complete observations.

These two `ncensored` and `ncomplete` quantities will add up to `n` above and be placed adjacent to it.

The presence of censored observations will convert columns such as the `Min` and `Max` to character values, with the appropriate ">" and "<" symbols for right-censoring and left-censoring, respectively.

For censored data, Kaplan-Meier estimates are used for the quantiles, as proposed by Gentleman and Crowley (1991). The **survival**::`survfit` conventions are followed for interpolation of these quantiles. With enough censored data observations in a group, certain quantiles may not be estimable. If any censored observations are present, the mean, geometric mean, and associated standard errors will not be calculated. The <NA> character representation is used.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Gentleman, R.C. and Crowley, J. (1991). "Graphical Methods for Censored Data", *Journal of the American Statistical Association*, Volume 86, 678-683.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
```

```
                                        digits=1, logscale=TRUE, refgrp="CC")

descriptiveTable(canine.data)

## Remove the geometric mean and standard error columns
descriptiveTable(canine.data, logscale=FALSE)

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
descriptiveTable(gmcsfcens.data)
```

`descriptiveTable.cgPairedDifferenceData` *Compute Descriptive Summary Statistics of Groups in a cgPairedDifferenceData object*

Description

Create a table of quantiles and other summary statistics of the data in a `cgPairedDifferenceData` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceData'
descriptiveTable(data, display = "print", ...)
```

Arguments

`data`	A `cgPairedDifferenceData` object, typically created by `prepareCGPairedDifferenceData`.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `descriptiveTable` object, which is a formatted text output of the table. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the cgPairedDifferenceData object components.
`...`	Additional arguments. Currently only one is valid: `logscale` A `logical` value, indicating whether or not the geometric means, their standard errors, and ratio differences should be included in the summary. If `logscale` is not specified, its value is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `logscale` argument.

Details

The returned table contains quantiles, means, sample sizes, and estimates of variability for each group, and also for the paired differences. It also presents the same summary measures for the paired differences from the groups. If the `logscale` option is specified (either explicitly, or implicitly from the `cgPairedDifferenceData` object), then the geometric mean and geometric standard error for each of the two groups are included. Also included are summary measures of the ratio and percent forms of the paired differences. See the Value section below for details.

Value

Creates an object of class `cgPairedDifferenceDescriptiveTable`, with the following slots:

`contents`	The table of descriptive summary statistics for each group, and also for paired differences. See below for the data frame structure of the table.
`settings`	A list of settings carried from the `cgPairedDifferenceData` `data` object. These are used for the `print.cgPairedDifferenceDescriptiveTable` method, invoked for example when `display="print"`.

The data frame structure of the descriptive table in a `contents` slot consists of `row.names` that specify the group or paired difference, and these columns:

`n`	The sample size.
`Min`	The minimum value.
`25%ile`	The 25th percentile, estimated with the `quantile` function.
`Median`	The median value.
`75%ile`	The 75th percentile, estimated with the `quantile` function.
`Max`	The maximum value.
`Mean`	The arithmetic mean value.
`StdDev`	The standard deviation value.
`StdErr`	The standard error value.

If `logscale=TRUE`, then two additional columns are added:

`GeoMean`	The geometric mean value of the group.
`SEGeoMean`	The estimated standard error associated with the geometric mean. This is calculated with the Delta Method, and will particularly lose accuracy in its useful approximation once the standard error in the log scale exceeds 0.50. A warning message is issued when this occurs.

The third row of simple difference summaries has `GeoMean` and `SEGeoMean` are set to <NA>.

Fourth and fifth rows are also added with summaries of the paired ratio differences and percent differences. The `StdDev` and `StdErr` values are set to <NA> in these two rows. The `GeoMean` and `SEGeoMean` values are calculated via the the Delta Method, with the same caveats described above.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1, logscale=TRUE)

descriptiveTable(anorexiaFT.data)

## Remove the geometric mean and standard error columns,
## and the Ratio / Percent Rows, since they are no longer applicable.

descriptiveTable(anorexiaFT.data, logscale=FALSE)
```

`diffGraph` *Graph Paired Differences*

Description

Generic function to create a graph of differences from a data object created by the **cg** package.

Usage

```
diffGraph(data, ...)
```

Arguments

`data`	A data object created with a `prepare` function or method from the **cg** package. The only class of object currently valid is `cgPairedDifferenceData`, which is created by the `prepareCGPairedDifferenceData` function.
`...`	Additional arguments, depending on the specific method written for the object. Currently, there is only one such specific method; see `diffGraph.cgPairedDifferenceData` for any additional arguments that can be specified.

Details

Minimum and maximum values from ranges of data are respectively labeled in the bottom and top left corners of graph regions.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

Value

The main purpose is the side effect of graphing to the current device. See the specific methods for discussion of any return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`diffGraph.cgPairedDifferenceData`

Examples

```
data(anorexia.FT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)
diffGraph(anorexiaFT.data)

# Graph the data on the original scale instead of the log scale.
diffGraph(anorexiaFT.data, logscale=FALSE)
```

`diffGraph.pairedDifferenceData` *Graph Paired Differences from a cgPairedDifferenceData object*

Description

Create a graph of profile pairs in a `cgOnePairedDifferenceData` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceData'
diffGraph(data, ...)
```

Arguments

`data`	A `cgPairedDifferenceData` object.
`...`	Additional arguments, both *optional*. Two are currently valid: `logscale` A `logical` value, indicating whether or not the inidividual differences along the y-axis in the difference graph should be plotted on the logarithmic scale. If `logscale` is not specified, its value is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `logscale` argument. `ticklabels` A list of two components: `mod` Can be either of these two values, `"replace"` Before graphing the data, remove any automatically generated tickmarks for the y-axis, and create the tickmarks specified in the `marks` component below. `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones. `marks` A vector of tickmarks to be placed on the y-axis. Any numeric representations will be coerced to character.

Details

The individual paired differences are displayed in 3 ways side by side.

Each difference is determined by subtracting the `refgrp` value from the other group's value, for each experimental unit in the `data` object of class `cgPairedDifferenceData`.

In the middle section, the individual paired difference points are `jitter`ed, and open circles are used to alleviate potential overlap and the danger of representing multiple points as a single point.

In the right hand section, a boxplot is added, similar to to that produced by **graphics**::`boxplot.default`, but with the median shown as a "+" and the mean shown as a "o". A warning on the lack of usefulness of a boxplot is added to the graph if there are have 5 or fewer paired differences.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

The heading for the graph includes a creation of a character string: The "other" group versus the `refgrp`, e.g. `B vs. A`. Also included in the heading is the `analysisname` setting from the `cgPairedDifferenceData` object.

The label for the y-axis also includes the `B vs. A` character description of the comparison. If `logscale=TRUE`, precent differences represent the tickmarks on the log-spaced scale, since the differences in the log scale correspond to ratios in the original scale, e.g. `B / A`. Also included in the y-axis label is a character string derived from the `endptname` and `endptunits` settings in the `cgPairedDifferenceData` object. Percent differences make up the left-hand y-axis, and the corresponding Ratios make the right-hand left axis.

Minimum and maximum values from the range of the differences are respectively labeled in the bottom and top left corners of the graph region. Percentages are displayed when `logscale=TRUE`.

Value

`diffGraph.cgPairedDifferenceData` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`diffGraph.cgPairedDifferenceData`

Examples

```
data(anorexia.FT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)
diffGraph(anorexiaFT.data)

## Graph the data on the original scale instead of the log scale.
diffGraph(anorexiaFT.data, logscale=FALSE)
```

`downweightedTable` *Create a table of downweighted observations from a Resistant & Robust fit.*

Description

Create a table of downweighted observations in a resistant & robust fit with the **cg** package.

Usage

```
downweightedTable(fit, cutoffwt, display="print", ...)
```

Arguments

`fit`	A fit object created with a `fit` method from the **cg** package.
`cutoffwt`	It has no default and must be specified as a numeric between 0 and 1 exclusive. It is a threshold. All observations that fall *beneath* the threshold will be identified. For example, a `cutoffwt=0.90` will yield those observations that were downweighted by at least 10%.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `downweightedTable` object, which is a formatted text output of the table(s). `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `downweightedTable` components.
`...`	Additional arguments, depending on the specific method written for the object. See the method-specific documentation for additional details.

Value

A method-specific `downweightedTable` object is returned. See the specific methods for discussion of return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Venables, W. N. and Ripley, B. D. (2002), *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`downweightedTable.cgOneFactorFit`, **MASS**::`rlm`

Examples

```
#### One Factor data

data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)
```

```
canine.dwtable <- downweightedTable(canine.fit, cutoff=0.95)

downweightedTable(canine.fit, cutoff=0.75) ## No observation
                                           ## downweighted at least 25%

## Paired Difference data

anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

downweightedTable(anorexiaFT.fit, cutoffwt=0.25) ## No observation

downweightedTable(anorexiaFT.fit, cutoffwt=0.75) ## downweighted at least 25%
```

`downweightedTable.cgOneFactorFit` *Create a table of downweighted observations from a Resistant & Robust fit.*

Description

Create a table of downweighted observations based on a rrfit object within a cgOneFactorFit object. A cgOneFactorDownweightedTable class object is created.

Usage

```
## S4 method for signature 'cgOneFactorFit'
downweightedTable(fit, cutoffwt, display="print", ...)
```

Arguments

`fit`	A fit object of class `cgOneFactorFit`.
`cutoffwt`	It has no default and must be specified as a numeric between 0 and 1 exclusive. It is a threshold. All observations that fall *beneath* the threshold will be identified. For example, a `cutoffwt=0.90` will yield those observations that were downweighted by at least 10%.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `cgOneFactorDownweightedTable` object, which is a formatted text output of the table(s).

`"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired.

`"show"` Calls the default `showDefault` method, which will just print out the `cgOneFactorDownweightedTable` components.

`...`	Additional arguments. None are currently defined for this method.

Details

If no observations meet the cutoff criteria, a text message of the `cgOneFactorDownweightedTable` content emptiness is output instead.

The reported weights are in the scale of the observation, not the sum of squared errors representation for the likelihood. Thus they are derived from the square root of the `$w` component from a **MASS**::`rlm` fit object.

Value

An object of class `cgOneFactorDownweightedTable`, with the following slots:

`contents` A data frame where each row is an observation from the fitted data set that meets the cutoff criteria, and these columns:

- `group` The group identified from the fitted data.
- `endpoint` The observed response value.
- `weight` The weight associated to the observation from the resistant / robust fit.
- `pct down-weighted` An expression of the weight in terms of percent reduction from the maximum of 1.

If no observations meet the cutoff criteria, the `contents` slot is set to `NULL`.

`cutoffwt` Taken from the specified `cutoffwt` argument value.

`settings` A list of settings carried from the `cgOneFactorFit` object, and the addition of the specified `cutoffwt` argument in the method call above. These are used for the `print.cgOneFactorDownweightedTable` method, invoked for example when `display="print"`.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Venables, W. N. and Ripley, B. D. (2002), *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`cgOneFactorFit`, **MASS**::`rlm`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.dwtable <- downweightedTable(canine.fit, cutoff=0.95)

downweightedTable(canine.fit, cutoff=0.75) ## No observation
                                           ## downweighted at least 25%
```

`downweightedTable.cgPairedDifferenceFit`

Create a table of downweighted observations from a Resistant & Robust fit.

Description

Create a table of downweighted observations based on a rrfit object within a cgPairedDifferenceFit object. A cgPairedDifferenceDownweightedTable class object is created.

Usage

```
## S4 method for signature 'cgPairedDifferenceFit'
downweightedTable(fit, cutoffwt, display = "print",...)
```

Arguments

`fit`	A fit object of class `cgPairedDifferenceFit`.
`cutoffwt`	It has no default and must be specified as a numeric between 0 and 1 exclusive. It is a threshold. All observations that fall *beneath* the threshold will be identified. For example, a `cutoffwt=0.90` will yield those observations that were downweighted by at least 10%.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `cgPairedDifferenceDownweightedTable` object, which is a formatted text output of the table(s). `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `cgPairedDifferenceDownweightedTable` components.
`...`	Additional arguments. None are currently defined for this method.

Details

If no observations meet the cutoff criteria, a text message of the `cgPairedDifferenceDownweightedTable` content emptiness is output instead.

The reported weights are in the scale of the observation, not the sum of squared errors representation for the likelihood. Thus they are derived from the square root of the `$w` component from a **MASS**::`rlm` fit object.

Value

An object of class `cgPairedDifferenceDownweightedTable`, with the following slots:

`contents` A data frame where each row is an observation from the fitted data set that meets the cutoff criteria, and these columns:

- `expunit` The experimental unit name identified from the fitted data.
- `grp1` The observed response value from group 1.
- `grp2` The observed response value from group 2.
- `weight` The weight associated to the observation from the resistant / robust fit.
- `pct down-weighted` An expression of the weight in terms of percent reduction from the maximum of 1.
- `Simple Diff` The difference of observed response value between the two groups.
- `Ratio Diff` The percent difference of observed response value between the two groups. NOTE this only occurs when `logscale=TRUE` from the `cgPairedDifferenceFit` object `settings` in the `fit` argument.
- `Pct Diff` The percent difference of observed response value between the two groups. NOTE this only occurs when `logscale=TRUE` from the `cgPairedDifferenceFit` object `settings` in the `fit` argument.

If no observations meet the cutoff criteria, the `contents` slot is set to `NULL`.

`messages` A message when the `contents` slot is set to `NULL`.

`settings` A list of settings carried from the `cgPairedDifferenceFit` object. These are used for the `print.cgPairedDifferenceDownweightedTable` method, invoked for example when `display="print"`.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Venables, W. N. and Ripley, B. D. (2002), *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`cgPairedDifferenceFit`, **MASS**::`rlm`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

downweightedTable(anorexiaFT.fit, cutoffwt=0.25) ## No observation

downweightedTable(anorexiaFT.fit, cutoffwt=0.75) ## downweighted at least 25%
```

`errorBarGraph` *Create an Error Bar graph amongst groups*

Description

Generic function to create a Error Bar graph based on a fit by the **cg** package.

Usage

```
errorBarGraph(fit, mcadjust=FALSE, alpha = 0.05,
 cgtheme = TRUE, device="single", ...)
```

Arguments

`fit`	A fit object created by a `fit` method from the **cg** package.
`mcadjust`	Do a multiple comparisons adjustment, based on the simultaneous inference capabilities of the **multcomp** package. See Details below. The default value is `FALSE`. If `mcadjust=TRUE` is specified, there will be a delay, usually just for a few seconds, due to computing time of the critical point in order to create the error bar intervals.
`alpha`	Significance level, by default set to `0.05` so that confidence levels are 95%.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`device`	Can be one of three values: `"single"` The default, which will put all graph panels on the same device page. `"multiple"` Relevant only when more than one panel of graphs is possible. In that case, a new graphics device is generated each newly generated single-paneled graph.

	`"ask"` Relevant only when more than one panel of graphs is possible. In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that the user input confirmation is needed before the graphs are drawn.
`...`	Additional arguments, depending on the specific method written for the object. See the method-specific documentation for additional details.

Details

When `mcadjust=TRUE`, a status message of `"Some time may be needed"` `"as the critical point from the multcomp::summary.glht function call is calculated"` is displayed at the console. This computed critical point is used for all subsequent p-value and confidence interval calculations.

Value

The main purpose is the side effect of graphing to the current device. See specific methods for discussion of any return values.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Hothorn, T., Bretz, F., Westfall, P., Heiberger, R.M., and Schuetzenmeister, A. (2010). The `multcomp` R package.

Hothorn, T., Bretz, F., and Westfall, P. (2008). "Simultaneous Inference in General Parametric Models", *Biometrical Journal*, 50, 3, 346-363.

See Also

`errorBarGraph.cgOneFactorFit`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

errorBarGraph(canine.fit)

errorBarGraph(canine.fit, mcadjust=TRUE, model="olsonly")
```

`errorbargraph` *Create an error bar graph based on pairwise multiple comparisons*

Description

Creates a graph to see pairwise comparisons amongst groups. The method of Andrews, Sarner, and Snee (1980) is applied to visualizes significant differences via non-overlapping error bars.

Usage

```
errorbargraph(estimates, centralvar, critpoint, endptscale="log",
 analysisname="", endptname="", alpha=0.05, digits=NULL, approxstamp=FALSE,
 titlestamp=TRUE, offset=NULL, ticklabels=NULL, ...)
```

Arguments

`estimates`	A named vector of estimates. Each estimate element is a measure that will be the center of the error bar of the group. The name of each group must be present in the names attribute of the vector.
`centralvar`	A single variance value to be used for each group's error bar construction. In the canonical case it is the *square of* the estimated standard error of the mean `estimate` of the group, where each group also has the same standard error (and sample size). If the standard errors / variances are similar enough across the groups, the visualization may still be effective even though the error bar lengths will be approximations.
`critpoint`	The single critical value of the theoretical reference distribution. In the canonical case it is the t-distribution quantile for estimates derived from a standard linear model with homoscedastic variance. It could also reflect a multiplicity adjustment, or like the `centralvar` discussion in the previous item, it may serve as part of a visually useful approximation for other cases.
`endptscale`	Must be specified as `"log"` or `"original"`. If the default `"log"` then the y-axis will be created with a logarithmic spacing. The tick marks will be calculated accordingly and expressed in the original scale of the estimates. The `estimates` vector must already be in the logarithmic scale.
`analysisname`	*Optional*, a character text or math-valid expression that will be set for default use in graph title and table methods. The default value is the empty `""`.
`endptname`	*Optional*, a character text or math-valid expression that will be set for default use as the y-axis label of graph methods, and also used for table methods. The default value is the empty `""`.
`alpha`	Significance level, by default set to `0.05`, which equates to a 95% confidence level. This is just used for labelling purposes.
`digits`	*Optional*, for output display purposes in graphs and table methods, values will be rounded to this numeric value. Only the integers of 0, 1, 2, 3, and 4 are accepted. No rounding is done during any calculations. The default value is `NULL`, which will examine each individual `estimates` value and choose the one

	that has the maximum number of digits after any trailing zeroes are ignored. The max number of digits will be 4.
`approxstamp`	Add text to the graph that acknowledges that the error bar method is approximate.
`titlestamp`	Add text to the top margin above the graph area.
`offset`	*Optional*, if for example a numeric constant was added to all response values before calculation of the estimate as a mean, this could be used to shift the axis marks appropriately. The default value is `NULL`.
`ticklabels`	*Optional*, before graphing the data, remove any automatically generated tickmarks for the y-axis, and use these tickmarks instead. A vector of tickmarks to be placed on the y-axis. Any numeric representations will be coerced to character.
`...`	Additional arguments. None are currently used.

Details

The statistical method of Andrews, Sarner, and Snee (1980) is applied to visualizes significant differences via non-overlapping error bars. The method is exact when there are equal standard errors amongst the groups, and approximate otherwise. The method's usefulness declines as the standard errors become more disparate.

When two groups are compared, nonoverlapping error bars indicate a statistically significant pairwise difference. Conversely, if the error bars overlap, there is no such significant difference. In cases of approximation, or borderline overlap that is seen, the actual comparison needs to be consulted to judge significance with a p-value.

The minimum and maximum values across all the bar ends are added inside the plot region in blue, flush against the y-axis. The number of decimal places are determined by the `digits` value.

Value

`errorbargraph` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Warning

This function was created for internal use in the **cg** package as its use can be seen in the `errorBarGraph` methods code. Therefore any direct use of it needs to be done cautiously.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Andrews, H.P., Snee, R.D., Sarner, M.H. (1980). "Graphical Display of Means," *The American Statistician*, 34, 195-199.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                       analysisname="Canine",
                                       endptname="Prostate Volume",
                                       endptunits=expression(plain(cm)^3),
                                       digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

## Easier way: notice the camel case of the errorBarGraph call
errorBarGraph(canine.fit, model="olsonly")

## Manual way
## Instead of errorBarGraph(canine.fit, model="olsonly")
errorbargraph(estimates=canine.fit@olsfit$coef,
              centralvar=((summary(canine.fit@olsfit)$sigma^2) /
                          unique(sapply(canine, length))),
              critpoint=qt(0.975, df=canine.fit@olsfit$df.residual),
              endptscale="log",
              analysisname="Canine",
              digits=1,
              endptname=expression(paste( plain('Prostate Volume'),
                                         ' (', plain(cm)^3  ,  ')' ))
              )
```

---

errorBarGraph.cgOneFactorFit

Create an Error Bar graph amongst groups in a cgOneFactorFit object

---

Description

Creates a graph to see comparisons amongst groups based on the cgOneFactorFit object. The method of Andrews, Sarner, and Snee (1980) is applied to visualizes significant differences via non-overlapping error bars.

Usage

```
## S4 method for signature 'cgOneFactorFit'
errorBarGraph(fit, mcadjust = FALSE, alpha =0.05,
 cgtheme = TRUE, device = "single", ...)
```

Arguments

`fit`	A fit object of class `cgOneFactorFit`.

`mcadjust`	Do a multiple comparisons adjustment, based on the simultaneous inference capabilities of the **multcomp** package. See Details below. The default value is `FALSE`. If `mcadjust=TRUE` is specified, there will be a delay, usually just for a few seconds, due to computing time of the critical point in order to conduct the adjusted comparisons.
`alpha`	Significance level, by default set to `0.05`, which equates to a 95% confidence level.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`device`	Can be one of three values: `"single"` The default, which will put all graphs on the same device page. For example, when resistant & robust and classical least squares are present and `model="both"` (the default), a 2 x 1 paneled graph will be created. `"multiple"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default). In that case, a new graphics device is generated to hold the resistant & robust version, as a single-paneled graph. The classical least squares version is on the previous device. `"ask"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default). In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that the user input confirmation is needed before the graphs are drawn.
`...`	Additional arguments. Two are currently valid: `model` For `cgOneFactorFit` objects that have classical least squares `lm()` or resistant & robust `rlm()` fits, the following argument values are possible: `"both"` Error Bar graphs based on both the ordinary classical least squares and resistant & robust fits are performed. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `fit` argument. If the resistant & robust fit is not available, this value is not relevant. `"olsonly"` Only an Error Bar graph based on the ordinary classical least squares `olsfit` fit is performed. `"rronly"` Only a Error Bar Graph based on the resistant and robust `rrfit` fit is performed. For other possible `cgOneFactorFit` fit slots such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the appropriate comparisons table will be calculated for these model types. `ticklabels` A list of two components: `mod` Can be either of these two values, `"replace"` Before graphing the data, remove any automatically generated tickmarks for the y-axis, and create the tickmarks specified in the `marks` component below. `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones. `marks` A vector of tickmarks to be placed on the y-axis. Any numeric representations will be coerced to character.

Details

When `mcadjust=TRUE`, a status message of `"Some time may be needed as the critical point"` `"from the multcomp::summary.glht function call is calculated"` is displayed at the console. This computed critical point is used for all interval calculations.

The **multcomp** package provides a unified way to calculate critical points based on the comparisons of interest in a "family". Thus a user does not need to worry about choosing amongst the myriad names of multiple comparison procedures.

The `errorBarGraph.cgOneFactorFit` method is only relevant for classical least squares and resistant & robust fits in the `cgOneFactorFit` object. There is an `errorbargraph` core function that could be used for approximations in other cases like accelerated failure time or unequal variance fits.

The statistical method of Andrews, Sarner, and Snee (1980) is applied to visualize significant differences via non-overlapping error bars. The method is exact when there are equal sample sizes amongst the groups for the classical least squares case. When there are unequal group sample sizes or a resistant & robust fit is used to create the graph, the method is approximate, and this is noted in the main title section of the graph. For the unequal sample sizes, the harmonic mean is calculated to use for all the groups. The method's usefulness declines as the sample sizes become more disparate.

When two groups are compared, nonoverlapping error bars indicate a statistically significant pairwise difference. Conversely, if the error bars overlap, there is no such significant difference. In cases of approximation, or borderline overlap that is seen, the `cgOneFactorComparisonsTable` object created with `type="pairwisereflect"` or `type="pairwise"` needs to be consulted to judge significance with a p-value.

The minimum and maximum values across all the bar ends are added inside the plot region in blue, flush against the y-axis. The number of decimal places are determined by the `digits` value in the `fit$settings` slot.

If group labels along the x-axis seem to overlap in the standard horizontal form, they will be rotated 45 degrees.

Value

`errorBarGraph.cgOneFactorFit` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Andrews, H.P., Snee, R.D., Sarner, M.H. (1980). "Graphical Display of Means," *The American Statistician*, 34, 195-199.

Hothorn, T., Bretz, F., Westfall, P., Heiberger, R.M., and Schuetzenmeister, A. (2010). The `multcomp` R package.

Hothorn, T., Bretz, F., and Westfall, P. (2008). "Simultaneous Inference in General Parametric Models", *Biometrical Journal*, 50, 3, 346-363.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

errorBarGraph(canine.fit)

errorBarGraph(canine.fit, mcadjust=TRUE, model="olsonly")
```

`fit` *Fit models to data*

Description

Fit data objects prepared by the **cg** package.

Usage

```
fit(data, type, ...)
```

Arguments

`data`	A data object prepared with a `prepare` call. This will involve a data frame and additional settings.
`type`	Type of model to fit, represented by a character string.
`...`	Additional arguments, depending on the specific method written for the object.

Value

A method-specific `fit` object is returned. See the specific methods for discussion of return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

fit.cgOneFactorData

Examples

```
## Unpaired Samples, One Factor
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data, type="rr")


data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

## Paired Difference

data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)
anorexiaFT.fit <- fit(data=anorexiaFT.data, type="rr")
```

`fit.cgOneFactorData` *Fit models to a cgOneFactorData object*

Description

Fits a one-factor model based on the `cgOneFactorData` object. The created object is designed for one-factor / one-way / unpaired samples collected data, and is of class `cgOneFactorFit`.

Usage

```
## S4 method for signature 'cgOneFactorData'
fit(data, type="rr", ...)
```

Arguments

`data`	A data object of class `cgOneFactorData`.
`type`	Type of model to fit, represented by a character value. The default value is `"rr"`. The four current possibilities are: `"ols"` Only fit an ordinary, classical least squares model with the base `lm` linear model function. `"rr"` Fit a Resistant / Robust model based on M- and S-estimation, using the `rlm` function from the **MASS** package (Venables and Ripley, 2002). This is the default character value for the `type` argument. See the `...` argument and Details below for what options are available when `rlm` is used in this wrapper method. `"aft"` Fit an accelerated failure time model, using the `survreg` function from the **survival** package. If the `data` object has censored data and a slot state of `has.censored=TRUE`, then `type="aft"` will be set. See the `...` argument and Details below for what options are used when `survreg` is used in this wrapper method; in particular, the `robust` argument. `"uv"` Fit an unequal variances model, with a simple wrapper method around the `gls` from the **nlme** package. No optional arguments are passed down to `gls` through the `...` argument.
`...`	Additional arguments, both *optional*, that are allowed to be specified dependent on the choice of the `type` argument. Otherwise they have no effect on the fit: `maxIter` If `type="rr"` or `type="aft"`, then `maxIter` can be specified as a numeric postive integer. The default value of `maxIter` is `100`. For `"rr"`, this gets passed to the `maxit` argument in the `rlm` method. For `"aft"`, this gets passed to the `maxiter` argument in the `survreg` function. `sandaft` If `type="aft"`, then `sandaft` is passed to the `robust` argument of the `survreg` function. The default value of `sandaft` is `TRUE` when `type="aft"`, which applies the Huber-type (1967) sandwich estimator to the variance-covariance matrix of the group estimates.

Details

In the current version of the **cg** package, most default settings for `rlm` are kept for the `fit.cgOneFactorData` method wrapper call when `type="rr"`, with no capability to choose another value for an arguments such as `psi`, `scale.est`, and `k2`. The `method` argument is set to `"MM"`.

Analogously most `survreg` default settings are kept for the `fit.cgOneFactorData` method wrapper call when `type="aft"`, with no capability to modify the arguments. Most notably the `dist` argument is set to `"lognormal"` or `"gaussian"`, depending on whether a log scale analysis request is evident in the `cgOneFactorData` object or not, respectively.

Value

Creates an object of class `cgOneFactorFit`, with the following slots:

`olsfit` The contents of a `lm` fit to the data. This is always populated with an `lm` object no matter the choice of the `type` argument, even though it is certainly inappropriate in the `type="aft"` case.

`rrfit` The contents of a `rlm` fit to the data, housed as a `rrfit` class object. If `type="rr"` is not selected, then this is set to a simple character value of `"No fit was selected."`.

`aftfit` The contents of a `survreg` fit to the data, with some annotations, to be a `aftfit` class object. If `type="aft"` is not selected, then this is set to a simple character value of `"No fit was selected."`.

`uvfit` The contents of a `gls` fit to the data, housed as a `uvfit` class object. If `type="uv"` is not selected, then this is set to a simple character value of `"No fit was selected."`.

`settings` A list of properties carried as-is from the `data` argument object of class `cgOneFactorData`. In particular, if `zeroscore` is specified as a non-`NULL` number in the `cgOneFactorData` object in the `data` argument, then a score value near zero was derived to replace all zeroes for subsequent log-scale analyses. Alternatively, if `addconstant` is specified as a non-`NULL` number in the `cgOneFactorData` object in the `data` argument, then a value was added to shift up all observations for subsequent log-scale analyses.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Huber, P. J. (1967), "The Behavior of Maximum Likelihood Estimates Under Nonstandard Conditions", *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1, 221-233.

Venables, W. N. and Ripley, B. D. (2002), *Modern Applied Statistics with S.* Fourth edition. Springer.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")

canine.fit <- fit(data=canine.data, type="rr")


data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")
```

`fit.cgPairedDifferenceData`

Fit models to a cgPairedDifferenceData object

Description

Fits a paired difference model based on the `cgPairedDifferenceData` object. The created object is designed for paired samples collected data, and is of class `cgPairedDifferenceFit`.

Usage

```
## S4 method for signature 'cgPairedDifferenceData'
fit(data, type="rr", ...)
```

Arguments

`data`	A data object of class `cgPairedDifferenceData`.
`type`	Type of model to fit, represented by a character value. The default value is `"rr"`. The two current possibilities are: `"ols"` Only fit an ordinary, classical least squares model with the base `lm` linear model function. `"rr"` Fit a Resistant / Robust model based on M- and S-estimation, using the `rlm` function from the **MASS** package (Venables and Ripley, 2002). This is the default character value for the `type` argument. See the `...` argument and Details below for what options are available when `rlm` is used in this wrapper method.
`...`	Additional arguments, both *optional*, that are allowed to be specified dependent on the choice of the `type` argument. Otherwise they have no effect on the fit: `maxIter` If `type="rr"` then `maxIter` can be specified as a numeric postive integer. The default value of `maxIter` is `100`. This gets passed to the `maxit` argument in the `rlm` method.

Details

In the current version of the **cg** package, most default settings for `rlm` are kept for the `fit.cgPairedDifferenceData` method wrapper call when `type="rr"`, with no capability to choose another value for an arguments such as `psi`, `scale.est`, and `k2`. The `method` argument is set to `"MM"`.

Value

Creates an object of class `cgPairedDifferenceFit`, with the following slots:

`olsfit` The contents of a `lm` fit to the data. This is always populated with an `lm` object no matter the choice of the `type` argument, such as `code="rr"`.

`rrfit` The contents of a `rlm` fit to the data, housed as a `rrfit` class object. If `type="rr"` is not selected, then this is set to a simple character value of `"No fit was selected."`.

`settings` A list of properties carried as-is from the `data` argument object of class `cgPairedDifferenceData`. In particular, if `zeroscore` is specified as a non-`NULL` number in the `cgPairedDifferenceData` object in the `data` argument, then a score value near zero was derived to replace all zeroes for subsequent log-scale analyses. Alternatively, if `addconstant` is specified as a non-`NULL` number in the `cgPairedDifferenceData` object in the `data` argument, then a value was added to shift up all observations for subsequent log-scale analyses.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Venables, W. N. and Ripley, B. D. (2002), *Modern Applied Statistics with S.* Fourth edition. Springer.

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)
anorexiaFT.fit <- fit(data=anorexiaFT.data, type="rr")
```

`globalTest` *Perform a global test of significance*

Description

Generic function to perform a global test of significance on a fit by the **cg** package.

Usage

```
globalTest(fit, display="print", ...)
```

Arguments

`fit`	A fit object created by a `fit` method from the **cg** package. The only class of object currently available is `cgOneFactorFit`, which is prepared by the `fit.cgOneFactorData` method.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `globalTest` object, which is formatted text output of the test p-values. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `globalTest` components.
`...`	Additional arguments, depending on the specific method written for the object. See the method-specific documentation for additional details.

Value

A method-specific `globalTest` object is returned. See the specific methods for discussion of return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`globalTest.cgOneFactorFit`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                       analysisname="Canine",
                                       endptname="Prostate Volume",
                                       endptunits=expression(plain(cm)^3),
                                       digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.globalTest <- globalTest(canine.fit)

data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                          analysisname="cytokine",
                                          endptname="GM-CSF (pg/ml)",
                                          logscale=TRUE)
```

```
gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

gmcsfcens.globalTest <- globalTest(gmcsfcens.fit)
```

`globalTest.cgOneFactorFit` *Perform a global Test of significance with cgOneFactorFit object*

Description

Performs a global test based on the cgOneFactorFit object, to assess whether there are any significant differences amongst levels of the factor, i.e. amongst the groups. A cgOneFactorGlobalTest class object is created.

Usage

```
## S4 method for signature 'cgOneFactorFit'
globalTest(fit, display="print", ...)
```

Arguments

`fit`	A fit object of class `cgOneFactorFit`.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `globalTest.cgOneFactorFit` object, which is formatted text output of the test p-values. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `globalTest.cgOneFactorFit` components.
`...`	Additional arguments. Only one is currently valid: `model` For `cgOneFactorFit` objects that have classical least squares `lm()` or resistant & robust `rlm()` fits, the following argument values are possible: `"both"` Global tests on both the ordinary classical least squares and resistant robust fits are performed. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `fit` argument. `"olsonly"` Only a global test on the ordinary classical least squares `olsfit` fit is performed. `"rronly"` Only a global test on the resistant and robust `rrfit` fit is performed. For other possible `cgOneFactorFit` fit components such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the appropriate global test will be detected and performed for these model types.

Details

The notion of a global F test, or equivalently, of R^2, for resistant & robust linear models is murky, as no clear theoretical analogue to the ordinary classical least squares approach exists. The approach taken here is ad-hoc, which is essentially to re-fit a linear model with `lm()` and weights from the resistant & robust fit. This ad-hoc approach is taken when there are 3 or more groups.

If there are only 2 groups, then the `comparisonsTable.cgOneFactorFit` method is used with the `rlm()` model component.

Value

Creates an object of class `cgOneFactorGlobalTest`, with the following slots:

`ols.gpval`	The p-value of a global F test applied to the `olsfit` component of the `cgOneFactorFit` object, unless `model="rronly"` is specified. Will not be appropriate in the case where a valid `aftfit` component is present in the `cgOneFactorFit` object.
`rr.gpval`	The p-value of an ad-hoc global test applied to the `rrfit` component of the `cgOneFactorFit` object, if a valid resistant & robust fit object is present. See the Details section above. If `rrfit` is a simple character value of `"No fit was selected."`, or `model="olsonly"` was specified, then the value is `NULL`.
`aft.gpval`	The p-value of a global chi-square test applied to the `aftfit` component of the `cgOneFactorFit` object if a valid accelerated failure time fit object is present. If `aftfit` is a simple character value of `"No fit was selected."`, then the value is `NULL`.
`uv.gpval`	The p-value of a global F test applied to the `uvfit` component of the `cgOneFactorFit` object if a valid unequal variances fit object is present.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.globalTest <- globalTest(canine.fit)

globalTest(canine.fit, model="both")

globalTest(canine.fit, model="olsonly")
```

```
globalTest(canine.fit, model="rronly")


data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

globalTest(gmcsfcens.fit)
```

`gmcsfcens` *GM-CSF censored data set in the cg package*

Description

A data frame used to illustrate the **cg** package. It has a One Factor / One-Way / Unpaired Samples layout. It also contains left-censored values of varying degree in each of the six groups.

Usage

```
data(gmcsfcens)
```

Format

A 8-by-6 data frame with up to 8 numeric observations per group from an experiment on the following 6 groups.

`PBS/Tg 197`	phosphate buffered saline control group
`1mg/kg/Tg 197`	1 mg/kg dose
`3mg/kg/Tg 197`	3 mg/kg dose
`10/mg/kg/Tg 197`	10 mg/kg dose
`30/mg/kg/Tg 197`	30 mg/kg dose
`PBS/WT`	phosphate buffered saline control group of wild-type mice

The first five groups have transgenic (Tg197) mice subjects, a well established model to induce arthritis. The sixth group are "wild-type" mice that did not have arthritis induced. The various doses of the inner four groups are administrations of golimumab, a monoclonal antibody therapy.

The individual group values are of mode character, since some of them are represented as left-censored values such as `<82.5`. Note that two of the groups have less than 8 observations, and the corresponding cells in the data frame actually contain empty quote `""` values.

Details

The `gmcsfcens` data set that comes with the **cg** package is in `groupcolumns` format. Each column represents a group, and the observations in that group's column are the individual response values. As described above, they are character valued potentially left-censored representations.

The 6 groups are regarded as levels of one factor in the `prepareCGOneFactorData`, `fit`, and other methods in the **cg** package.

Alternative formats of this data set is contained in `gmcsfcens.listfmt`. See that help file for details, including how such formats would be read and `prepare`d by **cg**.

GM-CSF stands for Granulocyte macrophage colony stimulating factor, a type of cytokine that is important in the growth of white blood cells. It is one of the outcomes measured in the experiment described in the references section below. Therapeutic inhibition of it may be beneficial in cases where too many white blood cells are produced, such as arthritis. In other situations where white blood cell counts are low, stimulation of it is desired. In the referenced study below, GM-CSF is evaluated in the context of inflammation.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

References

Shealy, D., Cai, A., Staquet, K., Baker, A., Lacy, E., Johns, L., Vafa, O., Gunn, G., Tam, S., Sague, S., Wang, D., Brigham-Burke, M., Dalmonte, P., Emmell, E., Pikounis, B., Bugelski, P., Zhou, H., Scallon, B., Giles-Komar, J. (2010). "Characterization of Golimumab (CNTO148), a human monoclonal antibody specific for human tumor necrosis factor ", *mAbs*, Volume 2, Issue 4, 428-439.

See Also

`gmcsfcens.listfmt`, `prepareCGOneFactorData`

Examples

```
data(gmcsfcens)
str(gmcsfcens)
```

`gmcsfcens.listfmt` *GM-CSF censored data set in the cg package*

Description

A data frame used to illustrate the **cg** package. It has a One Factor / One-Way / Unpaired Samples layout. It also contains left-censored values of varying degree in each of the six groups. There are three equivalent data frame versions documented here.

Usage

```
data(gmcsfcens.listfmt1)
       data(gmcsfcens.listfmt2)
       data(gmcsfcens.listfmt3)
```

Format

A 45 row data frame with up to 8 observations per group from an experiment on the following 6 groups.

`PBS/Tg 197`	phosphate buffered saline control group
`1mg/kg/Tg 197`	1 mg/kg dose
`3mg/kg/Tg 197`	3 mg/kg dose
`10/mg/kg/Tg 197`	10 mg/kg dose
`30/mg/kg/Tg 197`	30 mg/kg dose
`PBS/WT`	phosphate buffered saline control group of wild-type mice

The first five groups have transgenic (Tg197) mice subjects, a well established model to induce arthritis. The sixth group are "wild-type" mice that did not have arthritis induced. The various doses of the inner four groups are administrations of golimumab, a monoclonal antibody therapy.

There can be either 2, 3, or 4 columns in the data frame. The above 6 items are the levels of the first column's factor, named `grp`.

`2 columns` The data frame name is `gmcsfcens.listfmt1`. The second column `endpt` contains the character observations that can represent complete observations, and also left- or right-censored ones, in the same way that `gmcsfcens` does.

`3 columns` The data frame name is `gmcsfcens.listfmt2`. The second column `endpt` contains numeric observations, and the third column `status` indicates whether the observation is complete/not censored (1), and 0 if left-censored. See `prepareCGOneFactorData` for the explanation of why the value of 0 and not 2 is required. In the example code below, the `leftcensor=TRUE` argument needs to be specified when this format version is used.

`4 columns` The data frame name is `gmcsfcens.listfmt3`. The second and third columns `endpt1` and `endpt2` contain numeric observations, and the fourth column `status` indicates whether the observation is complete/not censored (1), and 2 if left-censored. See `prepareCGOneFactorData` for the explanation of this format, and the example code below.

Details

The `gmcsfcens.listfmt*` data sets that comes with the **cg** package are in a "listed" format, detailed below.

The 6 groups are regarded as levels of one factor in the `prepareCGOneFactorData`, `fit`, and other methods in the **cg** package.

The `gmcsfcens.listfmt` data sets are alternative formats of the `gmcsfcens` data set. See that help file for details. Once a `gmcsfcens.listfmt` data set is `prepared` into a `cgOneFactorData` object, all the subsequent methods work on the object in the same way.

GM-CSF stands for Granulocyte macrophage colony stimulating factor, a type of cytokine that is important in the growth of white blood cells. It is one of the outcomes measured in the experiment described in the references section below. Therapeutic inhibition of it may be beneficial in cases where too many white blood cells are produced, such as arthritis. In other situations where white blood cell counts are low, stimulation of it is desired. In the referenced study below, GM-CSF is evaluated in the context of inflammation.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

References

Shealy, D., Cai, A., Staquet, K., Baker, A., Lacy, E., Johns, L., Vafa, O., Gunn, G., Tam, S., Sague, S., Wang, D., Brigham-Burke, M., Dalmonte, P., Emmell, E., Pikounis, B., Bugelski, P., Zhou, H., Scallon, B., Giles-Komar, J. (2010). "Characterization of Golimumab (CNTO148), a human monoclonal antibody specific for human tumor necrosis factor ", *mAbs*, Volume 2, Issue 4, 428-439.

See Also

`gmcsfcens`, `prepareCGOneFactorData`

Examples

```
data(gmcsfcens.listfmt1)
str(gmcsfcens.listfmt1)

data(gmcsfcens.listfmt2)
str(gmcsfcens.listfmt2)

data(gmcsfcens.listfmt3)
str(gmcsfcens.listfmt3)

## Analogous to prepareCGOneFactorData call on gmcsfcens data frame format,
##  subsequent methods will work for gmcsfcens.listfmt.data objects below:

## leftcensor argument can be left as default NULL
gmcsfcens.listfmt1.data <- prepareCGOneFactorData(gmcsfcens.listfmt1, format="listed",
                                                  analysisname="cytokine",
                                                  endptname="GM-CSF (pg/ml)",
                                                  logscale=TRUE)

## leftcensor=TRUE argument needs to be set
gmcsfcens.listfmt2.data <- prepareCGOneFactorData(gmcsfcens.listfmt2, format="listed",
                                                  analysisname="cytokine",
                                                  endptname="GM-CSF (pg/ml)",
                                                  logscale=TRUE,
                                                  leftcensor=TRUE)

## leftcensor argument can be left as default NULL
gmcsfcens.listfmt3.data <- prepareCGOneFactorData(gmcsfcens.listfmt3, format="listed",
```

```
                                              analysisname="cytokine",
                                              endptname="GM-CSF (pg/ml)",
                                              logscale=TRUE)
## as they do on gmcsfcens.data:

gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
```

`grpSummaryTable` *Create a table of estimated group means and variability*

Description

Create a table of estimated group means based on a fit by the **cg** package.

Usage

```
grpSummaryTable(fit, mcadjust = FALSE, alpha = 0.05, display = "print",  ...)
```

Arguments

`fit`	An fit object created with a `fit` method from the **cg** package. The only class of object currently available is `cgOneFactorFit`, which is prepared by the `fit.cgOneFactorData` method.
`mcadjust`	Do a multiple comparisons adjustment, based on the simultaneous inference capabilities of the **multcomp** package. See Details below. The default value is `FALSE`. If `mcadjust=TRUE` is specified, there will be a delay, usually just for a few seconds, due to computing time of the critical point in order to conduct the adjusted comparisons.
`alpha`	Significance level, by default set to `0.05`.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `cgOneFactorGrpSummaryTable` object, which is a formatted text output of the table(s). `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `grpSummaryTable` object components.
`...`	Additional arguments, depending on the specific method written for the object. Currently, there is only one such specific method; see `grpSummaryTable.cgOneFactorFit` for any additional arguments that can be specified.

Details

When `mcadjust=TRUE`, a status message of `"Some time may be needed as the critical point"` `"from the multcomp::summary.glht function call is calculated"` is displayed at the console. This computed critical point is used for all subsequent p-value and confidence interval calculations.

Value

A method-specific `grpSummaryTable` object is returned. See the specific methods for discussion of return values.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Hothorn, T., Bretz, F., Westfall, P., Heiberger, R.M., and Schuetzenmeister, A. (2010). The `multcomp` package.

Hothorn, T., Bretz, F., and Westfall, P. (2008). "Simultaneous Inference in General Parametric Models", *Biometrical Journal*, 50, 3, 346-363.

See Also

`grpSummaryTable.cgOneFactorFit`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.grpsumm <- grpSummaryTable(canine.fit)


data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

gmcsfcens.grpsumm <- grpSummaryTable(gmcsfcens.fit)
```

`grpSummaryTable.cgOneFactorFit` *Create a table of estimated group means and variability with a cgOneFactorFit object.*

Description

Create a table of estimated group means based on the cgOneFactorFit object. Standard errors and confidence intervals are added. A cgOneFactorGrpSummaryTable class object is created.

Usage

```
## S4 method for signature 'cgOneFactorFit'
grpSummaryTable(fit, mcadjust=FALSE, alpha=0.05, display="print",  ...)
```

Arguments

`fit`	A fit object of class `cgOneFactorFit`.
`mcadjust`	Do a multiple comparisons adjustment, based on the simultaneous inference capabilities of the **multcomp** package. See Details below. The default value is `FALSE`. If `mcadjust=TRUE` is specified, there will be a delay, usually just for a few seconds, due to computing time of the critical point in order to calculate the confidence intervals.
`alpha`	Significance level, by default set to `0.05`.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `cgOneFactorGrpSummaryTable` object, which is a formatted text output of the table(s). `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `cgOneFactorGrpSummaryTable` components.
`...`	Additional arguments. Only one is currently valid: `model` For `cgOneFactorFit` `fit` objects that have classical least squares `lm()` or resistant & robust `rlm()` fits, the following argument values are possible: `"both"` Group summary tables based on both the ordinary classical least squares and resistant & robust fits are performed. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `fit` argument. If the resistant & robust fit is not available, this value is not relevant. `"olsonly"` Only a group summary table based on the ordinary classical least squares `olsfit` fit is performed.

`"rronly"` Only a group summary table based on the resistant and robust `rrfit` fit is performed.

For other possible `cgOneFactorFit` fit components such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the appropriate group summary table will be calculated for these model types.

Details

When `mcadjust=TRUE`, a status message of `"Some time may be needed as the critical point"` `"from the multcomp::summary.glht function call is calculated"` is displayed at the console. This computed critical point is used for all subsequent p-value and confidence interval calculations.

The **multcomp** package provides a unified way to calculate critical points based on the comparisons of interest in a "family". Thus a user does not need to worry about choosing amongst the myriad names of multiple comparison procedures.

Value

Creates an object of class `cgOneFactorGrpSummaryTable`, with the following slots:

`ols.grps` The table of group estimates based on the `olsfit` component of the `cgOneFactorFit`, unless `model="rronly"` is specified. In that case the slot value is `NULL`. Will not be appropriate in the case where a valid `aftfit` component is present in the `cgOneFactorFit` object. See below for the data frame structure of the table.

`rr.grps` The table of group estimates based on the `rrfit` component of the `cgOneFactorFit` object, if a valid resistant & robust fit object is present. If `rrfit` is a simple character value of `"No fit was selected."`, or `model="olsonly"` was specified, then the value is `NULL`. See below for the data frame structure of the table.

`aft.grps` The table of group estimates based on the `aftfit` component of the `cgOneFactorFit` object if a valid accelerated failure time fit object is present. If `aftfit` is a simple character value of `"No fit was selected."`, then the value is `NULL`. See below for the data frame structure of the table.

`uv.grps` The table of group estimates based on the `uvfit` component of the `cgOneFactorFit` object if a valid unequal variances fit object is present. If `uvfit` is a simple character value of `"No fit was selected."`, then the value is `NULL`. See below for the data frame structure of the table.

`settings` A list of settings carried from the `cgOneFactorFit` `fit` object, and the addition of some specified arguments in the method call above: `alpha` and `mcadjust`. These are used for the `print.cgOneFactorGrpSummaryTable` method, invoked for example when `display="print"`.

The data frame structure of the comparisons table in a `*.comprs` slot consists of `row.names` that specify group name (factor level), and these columns:

`estimate` The estimated group mean. If `settings$endptscale=="log"` in the `fit` object, this will be back-transformed to a geometric mean.

`se` The estimated standard error of the group mean `estimate`. If `settings$endptscale=="log"` in the `fit` object, this estimate will be based on the Delta method, and will begin to be a poor approximation when the standard error in the logscale exceeds 0.50.

`lowerci` The lower 100 * (1-`alpha`) % confidence limit of the group mean `estimate`. With the default `alpha=0.05`, this is 95%. If `settings$endptscale=="log"` in the `fit` object, the confidence limit is first computed in the logarithmic scale of analysis, and then back-transformed to the original scale.

`upperci` The upper 100 * (1-`alpha`) % confidence limit of the difference `estimate`. With the default `alpha=0.05`, this is 95%. If `settings$endptscale=="log"` in the `fit` object, the confidence limit is first computed in the logarithmic scale of analysis, and then back-transformed to the original scale.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Hothorn, T., Bretz, F., Westfall, P., Heiberger, R.M., and Schuetzenmeister, A. (2010). The `multcomp` package.

Hothorn, T., Bretz, F., and Westfall, P. (2008). "Simultaneous Inference in General Parametric Models", *Biometrical Journal*, 50, 3, 346-363.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

grpSummaryTable(canine.fit)

grpSummaryTable(canine.fit, mcadjust=TRUE, model="olsonly")


data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

grpSummaryTable(gmcsfcens.fit)
```

`kmGraph` *Graph Distribution Functions of Groups*

Description

Create non-parametric survival or cumulative distribution graphs based on a data object in the **cg** package.

Usage

```
kmGraph(data, cgtheme = TRUE, distfcn = "survival", ylab = NULL,
 title = NULL, ...)
```

Arguments

`data`	A data object created using the **cg** package. The only class of object currently available is `cgOneFactorData`, which is created by the `prepareCGOneFactorData` function.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme, namely `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`distfcn`	A `character`, which may be either `"survival"` to graph the survival function, or `"cumulative"` to graph the cumulative distribution function.
`ylab`	Specify a character value for the y-axis label. The default value is `NULL`.
`title`	Specify a character value for the main title at the top of the graph. The default value is `NULL`.
`...`	Additional arguments, depending on the specific method written for the object. Currently, there is only one such specific method; see `kmGraph.cgOneFactorData` for any additional arguments that can be specified.

Details

Color assignments of the graphed step functions lines for the groups match the order of the group name factor levels. The color order is given in `cgLineColors` . The line widths are set to be thicker (`lwd=2`), and the group name label is placed near the line using `label` methodology from the **Hmisc** package.

The x-axis represents response values, and y-axis represents estimated probabilities. Minimum and maximum values from ranges of data are respectively labeled in the bottom left and right corners of graph regions.

Value

The main purpose is the side effect of graphing to the current device. See the specific methods for discussion of any return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`kmGraph.cgOneFactorData`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
kmGraph(canine.data, distfcn="cumulative")
kmGraph(canine.data, distfcn="cumulative",
        ticklabels=list(mod="add", marks=c(2)))

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
kmGraph(gmcsfcens.data, distfcn="cumulative")
kmGraph(gmcsfcens.data, distfcn="cumulative", logscale=FALSE)
```

`kmGraph.cgOneFactorData` *Graph Distribution Functions of Groups in a cgOneFactorData object*

Description

Create a non-parametric survival or cumulative distribution graph of groups of data in a `cgOneFactorData` object.

Usage

```
## S4 method for signature 'cgOneFactorData'
kmGraph(data, cgtheme = TRUE, distfcn ="survival",
 ylab = NULL, title = NULL, ...)
```

Arguments

`data`	A `cgOneFactorData` object.
`cgtheme`	A `logical` value, indicating whether to use the cg limited color scheme (white backgrounds, including shingles) or the current Trellis color scheme. The default is `TRUE`.
`distfcn`	A `character` value, which may be either the default `"survival"` to graph the survival function, or `"cumulative"` to graph the cumulative distribution function.
`ylab`	*Optional*, a `character` value to label the y-axis of the graph. If `NULL`, it defaults to either `"Probability of Survival"` or `"Cumulative Probability"`, depending on the value of `distfcn`.
`title`	*Optional*, a `character` to show as the title of the graph. If `NULL`, it defaults to either `"Nonparametric Survival Curve Estimates"` or `"Nonparametric Cumulative Distribution Function Estimates"`, depending on the value of `distfcn`.
`...`	Additional arguments. One is currently valid: `ticklabels` A list of two components: `mod` Can be either of these two values, `"replace"` Before graphing the data, remove any automatically generated tickmarks for the x-axis, and create the tickmarks specified in the `marks` component below. `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones. `marks` A vector of tickmarks to be placed on the x-axis. Any numeric representations will be coerced to character.

Details

Graph the estimated survival function or cumulative distribution for each group in a `cgOneFactorData` object. For censored data, Kaplan-Meier estimates are used. For uncensored data, the conventional step function empirical estimates are used.

Color assignments of the graphed step functions lines for the groups match the order of the group name factor levels. The color order is given in `cgLineColors`. The line widths are set to be thicker (`lwd=2`), and the group name label is placed near the line using `label` methodology from the **Hmisc** package.

The x-axis represents response values, and y-axis represents estimated probabilities. Minimum and maximum values from ranges of data are respectively labeled in the bottom left and right corners of graph regions.

The label for the x-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `endptname` and `endptunits` arguments.

The number of decimal places printed in the ticks on the x-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `digits` argument.

Value

`kmGraph.cgOneFactorFit` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                       analysisname="Canine",
                                       endptname="Prostate Volume",
                                       endptunits=expression(plain(cm)^3),
                                       digits=1, logscale=TRUE, refgrp="CC")
kmGraph(canine.data, distfcn="cumulative")
kmGraph(canine.data, distfcn="cumulative",
        ticklabels=list(mod="add", marks=c(2)))

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                          analysisname="cytokine",
                                          endptname="GM-CSF (pg/ml)",
                                          logscale=TRUE)
kmGraph(gmcsfcens.data, distfcn="cumulative")
kmGraph(gmcsfcens.data, distfcn="cumulative", logscale=FALSE)
```

`pointGraph` *Graph Individual Data Points of Groups*

Description

Generic function to create point graphs (a.k.a. dot plot, strip plot, one-dimensional scatter plot) of a data object created by the **cg** package.

Usage

```
pointGraph(data, ...)
```

Arguments

`data`	A data object created with a `prepare` function or method from the **cg** package. The only class of object currently valid is `cgOneFactorData`, which is created by the `prepareCGOneFactorData` function.
`...`	Additional arguments, depending on the specific method written for the object. Currently, there is only one such specific method; see `pointGraph.cgOneFactorData` for any additional arguments that can be specified.

Details

Individual points are `jitter`ed, and open circles are used to alleviate potential overlap and the danger of representing multiple points as a single point.

The point graph is a vertical dot plot or strip plot, with separate areas for each group in the data set, and light gray lines between the groups. For censored data, left-censored values are shown as a shallow "V", which is actually just a rotated downward "<" sign. Similarly, right-censored values are shown as a deeper "^", which is a rotated upward ">" sign.

Minimum and maximum values from ranges of data are respectively labeled in the bottom and top left corners of graph regions.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

Value

The main purpose is the side effect of graphing to the current device. See the specific methods for discussion of any return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`pointGraph.cgOneFactorData`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
pointGraph(canine.data)
```

```
# Graph the data on the original scale instead of the log scale.
pointGraph(canine.data, logscale=FALSE)

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
pointGraph(gmcsfcens.data)
```

`pointGraph.cgOneFactorData`

Graph Individual Data Points in a cgOneFactorData object

Description

Create a point graph (a.k.a. dot plot, strip plot, one-dimensional scatter plot) of the data in a `cgOneFactorData` object.

Usage

```
## S4 method for signature 'cgOneFactorData'
pointGraph(data, ...)
```

Arguments

`data`	A `cgOneFactorData` object.
`...`	Additional arguments, both *optional*. Two are currently valid:

`logscale` A `logical` value, indicating whether or not the point graph should be plotted on the logarithmic scale. If `logscale` is not specified, its value is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `logscale` argument.

`ticklabels` A list of two components:

- `mod` Can be either of these two values,
  - `"replace"` Before graphing the data, remove any automatically generated tickmarks for the y-axis, and create the tickmarks specified in the `marks` component below.
  - `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones.
- `marks` A vector of tickmarks to be placed on the y-axis. Any numeric representations will be coerced to character.

Details

If `logscale=TRUE`, the tick marks for the y-axis on the left side of the plot show original values, while the tick marks for the y-axis on the right side of the graph show base 10 log values.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

Individual points are `jitter`ed, and open circles are used to alleviate potential overlap and the danger of representing multiple points as a single point.

The point graph is a vertical dot plot or strip plot, with separate areas for each group in the data set, and light gray lines between the groups. For censored data, left-censored values are shown as a shallow "V", which is actually just a rotated downward "<" sign. Similarly, right-censored values are shown as a deeper "^", which is a rotated upward ">" sign.

The heading for the graph is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `analysisname` argument.

The label for the y-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `endptname` and `endptunits` arguments.

The number of decimal places printed in the ticks on the y-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `digits` argument.

The minimum and maximum values from the range of the data are respectively labeled in the bottom and top left corners of the graph region.

If group labels along the x-axis seem to overlap in the standard horizontal form, they will be rotated 45 degrees.

Value

`pointGraph.cgOneFactorData` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)
pointGraph(canine.data)
```

```
# Graph the data on the original scale instead of the log scale.
pointGraph(canine.data, logscale=FALSE)

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
pointGraph(gmcsfcens.data)
```

`prepare` *Prepare a cg data object from a data frame*

Description

Reads in a data frame and settings in order to create a cg Data object.

Usage

```
prepare(type, ...)
```

Arguments

`type`	Values and synonyms to create a **cg** data object. For one factor / unpaired samples, either "onefactor" or "unpairedgroups" can be specified. For paired samples, either "paireddifference" or "pairedgroups" can be used. Partial matching also allows shortened forms such as "unpaired" or "paireddiff".
`...`	Depends on the specific function that is called according to the `type` argument. For current valid calls, no ... arguments are used.

Value

See `cgOneFactorData` and `cgPairedDifferenceData` for possible valid objects that are created, dependent on the `type` and `...` arguments that are correctly specified.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`prepareCGOneFactorData`, `prepareCGPairedDifferenceData`

Examples

```
data(canine)
canine.data <- prepare(type="unpairedgroups", dfr=canine,
                       format="groupcolumns",
                       analysisname="Canine",
                       endptname="Prostate Volume",
                       endptunits=expression(plain(cm)^3),
                       digits=1, logscale=TRUE, refgrp="CC")

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepare(type="onefactor",
                          dfr=gmcsfcens, format="groupcolumns",
                          analysisname="cytokine",
                          endptname="GM-CSF (pg/ml)",
                          logscale=TRUE)

## Paired Groups
data(anorexiaFT)
anorexiaFT.data <- prepare(type="paireddiff",  ## Partial matching
                           dfr=anorexiaFT, format="groupcolumns",
                           analysisname="Anorexia FT",
                           endptname="Weight",
                           endptunits="lbs",
                           expunitname="Patient",
                           digits=1, logscale=TRUE)
```

`prepareCGOneFactorData` *Prepare data object from a data frame for One Factor / One-Way / Unpaired Samples evaluations*

Description

The function `prepareCGOneFactorData` reads in a data frame and settings in order to create a `cgOneFactorData` object. The created object is designed to have exploratory and fit methods applied to it.

Usage

```
prepareCGOneFactorData(dfr, format = "listed", analysisname = "",
 endptname = "", endptunits = "", logscale = TRUE, zeroscore = NULL,
 addconstant = NULL, rightcensor = NULL, leftcensor = NULL, digits = NULL,
 refgrp = NULL, stamps = FALSE)
```

Arguments

`dfr`	A valid data frame, see the `format` argument.

`format`	Default value of `"listed"`. Either `"listed"` or `"groupcolumns"` must be used. Abbreviations of `"l"` or `"g"`, respectively, or otherwise sufficient matching values can be used: `"listed"` At least two columns, with the factor levels in the first column and response values in the second column. If there is censored data, then two or three more columns are required, see the Details Input Data Frame section below. `"groupcolumns"` Each column must represent a group. Each group is a unique level of the one factor, so the levels of the factor make up the column headers. The values in the data frame are for the response. If the groups have unequal sample sizes, the empty cells within the data frame can have `NA`'s or be left blank. Censored values can be represented; see the Details Input Data Frame section below. Otherwise, any character data will be coerced to numeric data with possibly undesirable results.
`analysisname`	*Optional*, a character text or math-valid expression that will be set for default use in graph title and table methods. The default value is the empty `""`.
`endptname`	*Optional*, a character text or math-valid expression that will be set for default use as the y-axis label of graph methods, and also used for table methods. The default value is the empty `""`.
`endptunits`	*Optional*, a character text or math-valid expression that can be used in combination with the endptname argument. Parentheses are automatically added to this input, which will be added to the end of the endptname character value or expression. The default value is the empty `""`.
`logscale`	Apply a log-transformation to the data for evaluations. The default value is `TRUE`.
`zeroscore`	*Optional*, replace response values of zero with a derived or specified numeric value, as an approach to overcome the presence of zeroes when evaluation in the logarithmic scale (`logscale=TRUE`) is specified. The default value is `NULL`. To derive a score value to replace zero, `"estimate"` can be specified, see Details below on the algorithm used.
`addconstant`	*Optional*, add a numeric constant to all response values, as an approach to overcome the presence of zeroes when evaluation in the logarithmic scale `logscale=TRUE` is desired. The default value is `NULL`. positive numeric value can be specified to be added, or a "simple" algorthm specified to estimate a value to add. See Details secion below on the algorithm used.
`rightcensor`	*Optional*, can be specified with a numeric value where any value equal to or greater will be regarded as right censored in the evaluation. The value of `TRUE` can be used to coerce a binary status variable in the data frame to be right censored for its values.The default value is `NULL`. See the Details Input Data Frame section below for specifications and consequences.
`leftcensor`	*Optional*, can be specified with a numeric value where any value equal to or lesser will be regarded as left censored in the evaluation. The value of `TRUE` can be used to coerce a binary status variable in the data frame to be right censored for its values. The default value is `NULL`. See the Details Input Data Frame section below for specifications and consequences.

`digits`	*Optional*, for output display purposes in graphs and table methods, values will be rounded to this numeric value. Only the integers of 0, 1, 2, 3, and 4 are accepted. No rounding is done during any calculations. The default value is `NULL`, which will examine each individual data value and choose the one that has the maximum number of digits after any trailing zeroes are ignored. The max number of digits will be 4.
`refgrp`	*Optional*, specify one of the factor levels to be the "reference group", such as a "control" group. The default value is `NULL`, which will just use the first level determined in the data frame.
`stamps`	*Optional*, specify a time stamp in graphs, along with **cg** package version identification. The default value is `FALSE`.

Details

Input Data Frame The input data frame `dfr` can be of the format `"listed"` or `"groupcolumns"`. Another distinguishing characteristic is whether or not it contains censored data representations.

Censored observations can be represented by < for left-censoring and > for right-censoring. The < value refers to values less than or equal to a numeric value. For example, `<0.76` denotes a left-censored value of 0.76 or less. Similarly, `>2.02` denotes a value of 2.02 or greater for a right-censored value. There must be no space between the direction indicator and the numeric value. These representations can be used in either the `listed` or `groupcolumns` formats for `dfr`.

No interval-censored representations are currently handled when `format="groupcolumns"`.

If `format="groupcolumns"` for `dfr` is specified, then the number of columns must equal the number of groups, and any censored values must follow the < and > representations. The individual group values are of mode character, since any censored values will be represented for example as `<0.76` or `>2.02`. If any of the groups have less number of observations than any others, i.e. there are unequal sample sizes, then the corresponding "no data" cells in the data frame need to contain empty quote `""` values.

If `format="listed"` for `dfr` is specified, then there may be anywhere from two to four columns for an input data frame.

two columns The first column has the group levels to define the factor, and the second column contains the response values. Censored representations of < and > can be used here. One or both of `rightcensor` or `leftcensor` may also be specified as a number. If a number is specified for `rightcensor`, then all values in the second column equal to this value will be processed as right-censored. Analogously, if a number is specified for `leftcensor`, then all values in the second column equal to this value will be processed as left-censored. **WARNING:** This should be used cautiously to make sure the equality occurs as desired. This convention is designed for simple Type I censoring scenarios.

three columns Like the two column case, the first column has the group levels to define the factor, and the second column contains the response values, which will all be coerced to numeric. Any censoring information must be specified in the third column. Borrowing the convention of `Surv` from the **survival** package, `0`=right censored, `1`=no censoring, and `2`=left censored. If `rightcensor=NULL` and `leftcensor=NULL` are left as defaults in the call, and values of 0, 1, and 2 are all represented, then the processing will create a suitable data frame `dfru` for modeling that the canonical `survreg` function understands.

However, if 0 and 1 are the only specified values in the third censoring status column, then one of `rightcensor=TRUE` or `leftcensor=TRUE` must be specified, but NOT both, or an error message will occur. A column of all 1's or all 0's will also raise an error message.

four columns Like the two column case, the first column has the group levels to define the factor. The second and third columns need to have numeric response information, and the fourth column needs to have censoring status. This is the most general representation, where any combination of left-censoring, right-censoring, and interval-censoring is permitted. The `rightcensor` and `leftcensor` input arguments are ignored and set to `NULL`. **IMPORTANT:** The convention of `Surv` from the **survival** package, 0=right censored, 1=no censoring, and 2=left censored, 3=interval censored, and `type="interval"`, is followed. For status=0, 1, and 2, the second and third columns match in value, so that the status variable in the fourth column distinguishes the lower and upper bounds for the right-censored (0) and left-censored (2) cases. For status=3, the two values differ to define the interval boundaries. The processing will create a suitable data frame `dfru` for modeling that the canonical `survreg` and `survfit` functions from the **survival** package understand.

zeroscore If `zeroscore="estimate"` is specified, a number close to zero is derived to replace all zeroes for subsequent log-scale analyses. A spline fit (using `spline` and `method="natural"`) of the log of the response vector on the original response vector is performed. The zeroscore is then derived from the log-scale value of the spline curve at the original scale value of zero. This approach comes from the concept of arithmetic-logarithmic scaling discussed in Tukey, Ciminera, and Heyse (1985).

addconstant If `addconstant="simple"` or `addconstant="VR"` is specified, a number is derived and added to all response values.

`"simple"` Taken from the "white" book on S (Chambers and Hastie, 1992), page 68. The range (`max - min`) of the response values is multiplied by `0.0001` to derive the number to add to all the response values.

`"VR"` Based on the `logtrans` function discussed in Venables and Ripley (2002), pages 171-172 and available in the **MASS** package. The algorithm applies a Box-Cox profile likelihood approach with a log scale translation model.

Value

A `cgOneFactorData` object is returned, with the following slots:

`dfr`	The original input data frame that is the specified value of the `dfr` argument in the function call.
`dfru`	Processed version of the input data frame, which will be used for the various evaluation methods.
`fmt.dfru`	A list version of the input data frame, which will only differ from the `dfr` value if the input data frame was specified in the `groupcolumns` format.
`has.censored`	Boolean `TRUE` or `FALSE` on whether there are any censored data observations.
`settings`	A list of properties associated with the data frame: `analysisname` Drawn from the input argument value of `analysisname`. `endptname` Drawn from the input argument value of `endptname`.

`endptunits` Drawn from the input argument value of `endptunits`.

`endptscale` Has the value of `"log"` if `logscale=TRUE` and `"original"` if `logscale=FALSE`.

`zeroscore` Has the value of `NULL` if the input argument was `NULL`. Otherwise has the derived (from `zeroscore="estimate"`) or specified numeric value.

`addconstant` Has the value of `NULL` if the input argument was `NULL`. Otherwise has the specified numeric value.

`rightcensor` Has the value of the input argument `rightcensor` or is set to `NULL` if no censored observations are determined.

`leftcensor` Has the value of the input argument `leftcensor` or is set to `NULL` if no censored observations are determined.

`digits` Has the value of the input argument `digits` or is set to the determined value of digits from the input data. Will be an integer of 0, 1, 2, 3, or 4.

`grpnames` Determined from the single factor identified of the group names. The order is determined by their first occurence in the input data frame `dfr`.

`refgrp` Drawn from the input argument of `refgrp`.

`stamps` Drawn from the input argument of `stamps`.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Tukey, J.W., Ciminera, J.L., and Heyse, J.F. (1985). "Testing the Statistical Certainty of a Response to Increasing Doses of a Drug," *Biometrics*, Volume 41, 295-301.

Chambers, J.M, and Hastie, T.R. (1992), *Statistical Modeling in S.* Chapman & Hall/CRC.

Venables, W. N., and Ripley, B. D. (2002), *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

`Surv`, `canine`, `gmcsfcens`, `prepare`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")

## Censored Data
```

```
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
```

---

prepareCGPairedDifferenceData

Prepare data object from a data frame for Paired Samples evaluations

---

Description

The function `prepareCGPairedDifferenceData` reads in a data frame and settings in order to create a `cgPairedDifferenceData` object. The created object is designed to have exploratory and fit methods applied to it.

Usage

```
prepareCGPairedDifferenceData(dfr, format = "listed", analysisname = "",
 endptname = "", endptunits = "", logscale = TRUE, zeroscore = NULL,
 addconstant = NULL, digits = NULL, expunitname= "",
 refgrp = NULL, stamps = FALSE)
```

Arguments

`dfr` A valid data frame, see the `format` argument.

`format` Default value of `"listed"`. Either `"listed"` or `"groupcolumns"` must be used. Abbreviations of `"l"` or `"g"`, respectively, or otherwise sufficient matching values can be used:

- `"listed"` At least two columns, with the two and only two levels of a factor to represent the samples. These factor levels would need to be in the first column and response values in the second column. If there are three columns, then an experimental unit identifier need to be defined in the first column instead, with the second column having the two level factor, and the third column having the response values. See the Details Input Data Frame section below.
- `"groupcolumns"` At least two columns and no more than three are permitted. In the two columns case, each column must uniquely represent one of the two samples, implying a factor with two and only two levels. The levels of the factor make up the column headers. The values in the data frame are for the response. Each row assumes the pairing of the observation within an experimental unit, such as the same subject. If there are three columns, then an experimental unit identifier need to be defined in the first column instead, with the second and third column having the response values and headers to represent the two factor levels. See the Details Input Data Frame section below.

`analysisname`	*Optional*, a character text or math-valid expression that will be set for default use in graph title and table methods. The default value is the empty `""`.
`endptname`	*Optional*, a character text or math-valid expression that will be set for default use as the y-axis label of graph methods, and also used for table methods. The default value is the empty `""`.
`endptunits`	*Optional*, a character text or math-valid expression that can be used in combination with the endptname argument. Parentheses are automatically added to this input, which will be added to the end of the endptname character value or expression. The default value is the empty `""`.
`logscale`	Apply a log-transformation to the data for evaluations. The default value is `TRUE`.
`zeroscore`	*Optional*, replace response values of zero with a derived or specified numeric value, as an approach to overcome the presence of zeroes when evaluation in the logarithmic scale (`logscale=TRUE`) is specified. The default value is `NULL`. To derive a score value to replace zero, `"estimate"` can be specified, see Details below on the algorithm used.
`addconstant`	*Optional*, add a numeric constant to all response values, as an approach to overcome the presence of zeroes when evaluation in the logarithmic scale `logscale=TRUE` is desired. The default value is `NULL`. A positive numeric value can be specified to be added, or a "simple" algorthm specified to estimate a value to add. See Details secion below on the algorithm used.
`digits`	*Optional*, for output display purposes in graphs and table methods, values will be rounded to this numeric value. Only the integers of 0, 1, 2, 3, and 4 are accepted. No rounding is done during any calculations. The default value is `NULL`, which will examine each individual data value and choose the one that has the maximum number of digits after any trailing zeroes are ignored. The max number of digits will be 4.
`expunitname`	*Optional*, a character text that will be set for default use as the experimental unit label of graph methods, and also used for table methods. The default value is the empty `""`.
`refgrp`	*Optional*, specify one of the factor levels to be the “reference group”, such as a “control” group. The default value is `NULL`, which will just use the first level determined in the data frame.
`stamps`	*Optional*, specify a time stamp in graphs, along with **cg** package version identification. The default value is `FALSE`.

Details

Input Data Frame The input data frame `dfr` can be of the format `"listed"` or `"groupcolumns"`.

If `format="listed"` for `dfr` is specified, then there must be three columns for an input data frame. The first column needs to be the experimental unit identifier, the second column needs to be the group identifier, and the third is the endpoint. The first column of the listed input data format, needs to have two sets of distinct values since it is the experimental unit identifier of response pairs. The second column of the listed input data format needs to have exactly 2 distinct values since it is the group identifier.

If `format="groupcolumns"` for `dfr` is specified, then there can be two columns or three columns.

two columns The column headers specify the two paired group names. Each row contains the experimental unit of paired numeric values under those two groups. In the course of creating the `cgPairedDifferenceData` object, another column will be binded from the left and become the first column, with the column header of `expunitname` is specified, and "expunit" if the default `expunitname=""` is specified. A sequence of integers starting with 1 up to the number of pairs/rows will be generated to uniquely identify each experimental unit pair.

three columns The first column needs to be unique experimental unit identifiers of the paired numeric values in the second and third columns. The second and third column headers will be used to identify the two paired group names. Each row's second and third column needs to contain the experimental unit of paired numeric values under those two groups. The name of the first column will be assigned to the `expunitname` setting if `expunitname` is not explicity specified to something else instead of its default `expunitname=""`.

As the evaluation data set is prepared for `cgPairedDifferenceData` object, any experimental unit pairs/rows with missing values in the endpoint are flagged. This includes a check to make sure that each experimental unit identified has a complete pair of numeric observations.

zeroscore If `zeroscore="estimate"` is specified, a number close to zero is derived to replace all zeroes for subsequent log-scale analyses. A spline fit (using `spline` and `method="natural"`) of the log of the response vector on the original response vector is performed. The zeroscore is then derived from the log-scale value of the spline curve at the original scale value of zero. This approach comes from the concept of arithmetic-logarithmic scaling discussed in Tukey, Ciminera, and Heyse (1985).

addconstant If `addconstant="simple"` is specified, a number is derived and added to all response values. The approach taken is from the "white" book on S (Chambers and Hastie, 1992), page 68. The range (`max - min`) of the response values is

multiplied by `0.0001` to derive the number to add to all the response values.

Value

A `cgPairedDifferenceData` object is returned, with the following slots:

`dfr`	The original input data frame that is the specified value of the `dfr` argument in the function call.
`dfru`	Processed version of the input data frame, which will be used for the various evaluation methods.
`dfr.gcfmt`	A groupcolumns version of the input data frame with an additional column of the differences between groups, where the `regfrp` column of values is the subtrahend (second term) in the subtraction.
`settings`	A list of properties associated with the data frame: `analysisname` Drawn from the input argument value of `analysisname`. `endptname` Drawn from the input argument value of `endptname`, and set to `"Endpoint"` if input was left at the default `""`. `endptunits` Drawn from the input argument value of `endptunits`. `endptscale` Has the value of `"log"` if `logscale=TRUE` and `"original"` if `logscale=FALSE`.

`zeroscore` Has the value of `NULL` if the input argument was `NULL`. Otherwise has the derived (from `zeroscore="estimate"`) or specified numeric value.

`addconstant` Has the value of `NULL` if the input argument was `NULL`. Otherwise has the specified or derived numeric value.

`digits` Has the value of the input argument `digits` or is set to the determined value of digits from the input data. Will be an integer of 0, 1, 2, 3, or 4.

`grpnames` Of length 2 and determined from the single factor identified of the group names. The order is determined by the first occurence in the input data frame header in `dfr` and the `refgrp` specification.

`expunitname` Drawn from the input argument value of `expunitname` and processing of the data frame.

`refgrp` Drawn from the input argument of `refgrp`.

`stamps` Drawn from the input argument of `stamps`.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Tukey, J.W., Ciminera, J.L., and Heyse, J.F. (1985). "Testing the Statistical Certainty of a Response to Increasing Doses of a Drug," *Biometrics*, Volume 41, 295-301.

Chambers, J.M, and Hastie, T.R. (1992), *Statistical Modeling in S.* Chapman&Hall/CRC.

See Also

`prepare`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1, logscale=TRUE)
```

`print.cgOneFactorComparisonsTable` *Print One Factor Comparisons Table object with some format options*

Description

Print a `cgOneFactorComparisonsTable` object, which contains a table of comparisons based on the `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorComparisonsTable'
print(x, digits = NULL, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	An `cgOneFactorComparisonsTable` object, typically created by `comparisonsTable.cgOneFactorFit`.
`digits`	The number of decimal places to use in the output. If `NULL`, then the number of decimal places is taken from the `digits` value in the `settings` slot of the `cgOneFactorComparisonsTable` object.
`title`	The title printed out with the table. If `NULL`, it is set to be "Comparisons Table of" the `analysisname` value from the `settings` slot of the `cgOneFactorComparisonsTable` object.
`endptname`	The endpoint name, printed out with the table. If `NULL`, it is set to the `endptname` value from the `settings` slot of the `cgOneFactorComparisonsTable` object.
`...`	Additional arguments. Only one is currently valid: `model` For `cgOneFactorComparisonsTable` objects that have tables derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible: `"both"` Both the ordinary classical least squares and resistant robust comparisons tables are printed. This is the default when both fits are present in the `cgOneFactorComparisonsTable` object specified in the `x` argument. `"olsonly"` Only the ordinary classical least squares group comparisons table is printed. `"rronly"` Only the resistant & robust comparisons table is printed. For other possible `cgOneFactorComparisonsTable` table components such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the single table will just be printed for these model types.

Details

The smallest actual p-value that will be printed is `0.001`. Anything less than `0.001` will be displayed as < `0.001`. If you need more digits, see the `cgOneFactorComparisonsTable` object.

The object is printed using a mix of `cat` and `print` calls. See `cgOneFactorComparisonsTable` for details of the `*.comprs` and other object slots.

Value

`print.cgOneFactorComparisonsTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")

canine.fit <- fit(canine.data)

canine.comps0 <- comparisonsTable(canine.fit)

print(canine.comps0, digits=1)

canine.comps1 <- comparisonsTable(canine.fit,  mcadjust=TRUE,
                                  type="allgroupstocontrol", refgrp="CC")

print(canine.comps1, model="olsonly")
```

`print.cgOneFactorDescriptiveTable`

Print a One Factor Descriptive Table object with some format options

---

Description

Print a `cgOneFactorDescriptiveTable` object, which contains a table of quantiles and other summary statistics of the data from a `cgOneFactorData` object.

Usage

```
## S4 method for signature 'cgOneFactorDescriptiveTable'
print(x, digits = NULL, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	A `cgOneFactorDescriptiveTable` object, typically created by `descriptiveTable.cgOneFactorData`.
`digits`	The number of decimal places to use in the output. If `NULL`, then the number of decimal places is taken from the `digits` value in the `settings` slot of the `cgOneFactorDescriptiveTable` object.
`title`	The title printed out with the table. If `NULL`, it is set to be "Descriptive Table of" the `analysisname` value taken from the `settings` slot of the `cgOneFactorDescriptiveTable` object.
`endptname`	The endpoint name of the data summarized in the table. If NULL, it is set to the `endptname` value taken from the `settings` slot of the `cgOneFactorDescriptiveTable` object.
`...`	Additional arguments. None are currently defined for this method.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgOneFactorDescriptiveTable` for details of the `contents` and other object slots.

Value

`print.cgOneFactorDescriptiveTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorDescriptiveTable`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
```

```
## Next two calls are equivalent
descriptiveTable(canine.data)

print(descriptiveTable(canine.data, display="none"))

print(descriptiveTable(canine.data, display="none"),
      title="Quantiles and Summary Statistics")

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)

## Next two calls are equivalent
descriptiveTable(gmcsfcens.data, display="print")
print(descriptiveTable(gmcsfcens.data, display="none"))
```

`print.cgOneFactorDownweightedTable` *Print Downweighted Observations Table object with some format options*

Description

Print a `cgOneFactorDownweightedTable` object, as a table of downweighted observations in a resistant & robust fit from a `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorDownweightedTable'
print(x, digits=NULL, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	An object of class `cgOneFactorDownweightedTable`, typically created by `downweightedTable.cgOneFactorFit`.
`digits`	The number of decimal places to use in the output. If `NULL`, then the number of decimal places is taken from the `cgOneFactorDownweightedTable` object.
`title`	The title printed out with the p-value. If `NULL`, it is set to be "Downweighted Observations Table from Resistant & Robust Fit" of the `analysisname` value in the settings of the `cgOneFactorDownweightedTable` object.
`endptname`	The endpoint name, printed out with the p-value. If `NULL`, it is set to the `endptname` value in the `cgOneFactorDownweightedTable` object.
`...`	Additional arguments. None are currently defined for this method.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgOneFactorDownweightedTable` for details of the `contents` and other object slots.

Value

`print.cgOneFactorDownweightedTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console. If any observations meet the cutoff criteria, a table is displayed.

If no observations meet the cutoff criteria, a text message of table emptiness is displayed instead.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorDownweightedTable`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.dwtable <- downweightedTable(canine.fit, cutoff=0.95)

downweightedTable(canine.fit, cutoff=0.75) ## No observation
                                           ## downweighted at least 25%
```

`print.cgOneFactorFit` *Print One Factor Model Fit object with some format options*

Description

Print a `cgOneFactorFit` object, which contains fitted model information.

Usage

```
## S4 method for signature 'cgOneFactorFit'
print(x, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	An `cgOneFactorFit` object.
`title`	The title printed out with the fitted model information. If `NULL`, it is set to be "Fitted Models of" the `analysisname` value in the `settings` slot of the `cgOneFactorFit` object.
`endptname`	The endpoint name, printed out with the fitted model information. If `NULL`, it is set to the `endptname` value in the `settings` slot of the `cgOneFactorFit` object.
`...`	Additional arguments. Only one is currently valid: `model` For `cgOneFactorFit` objects that have output derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible: `"both"` Both the ordinary classical least squares and resistant & robust model fits are printed. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `x` argument. `"olsonly"` Only the ordinary classical least squares model fit is printed. `"rronly"` Only the resistant & robust model fit is printed. For other possible `cgOneFactorFit` components such as accelerated failure time or unequal variance model fits, the `model` argument is not relevant, and the single model fit will just be printed for these model types.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgOneFactorFit` for details of the `*fit` and other object slots.

This method simply echoes print methods for individual fit classes, such as `lm` and `rlm`.

Note that `show` is an alias for `print` for this method. A `showObj.cgOneFactorFit` method is defined to display the raw form of the object.

Value

`print.cgOneFactorFit` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorFit`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

print(canine.fit)
```

`print.cgOneFactorGlobalTest`

Print One Factor Global F-test object with some format options

Description

Print a `cgOneFactorGlobalTest` object, which contains global F-test p-value information taken from a `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorGlobalTest'
print(x, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	An `cgOneFactorGlobalTest` object, typically created by `globalTest.cgOneFactorFit`.
`title`	The title printed out with the p-value. If `NULL`, it is set to be "Group Test P-value of" the `analysisname` value in the `settings` slot of the `cgOneFactorGlobalTest` object.
`endptname`	The endpoint name, printed out with the p-value. If `NULL`, it is set to the `endptname` value in the `settings` slot of the `cgOneFactorGlobalTest` object.
`...`	Additional arguments. Only one is currently valid: `model` For `cgOneFactorGlobalTest` objects that have p-values derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible: `"both"` Both the ordinary classical least squares and resistant & robust p-values are printed. This is the default when both fits are present in the `cgOneFactorGlobalTest` object specified in the `x` argument. `"olsonly"` Only the ordinary classical least squares p-value is printed. `"rronly"` Only the resistant & robust approximated p-value is printed. For other possible `cgOneFactorGlobalTest` p-value components such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the single p-value will just be printed for these model types.

Details

The smallest actual p-value that will be printed is `0.001`. Anything less than `0.001` will be displayed as < `0.001`. If you need more digits, see the `cgOneFactorGlobalTest` object.

The notion of a global F test, or equivalently, of R^2, for resistant & robust linear models is murky, as no clear theoretical analogue to the ordinary classical least squares approach exists. See `cgOneFactorGlobalTest` for details, and regard the output p-value here as ad-hoc.

The object is printed using a mix of `cat` and `print` calls. See `cgOneFactorGlobalTest` for details of the `*.gpval` and other object slots.

Value

`print.cgOneFactorGlobalTest` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorGlobalTest`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.global <- globalTest(canine.fit)

print(canine.global)
```

`print.cgOneFactorGrpSummaryTable`

Print One Factor Group Summary Table object with some format options

---

Description

Print a `cgOneFactorGrpSummaryTable` object, which contains a table of group means and variability based on the `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorGrpSummaryTable'
print(x, digits = NULL, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	An `cgOneFactorGrpSummaryTable` object, typically created by `grpSummaryTable.cgOneFactorFit`.
`digits`	The number of decimal places to use in the output. If `NULL`, then the number of decimal places is taken from the `digits` value in the `settings` slot of the `cgOneFactorGrpSummaryTable` object.
`title`	The title printed out with the table. If `NULL`, it is set to be "Group Summary Table of" the `analysisname` value from the `settings` slot of the `cgOneFactorGrpSummaryTable` object.
`endptname`	The endpoint name, printed out with the table. If `NULL`, it is set to the `endptname` value from the `settings` slot of the `cgOneFactorGrpSummaryTable` object.
`...`	Additional arguments. Only one is currently valid: `model` For `cgOneFactorGrpSummaryTable` objects that have tables derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible: `"both"` Both the ordinary classical least squares and resistant robust comparisons tables are printed. This is the default when both fits are present in the `cgOneFactorGrpSummaryTable` object specified in the `x` argument. `"olsonly"` Only the ordinary classical least squares comparisons table is printed. `"rronly"` Only the resistant and robust comparisons table is printed. For other possible `cgOneFactorGrpSummaryTable` table components such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the single table will just be printed for these model types.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgOneFactorGrpSummaryTable` for details of the `*.grps` and other object slots.

Value

`print.cgOneFactorGrpSummaryTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorGrpSummaryTable`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.grpsumm <- grpSummaryTable(canine.fit)

print(canine.grpsumm, digits=2)
```

`print.cgOneFactorSampleSizeTable`
Print a One Factor Sample Size Table object with some format options

Description

Print a `cgOneFactorSampleSizeTable` object, which contains a table of sample size estimates based on a `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorSampleSizeTable'
print(x, title=NULL, endptname=NULL, ...)
```

Arguments

`x`	A `cgOneFactorSampleSizeTable` object, typically created by `samplesizeTable.cgOneFactorFit`.
`title`	The title for the table. If `NULL`, it is set to be "Sample Size Table from" concatenated to `planningname` value in the `settings` slot of the `cgOneFactorSampleSizeTable` object.
`endptname`	The endpoint name, printed out with the table. If `NULL`, it is set to the `endptname` value from the `settings` slot of the `cgOneFactorSampleSizeTable` object.
`...`	Additional arguments. Currently one is valid: `model` For `cgOneFactorComparisonsTable` objects that have tables derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible: `"both"` Both the ordinary classical least squares and resistant robust sample size tables are printed. This is the default when both fits are present in the `cgOneFactorSampleSizeTable` object specified in the `x` argument. If only one of the two tables were computed, then only that single table is printed. `"olsonly"` Only the ordinary classical least squares group summary table is printed. `"rronly"` Only the resistant & robust group summary table is printed.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgOneFactorSampleSizeTable` for details of the `*.sstable` and other object slots.

Value

`print.cgOneFactorSampleSizeTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorSampleSizeTable`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.samplesize <- samplesizeTable(canine.fit, direction="increasing",
                                     model="olsonly",
                                     mmdvec=c(10, 25, 50, 75, 100), display="none")
print(canine.samplesize)
```

`print.cgPairedDifferenceComparisonsTable` *Print Paired Difference Comparisons Table object with some format options*

Description

Print a `cgPairedDifferenceComparisonsTable` object, which contains a table of comparisons based on the `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceComparisonsTable'
print(x, digits = NULL, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	An `cgPairedDifferenceComparisonsTable` object, typically created by `comparisonsTable.cgPairedDifferenceFit`.
`digits`	The number of decimal places to use in the output. If `NULL`, then the number of decimal places is taken from the `digits` value in the `settings` slot of the `cgPairedDifferenceComparisonsTable` object.
`title`	The title printed out with the table. If `NULL`, it is set to be "Comparisons Table of" the `analysisname` value taken from the `settings` slot of the `cgPairedDifferenceComparisonsTable` object.
`endptname`	The endpoint name, printed out with the table. If `NULL`, it is set to the `endptname` value from the `settings` slot of the `cgPairedDifferenceComparisonsTable` object.
`...`	Additional arguments. Only one is currently valid: `model` For `cgPairedDifferenceComparisonsTable` objects that have tables derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible:

`"both"` Both the ordinary classical least squares and resistant & robust comparisons tables are printed. This is the default when both fits are present in the `cgPairedDifferenceComparisonsTable` object specified in the `x` argument.

`"olsonly"` Only the ordinary classical least squares group comparisons table is printed.

`"rronly"` Only the resistant & robust comparisons table is printed.

Details

The smallest actual p-value that will be printed is `0.001`. Anything less than `0.001` will be displayed as < `0.001`. If you need more digits, see the `cgPairedDifferenceComparisonsTable` object.

The object is printed using a mix of `cat` and `print` calls. See `cgPairedDifferenceComparisonsTable` for details of the `*.comprs` and other object slots.

Value

`print.cgPairedDifferenceComparisonsTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(anorexiaFT)

anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)

anorexiaFT.fit <- fit(anorexiaFT.data)

anorexiaFT.comps <- comparisonsTable(anorexiaFT.fit)

print(anorexiaFT.comps, digits=2)

print(anorexiaFT.comps, model="olsonly")
```

`print.cgPairedDifferenceCorrelationTable`

Print a Paired Difference Correlation Table object with some format options

---

Description

Print a `cgPairedDifferenceCorrelationTable` object, which contains a table of correlations of the data from a `cgPairedDifferenceData` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceCorrelationTable'
print(x, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	A `cgPairedDifferenceCorrelationTable` object, typically created by `correlationTable.cgPairedDifferenceData`.
`title`	The title printed out with the table. If `NULL`, it is set to be "Correlation Table of" the `analysisname` value taken from the `settings` slot of the `cgPairedDifferenceCorrelationTable` object.
`endptname`	The endpoint name of the data summarized in the table. If NULL, it is set to the `endptname` value taken from the `settings` slot of the `cgPairedDifferenceCorrelationTable` object.
`...`	Additional arguments. None are currently defined for this method.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgPairedDifferenceCorrelationTable` for details of the `contents` and other object slots.

Two decimal places are used in the display of the correlations.

Value

`print.cgPairedDifferenceCorrelationTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceCorrelationTable`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1,
                                                  logscale=TRUE)
## Next two calls are equivalent
correlationTable(anorexiaFT.data)

print(correlationTable(anorexiaFT.data, display="none"))

## A change in title
print(correlationTable(anorexiaFT.data, display="none"),
      title="Correlations")
```

`print.cgPairedDifferenceDescriptiveTable`

Print a Paired Difference Descriptive Table object with some format options

---

Description

Print a `cgPairedDifferenceDescriptiveTable` object, which contains a table of quantiles and other summary statistics of the data from a `cgPairedDifferenceData` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceDescriptiveTable'
print(x, digits = NULL, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	A `cgPairedDifferenceDescriptiveTable` object, typically created by `descriptiveTable.cgPairedDifferenceData`.
`digits`	The number of decimal places to use in the output. If `NULL`, then the number of decimal places is taken from the `digits` value in the `settings` slot of the `cgPairedDifferenceDescriptiveTable` object.
`title`	The title printed out with the table. If `NULL`, it is set to be "Descriptive Table of" the `analysisname` value taken from the `settings` slot of the `cgPairedDifferenceDescriptiveTable` object.

`endptname`	The endpoint name of the data summarized in the table. If NULL, it is set to the `endptname` value taken from the `settings` slot of the `cgPairedDifferenceDescriptiveTable` object.
`...`	Additional arguments. None are currently defined for this method.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgPairedDifferenceDescriptiveTable` for details of the `contents` and other object slots.

Value

`print.cgPairedDifferenceDescriptiveTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceDescriptiveTable`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
## Next two calls are equivalent
descriptiveTable(anorexiaFT.data)

print(descriptiveTable(anorexiaFT.data, display="none"))

## A change in title
print(descriptiveTable(anorexiaFT.data, display="none"),
      title="Quantiles and Summary Statistics")
```

`print.cgPairedDifferenceDownweightedTable`
Print Downweighted Observations Table object with some format options

---

Description

Print a `cgPairedDifferenceDownweightedTable` object, as a table of downweighted observations in a resistant & robust fit from a `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceDownweightedTable'
print(x, digits=NULL, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	An object of class `cgPairedDifferenceDownweightedTable`, typically created by `downweightedTable.cgPairedDifferenceFit`.
`digits`	The number of decimal places to use in the output. If `NULL`, then the number of decimal places is taken from the `cgPairedDifferenceDownweightedTable` object.
`title`	The title printed out with the p-value. If `NULL`, it is set to be "Downweighted Observations Table from Resistant & Robust Fit" of the `analysisname` value in the settings of the `cgPairedDifferenceDownweightedTable` object.
`endptname`	The endpoint name, printed out with the p-value. If `NULL`, it is set to the `endptname` value in the `cgPairedDifferenceDownweightedTable` object.
`...`	Additional arguments. None are currently defined for this method.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgPairedDifferenceDownweightedTable` for details of the `contents` and other object slots.

Value

`print.cgPairedDifferenceDownweightedTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console. If any observations meet the cutoff criteria, a table is displayed.

If no observations meet the cutoff criteria, a text message of table emptiness is displayed instead.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceDownweightedTable`

Examples

```
data(anorexiaFT)

anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1,
                                                  logscale=TRUE)

anorexiaFT.fit <- fit(anorexiaFT.data)

anorexiaFT.dw <- downweightedTable(anorexiaFT.fit, cutoffwt=0.25, display='none')

print(anorexiaFT.dw)  ## No observation
```

---

print.cgPairedDifferenceFit

Print One Factor Model Fit object with some format options

Description

Print a `cgPairedDifferenceFit` object, which contains fitted model information.

Usage

```
## S4 method for signature 'cgPairedDifferenceFit'
print(x, title = NULL, endptname = NULL, ...)
```

Arguments

`x`	An `cgPairedDifferenceFit` object.
`title`	The title printed out with the fitted model information. If `NULL`, it is set to be "Fitted Models of" the `analysisname` value in the `settings` slot of the `cgPairedDifferenceFit` object.
`endptname`	The endpoint name, printed out with the fitted model information. If `NULL`, it is set to the `endptname` value in the `settings` slot of the `cgPairedDifferenceFit` object.

`...` Additional arguments. Only one is currently valid:

`model` For `cgPairedDifferenceFit` objects that have output derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible:

- `"both"` Both the ordinary classical least squares and resistant & robust model fits are printed. This is the default when both fits are present in the `cgPairedDifferenceFit` object specified in the `x` argument.
- `"olsonly"` Only the ordinary classical least squares model fit is printed.
- `"rronly"` Only the resistant & robust model fit is printed.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgPairedDifferenceFit` for details of the `*fit` and other object slots.

This method simply echoes print methods for individual fit classes, such as `lm` and `rlm`.

Note that `show` is an alias for `print` for this method. A `showObj.cgPairedDifferenceFit` method is defined to display the raw form of the object.

Value

`print.cgPairedDifferenceFit` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorFit`, `cgPairedDifferenceFit`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1, logscale=TRUE)
anorexiaFT.fit <- fit(data=anorexiaFT.data, type="rr")

print(anorexiaFT.fit)
```

`print.cgPairedDifferenceSampleSizeTable` *Print a Paired Difference Sample Size Table object with some format options*

Description

Print a `cgPairedDifferenceSampleSizeTable` object, which contains a table of sample size estimates based on a `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceSampleSizeTable'
print(x, title=NULL, endptname=NULL, ...)
```

Arguments

`x`	A `cgPairedDifferenceSampleSizeTable` object, typically created by `samplesizeTable.cgPairedDifferenceFit`.
`title`	The title for the table. If `NULL`, it is set to be "Sample Size Table from" concatenated to `planningname` value in the `settings` slot of the `cgPairedDifferenceSampleSizeTable` object.
`endptname`	The endpoint name, printed out with the table. If `NULL`, it is set to the `endptname` value from the `settings` slot of the `cgPairedDifferenceSampleSizeTable` object.
`...`	Additional arguments. None are currently defined.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgPairedDifferenceSampleSizeTable` for details of the `*.sstable` and other object slots.

Value

`print.cgPairedDifferenceSampleSizeTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceSampleSizeTable`

Examples

```
data(anorexiaFT)

anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)

anorexiaFT.fit <- fit(anorexiaFT.data)

## The interest is in increased weight for the anorexia FT
## (family treatment) group of patients

anorexiaFT.samplesize <- samplesizeTable(anorexiaFT.fit, direction="increasing",
                         mmdvec=c(5, 10, 15, 20), display="none")

print(anorexiaFT.samplesize)

## The above two calls produce the same screen output as
samplesizeTable(anorexiaFT.fit, direction="increasing",
                mmdvec=c(5, 10, 15, 20))

## since the default in the call is display="print"
```

`print.cgPairedDifferenceVarianceTable`

Print a Paired Difference Variance Table object with some format options

---

Description

Print a `cgPairedDifferenceVarianceTable` object, which contains a table of variances from a `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceVarianceTable'
print(x, digits = NULL, title = NULL, endptname = NULL, ...)
```

Arguments

x	A `cgPairedDifferenceVarianceTable` object, created by `varianceTable.cgPairedDifferenceFit`.

`digits`	The number of decimal places to use in the output, after any leading zeroes right of the decimal point. If `NULL`, then the number of decimal places is taken from the `digits` value in the `settings` slot of the `cgPairedDifferenceComparisonsTable` object.
`title`	The title printed out with the table. If `NULL`, it is set to be "Variance Components Table of" the `analysisname` value taken from the `settings` slot of the `cgPairedDifferenceVarianceTable` object.
`endptname`	The endpoint name of the data summarized in the table. If NULL, it is set to the `endptname` value taken from the `settings` slot of the `cgPairedDifferenceVarianceTable` object.
`...`	Additional arguments. None are currently defined for this method.

Details

The object is printed using a mix of `cat` and `print` calls. See `cgPairedDifferenceVarianceTable` for details of the `contents` and other object slots.

Two decimal places (after any leading zeroes) are used by default in the display of the variances.

As described in `cgPairedDifferenceVarianceTable`, the table displays a decomposition of the total variance into its within-experimental unit and between-experimential unit variance components. The variance estimates are provided in the first column, and the relative percents of these two components are in the second column. The third column is the square root of the first column of variances, to provide `Spread/StdDev` values in the units of the endpoint.

Below the printed table is a series of Notes. The first note narrates the estimated gain in sensitivity from using a paired groups design instead of an unpaired groups design. The gains are expressed in terms of reduced experimental unit sample size.

The label portion "experimental unit" in the printed output is replaced by the `expunitname` component of the `settings` slot of the `cgPairedDifferenceVarianceTable` object.

Value

`print.cgPairedDifferenceVarianceTable` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceVarianceTable`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

## Next two calls are equivalent
varianceTable(anorexiaFT.fit)

print(varianceTable(anorexiaFT.fit, display="none"))

## A change in title
print(varianceTable(anorexiaFT.fit, display="none"),
      title="Estimated Variances")

## Show three digits in display
print(varianceTable(anorexiaFT.fit), digits=3)
```

`profileGraph` *Graph Profiles of Experimental Units*

Description

Generic function to create a graph of experimental unit profiles of a data object created by the **cg** package.

Usage

```
profileGraph(data, ...)
```

Arguments

`data`	A data object created with a `prepare` function or method from the **cg** package. The only class of object currently valid is `cgPairedDifferenceData`, which is created by the `prepareCGPairedDifferenceData` function.
`...`	Additional arguments, depending on the specific method written for the object. Currently, there is only one such specific method; see `profileGraph.cgPairedDifferenceData` for any additional arguments that can be specified.

Details

Individual points are jittered, and open circles are used to alleviate potential overlap and the danger of representing multiple points as a single point.

The profile graph for paired difference data is the simplest of profiles as each experimental unit has exactly two points connected by a straight line. Labels for the experimental units are added for identification.

Minimum and maximum values from ranges of data are respectively labeled in the bottom and top left corners of graph regions.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

Value

The main purpose is the side effect of graphing to the current device. See the specific methods for discussion of any return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`profileGraph.cgPairedDifferenceData`

Examples

```
data(anorexia.FT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1, logscale=TRUE)
profileGraph(anorexiaFT.data)

# Graph the data on the original scale instead of the log scale.
profileGraph(anorexiaFT.data, logscale=FALSE)
```

`profileGraph.cgPairedDifferenceData` *Graph Profiles of Experimental Unit Pairs in a cgPairedDifferenceData object*

Description

Create a profile graph of the data in a `cgPairedDifferenceData` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceData'
profileGraph(data, ...)
```

Arguments

`data`	A `cgPairedDifferenceData` object.
`...`	Additional arguments, both *optional*. Two are currently valid:

`logscale` A `logical` value, indicating whether or not the point graph should be plotted on the logarithmic scale. If `logscale` is not specified, its value is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `logscale` argument.

`ticklabels` A list of two components:

- `mod` Can be either of these two values,
  - `"replace"` Before graphing the data, remove any automatically generated tickmarks for the y-axis, and create the tickmarks specified in the `marks` component below.
  - `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones.
- `marks` A vector of tickmarks to be placed on the y-axis. Any numeric representations will be coerced to character.

Details

The profile graph for paired difference data is the simplest of profiles as each experimental unit has exactly two points connected by a straight line. Labels for the experimental units are added for identification.

Individual points are `jitter`ed, and open circles are used to alleviate potential overlap and the danger of representing multiple points as a single point.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

The heading for the graph is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `analysisname` argument.

The label for the y-axis is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `endptname` and `endptunits` arguments.

The number of decimal places printed in the ticks on the y-axis is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `digits` argument.

Minimum and maximum values from ranges of data are respectively labeled in the bottom and top left corners of graph regions.

Value

`profileGraph.cgPairedDifferenceData` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(anorexia.FT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)
profileGraph(anorexiaFT.data)

# Graph the data on the original scale instead of the log scale.
profileGraph(anorexiaFT.data, logscale=FALSE)
```

`qqGraph` *Quantile-Quantile Graphs*

Description

Create a Quantile-Quantile (Q-Q) Gaussian graph of the residuals of a fitted object from the **cg** package.

Usage

```
qqGraph(fit, line = TRUE, cgtheme = TRUE, device = "single", ...)
```

Arguments

`fit`	A fit object, typically created by the `fit` generic function.
`line`	Add a line to help assess the distribution of the residuals. See specific method written for the `fit` argument.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`device`	Can be one of three values: `"single"` The default, which will put all graphs on the same device page. `"multiple"` Relevant only when multiple fitted models are present in the `fit` object and requested to be plotted. In those cases, a new graphics device is generated to hold each additional plot beyond the first. `"ask"` Relevant only when multiple fitted models are present in the `fit` object and requested to be plotted. In these cases, each plot is portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that the user input confirmation is needed before the graphs are drawn.
`...`	Additional arguments, depending on the specific method written for the object. See the method-specific documentation for additional details.

Value

`qqGraph` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`qqGraph.cgOneFactorFit`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

qqGraph(canine.fit)
```

`qqGraph.cgOneFactorFit` *Quantile-Quantile (QQ) Graphs of a cgOneFactorFit object*

Description

Create a Q-Q Gaussian graph of the residuals of a cgOneFactorFit object

Usage

```
## S4 method for signature 'cgOneFactorFit'
qqGraph(fit, line=NULL, cgtheme = TRUE, device = "single", ...)
```

Arguments

`fit`	A fit object of class `cgOneFactorFit`.
`line`	Add a line through the estimated 25th and 75th percentiles. When set to the default `NULL`, the addition of a line depends on the following: When there is no censored data, the line will be added with the `qqline` algorithm. If any censored data residuals are present, no line is added unless `line=TRUE` is explicitly specified below.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely, `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`device`	Can be one of three values: `"single"` The default, which will put all graphs on the same device page. For example, when resistant & robust and classical least squares are present and `model="both"` (the default), a 2 x 1 paneled graph will be created. `"multiple"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default) or `model="extended"`. In those cases, a new graphics device is generated to hold the resistant & robust version, as a single-paneled graph, and the classical least squares version is on the previous device. If `model="extended"`, then a second new graphics device is generated to hold the unweighted resistant & robust residuals, as another single-paneled graph. `"ask"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default) or `model="extended"`. In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that the user input confirmation is needed before the graphs are drawn.
`...`	Additional arguments. One is currently valid: `model` For `cgOneFactorFit` `fit` objects that have classical least squares `lm()` or resistant & robust `rlm()` fits, the following argument values are possible:

`"both"` Q-Q graphs of the residuals from both the ordinary classical least squares and resistant & robust fits are plotted. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `fit` argument. If the resistant & robust fit is not available, this value is not relevant.

`"olsonly"` Only a Q-Q graph of the residuals from the ordinary classical least squares `olsfit` fit is performed.

`"rronly"` **or** `"rrwtdonly"` Only a Q-Q graph of the weighted residuals from the resistant and robust `rrfit` fit is plotted.

`"rrunwtdonly"` Only a Q-Q graph of the unweighted residuals from the resistant and robust `rrfit` fit is plotted.

For other possible `cgOneFactorFit` fit slots such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the appropriate Q-Q graph will be plotted for these model types.

Details

For censored data residuals, left-censored values are shown as a shallow "V", which is actually just a rotated downward "<" sign. Similarly, right-censored values are shown as a deeper "^", which is a rotated upwared ">" sign.

For the `line` argument, an added line when censored data residuals are present needs to be interpreted very cautiously. If "too many" censored data values are present, the line will appear nonsensical if indeed it can even be estimated with 25th and 75th percentiles in the presence of the the censored data residuals. These percentiles are estimated via the Kaplan-Meier method as proposed by Gentleman and Crowley (1991), with the **survival**::`survfit` function.

The heading for the graph is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `analysisname` argument.

The label for the Y-axis of residuals is derived from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `endptname` argument.

The number of decimal places printed in the ticks on the Y-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `digits` argument.

The minimum and maximum values from the range of the residuals are respectively labeled in the bottom and top left corners of the graph region.

Value

`qqGraph.cgOneFactorFit` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Gentleman, R.C. and Crowley, J. (1991). "Graphical Methods for Censored Data", *Journal of the American Statistical Association*, Volume 86, 678-683.

See Also

`qqline`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

qqGraph(canine.fit)

qqGraph(canine.fit, model="olsonly")
```

---

qqGraph.cgPairedDifferenceFit

Quantile-Quantile (QQ) Graphs of a cgPairedDifferenceFit object

Description

Create a Q-Q Gaussian graph of the residuals of a cgPairedDifferenceFit object

Usage

```
## S4 method for signature 'cgPairedDifferenceFit'
qqGraph(fit, line=TRUE, cgtheme = TRUE, device = "single", ...)
```

Arguments

`fit`	A fit object of class `cgPairedDifferenceFit`.
`line`	Add a line through the estimated 25th and 75th percentiles, when set to the default `TRUE`.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely, `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`device`	Can be one of three values:

`"single"` The default, which will put all graphs on the same device page. For example, when resistant & robust and classical least squares are present and `model="both"` (the default), a 2 x 1 paneled graph will be created.

`"multiple"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default) or `model="extended"`. In those cases, a new graphics device is generated to hold the resistant & robust version, as a single-paneled graph, and the classical least squares version is on the previous device. If `model="extended"`, then a second new graphics device is generated to hold the unweighted resistant & robust residuals, as another single-paneled graph.

`"ask"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default) or `model="extended"`. In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that the user input confirmation is needed before the graphs are drawn.

`...` Additional arguments. One is currently valid:

`model` For `cgPairedDifferenceFit` `fit` objects that have classical least squares `lm()` or resistant & robust `rlm()` fits, the following argument values are possible:

- `"both"` Q-Q graphs of the residuals from both the ordinary classical least squares and resistant & robust fits are plotted. This is the default when both fits are present in the `cgPairedDifferenceFit` object specified in the `fit` argument. If the resistant & robust fit is not available, this value is not relevant.
- `"olsonly"` Only a Q-Q graph of the residuals from the ordinary classical least squares `olsfit` fit is performed.
- `"rronly"` **or** `"rrwtdonly"` Only a Q-Q graph of the weighted residuals from the resistant and robust `rrfit` fit is plotted.
- `"rrunwtdonly"` Only a Q-Q graph of the unweighted residuals from the resistant and robust `rrfit` fit is plotted.

Details

The heading for the graph is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `analysisname` argument.

The label for the Y-axis is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `endptname` argument.

The number of decimal places printed in the ticks on the Y-axis is taken from the `cgPairedDifferenceData` object, which `prepareCGPairedDifferenceData` sets from its `digits` argument.

The minimum and maximum values from the range of the residuals are respectively labeled in the bottom and top left corners of the graph region.

Value

`qqGraph.cgPairedDifferenceFit` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <cg@billpikounis.net> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`qqline`

Examples

```
data(anorexiaFT)

anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)

anorexiaFT.fit <- fit(anorexiaFT.data)

qqGraph(anorexiaFT.fit)

qqGraph(anorexiaFT.fit, model="olsonly")
```

`samplesizeGraph` *Graph Estimated Sample Sizes*

Description

Generic function to graph a table of estimated sample sizes, using a Sample Size table created by the **cg** package.

Usage

```
samplesizeGraph(sstable, Nscale = "log", mmdscale = "log", ...)
```

Arguments

`sstable`	A `samplesizeTable` created by a `samplesizeTable` method from the **cg** package.
`Nscale`	A `character` indicating whether the y-axis, which shows the estimated sample sizes, should be drawn on the log scale (`"log"`) or the original scale (`"original"`).

`mmdscale`	A `character` indicating whether the x-axis, which shows the minimum meaningful differences to be detected, should be drawn on the log scale (`"log"`) or the original scale (`"original"`).
`...`	Additional arguments, depending on the specific method written for the object. See the specific methods for additional details.

Value

The main purpose is the side effect of graphing to the current device. See the specific methods for discussion of any return values.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`samplesizeGraph.cgOneFactorSampleSizeTable`

Examples

```
#### One Factor data

data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.samplesize <- samplesizeTable(canine.fit, direction="increasing",
                                     mmdvec=c(10, 25, 50, 75, 100))

samplesizeGraph(canine.samplesize)

#### Paired Difference data
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)
```

```
## The interest is in increased weight for the anorexia FT
## (family treatment) group of patients
anorexiaFT.samplesize <- samplesizeTable(anorexiaFT.fit, direction="increasing",
                         mmdvec=c(5, 10, 15, 20)) ## print method shows table

samplesizeGraph(anorexiaFT.samplesize)
```

---

samplesizeGraph.cgOneFactorSampleSizeTable

Graph estimated sample sizes from a cgOneFactorSampleSizeTable object

---

Description

Creates a graph to see estimated sample sizes in a cgOneFactorSampleSizeTable object.

Usage

```
## S4 method for signature 'cgOneFactorSampleSizeTable'
samplesizeGraph(sstable, Nscale="log", mmdscale = "log", ...)
```

Arguments

`sstable`	A sample size object of class `cgOneFactorSampleSizeTable`.
`Nscale`	A `character` indicating whether the Y-axis, which shows the estimated samples sizes, should be drawn on the log scale (`"log"`) or the original scale (`"original"`).
`mmdscale`	A `character` indicating whether the X-axis, which shows the minimum meaningful differences to be detected, should be drawn on the log scale (`"log"`) or the original scale (`"original"`).
`...`	Additional arguments:

`cgtheme` When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.

`device` Can be one of three values:

- `"single"` The default, which will put all graphs on the same device page. For example, when resistant & robust and classical least squares sample size estimates are present and `model="both"` (the default), a 2 x 1 paneled graph will be created.
- `"multiple"` Relevant only when resistant & robust and classical least squares sample size estimates are present and `model="both"`. In those cases, a new graphics device is generated to hold the resistant & robust version, as a single-paneled graph, and the classical least squares version is on the previous device.

- `"ask"` Relevant only when resistant & robust and classical least squares samples size estimates are present and `model="both"` (the default). In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that the user input confirmation is needed before the graphs are drawn.

`model` For `cgOneFactorOneSampleSizeTable` objects that have classical least squares `lm()` filled or resistant & robust `rlm()` filled slots, the following argument values are possible:

- `"both"` Sample Size graphs based on both the ordinary classical least squares and resistant & robust slots are populated. This is the default when both slots are present in the `cgOneFactorSampleSizeTable` object specified in the `sstable` argument. If the resistant & robust fit is not available, this value is not relevant.
- `"olsonly"` Only an Sample Size Table Graph based on the ordinary classical least squares table slot is created.
- `"rronly"` Only a Sample Size Table Graph based on the resistant and robust table slot is created.

`mmdticklabels` A list of two components:

- `mod` Can be either of these two values,
  - `"replace"` Before graphing the data, remove any automatically generated tickmarks for the y-axis, and create the tickmarks specified in the `marks` component below.
  - `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones.
- `marks` A vector of tickmarks to be placed on the x-axis. Any numeric representations will be coerced to character.

`Nticklabels` A list of two components:

- `mod` Can be either of these two values,
  - `"replace"` Before graphing the data, remove any automatically generated tickmarks for the x-axis, and create the tickmarks specified in the `marks` component below.
  - `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones.
- `marks` A numeric vector of tickmarks to be placed on the y-axis. Any numeric representations will be coerced to character.

Details

The minimum and maximum sample size values are added inside the plot region in blue, flush against the y-axis in the top and bottom left corners.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

The method essentially portrays in a graph the same information shown by the print method of the `cgOneFactorSampleSizeTable` object.

Value

`samplesizeGraph.cgOneFactorSampleSizeTable` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorSampleSizeTable`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.samplesize <- samplesizeTable(canine.fit, direction="increasing",
                               mmdvec=c(10, 25, 50, 75, 100)) ## print method shows table

samplesizeGraph(canine.samplesize)

samplesizeGraph(canine.samplesize, model="olsonly",
                mmdticklabels=list(mod="add", marks=100))
```

`samplesizeGraph.cgPairedDifferenceSampleSizeTable`
Graph estimated sample sizes from a cgPairedDifferenceSampleSizeTable object

Description

Creates a graph to see estimated sample sizes in a cgPairedDifferenceSampleSizeTable object.

Usage

```
## S4 method for signature 'cgPairedDifferenceSampleSizeTable'
samplesizeGraph(sstable, Nscale, mmdscale, ...)
```

Arguments

`sstable`	A sample size object of class `cgPairedDifferenceSampleSizeTable`.
`Nscale`	A `character` indicating whether the left-hand side Y-axis, which shows the estimated sample sizes in terms of the number of experimental units, should be drawn on the log scale (`"log"`) or the original scale (`"original"`).
`mmdscale`	A `character` indicating whether the X-axis, which shows the minimum meaningful differences to be detected, should be drawn on the log scale (`"log"`) or the original scale (`"original"`).
`...`	Additional arguments. Two are currently valid:

- `nscale` A `character` indicating whether the Y-axis, which shows the estimated sample sizes in terms of the number of `n` experimental units on the left-hand axis, should be drawn on the log scale (`"log"`) or the original scale (`"original"`). This will override the `Nscale` argument, which serves the same purpose.
- `mmdticklabels` A list of two components:
  - `mod` Can be either of these two values,
    - `"replace"` Before graphing the data, remove any automatically generated tickmarks for the x-axis, and create the tickmarks specified in the `marks` component below.
    - `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones.
  - `marks` A vector of tickmarks to be placed on the x-axis. Any numeric representations will be coerced to character.
- `nticklabels` A list of two components:
  - `mod` Can be either of these two values,
    - `"replace"` Before graphing the data, remove any automatically generated tickmarks for the y-axis, and create the tickmarks specified in the `marks` component below.
    - `"add"` Before graphing the data, add tickmarks specified in the `marks` component below, to the automatically generated ones.
  - `marks` A numeric vector of tickmarks to be placed on the y-axis. Any numeric representations will be coerced to character.

Details

The minimum and maximum experimental unit sample size values are added inside the plot region in blue, flush against the y-axis in the top and bottom left corners.

Tick marks are attempted to be chosen wisely. For log-scaled axes in particular, leading digits of 2, 5, and 10 for values are included if possible. Since the algorithm is empirical, the `ticklabels` argument is available for further refinement or complete replacement of tickmarks.

The method essentially portrays in a graph the same information shown by the print method of the `cgPairedDifferenceSampleSizeTable` object.

Value

`samplesizeGraph.cgPairedDifferenceSampleSizeTable` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceSampleSizeTable`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

## The interest is in increased weight for the anorexia FT
## (family treatment) group of patients
anorexiaFT.samplesize <- samplesizeTable(anorexiaFT.fit, direction="increasing",
                         mmdvec=c(5, 10, 15, 20)) ## print method shows table

samplesizeGraph(anorexiaFT.samplesize)

samplesizeGraph(anorexiaFT.samplesize, nticklabels=list(mod="add", marks=3))
```

`samplesizeTable` *Estimate Required Sample Sizes*

Description

Estimate the sample size required to detect a specified difference in a future study. The estimate is based on the variability in a data fit with the **cg** package.

Usage

```
samplesizeTable(fit, direction, mmdvec, power = 0.80,
 alpha = 0.05, nmax = 1000, display = "print", ...)
```

Arguments

`fit`	An object created by calling a `fit` method from the **cg** package. There is currently one class of objects that are currently available: `cgOneFactorFit`, which is prepared by the `fit.cgOneFactorData` method.
`direction`	A `character` indicating whether the sample size should be estimated to detect an `"increase"` or a `"decrease"`. This only effects the sample size estimates if the previous study in `fit` was analyzed on the log scale, in which case the differences in `mmdvec` are relative differences instead of absolute differences. For detecting relative changes, the sample size required to detect a relative increase of 25% is not the same as the sample size to detect a relative decrease of 25%, for example. But for detecting absolute changes, the sample size required to detect an absolute increase of 25 is the same as the sample size to detect an absolute decrease of 25.
`mmdvec`	A `numeric vector` specifying the minimum meaningful differences to be detected in the future study. If the previous study in `fit` was analyzed on the log scale, then the values in `mmdvec` are assumed to be relative percentage increases or decreases, depending on the value of `direction`. If the previous study in `fit` was not analyzed on the log scale, then the values in `mmdvec` are assumed to be absolute increases or decreases, depending on the value of `direction`. Each value in `mmdvec` needs to be positive.
`power`	The power for the future study, set by default to be `0.80`. This is equivalent to $1 - \beta$, where β is the probability of committing a Type II error: accepting the null hypothesis of no difference when differences truly exist.
`alpha`	The significance level or alpha for the future study, set by default as `0.05`.
`nmax`	The maximum number of subjects per group. If more subjects are estimated to be required, than the exact number required is not reported, only the fact that more than the maximum number would be required. This is in place to prevent long and likely unnecessary calculations.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `samplesizeTable` object, which is a formatted text output of the table(s). `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `samplesizeTable` components.
`...`	Additional arguments, depending on the specific method written for the object. See the method-specific documentation for additional details.

Value

A method-specific `SampleSizeTable` object is returned. See the specific methods for discussion of return values.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`samplesizeTable.cgOneFactorFit`

Examples

```
#### One Factor data

data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.samplesize <- samplesizeTable(canine.fit, direction="increasing",
                                     mmdvec=c(10, 25, 50, 75, 100))

samplesizeGraph(canine.samplesize)
```

---

samplesizeTable.cgOneFactorFit

Estimate Sample Sizes based on a cgOneFactorFit object

Description

Estimate the sample size that would be required to detect a specified difference in a one factor study. The estimate is based on the variability that was observed in a previous one factor study. A cgOneFactorSampleSizeTable class object is created.

Usage

```
## S4 method for signature 'cgOneFactorFit'
samplesizeTable(fit, direction, mmdvec,
 power = 0.80, alpha = 0.05, nmax = 1000, display = "print", ...)
```

Arguments

`fit`	A `cgOneFactorFit` object from a previous one factor study.

`direction`	A `character` value indicating whether the sample size should be estimated to detect an `"increase"` or a `"decrease"`. This only effects the sample size estimates if the previous study in `fit` was analyzed on the log scale, in which case the differences in `mmdvec` are relative differences instead of absolute differences. For detecting relative changes, the sample size required to detect a relative increase of 25% is not the same as the sample size to detect a relative decrease of 25%, for example. But for detecting absolute changes, the sample size required to detect an absolute increase of 25 is the same as the sample size to detect an absolute decrease of 25.
`mmdvec`	A `numeric vector` specifying the minimum meaningful differences to be detected in the future study. If the previous study in `fit` was analyzed on the log scale, then the values in `mmdvec` are assumed to be relative percentage increases or decreases, depending on the value of `direction`. If the previous study in `fit` was not analyzed on the log scale, then the values in `mmdvec` are assumed to be absolute increases or decreases, depending on the value of `direction`. Each value in `mmdvec` needs to be positive.
`power`	The power for the future study, set by default to be `0.80`. This is equivalent to $1 - \beta$, where β is the probability of committing a Type II error: accepting the null hypothesis of no difference when differences truly exist.
`alpha`	The significance level or alpha for the future study, set by default as `0.05`.
`nmax`	The maximum number of subjects per group. If more subjects are estimated to be required, then the exact number required is not reported, only the fact that more than the maximum number would be required. This is in place to prevent long and likely unnecessary calculations.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `cgOneFactorSamplesizeTable` object, which is a formatted text output of the table(s). `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default showDefault method, which will just print out the `cgOneFactorSamplesizeTable` components.
`...`	Additional arguments. `ngrps` The number of groups that will be in the future one factor study. The default is `ngrps = 2`. `model` A `character` value indicating which variability estimate in `fit` should be used to estimate the sample size: the robust model (`"rronly"`), the classical model (`"olsonly"`), or both (`"both"`). If an estimate is requested for a model that was not fit, then no sample sizes are returned for that model but an error is not reported (e.g. if only the classical model was fitted but "both" are requested, only the classical model estimates will be returned): `"both"` Sample Size tables based on both the ordinary classical least squares and resistant & robust fits are performed. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `fit` argument. If the resistant & robust fit is not available, this value is not relevant.

`"olsonly"` Only a sample size table based on the ordinary classical least squares `olsfit` fit is calculated.

`"rronly"` Only a sample size table based on the resistant and robust `rrfit` fit is calculated.

Details

This sample size method does not work for fitted models that allowed unequal variances or censored observations.

Sample sizes are estimated for detecting a minimum difference with a global F test. The algorithm is detailed in Fleiss (1986), Appendix A. When there are more than 2 groups, the lower bound of possible noncentrality parameter values is calculated from assuming only two of the `ngrps` number of groups differ by the `mmdvec/2` amount from the "grand mean" while the rest of the groups are equal to the grand mean.

For detecting an absolute difference, the sample size is the smallest group size n for which
`1 - pf (qf (1 - alpha, numdf, dendf), numdf, dendf, ncp)` exceeds `power`,
where `ncp = (n * mmdvec ^ 2) / (2 * sigamest ^ 2)`, and `sigmaest` is the residual mean square error from the model in `fit`. For detecting a relative difference, the calculations are the same except
`ncp = (n * (log (sign * mmdvec / 100 + 1) ) ^ 2) / (2 * sigmaest ^ 2)`, where
`sign = -1` if `direction="decreasing"`, and `sign = 1` if `direction = "increasing"`.

Value

Creates an object of class `cgOneFactorSampleSizeTable`, with the following slots:

`ols.sstable` A matrix with the estimated sample sizes based on the classical model variance estimates, or `NULL`. The matrix has 3 columns and one row for each element of the `mmdvec` `vector`. The first column specifies the minimum meaningful difference (`"mmd"`). The second column gives the number of subjects required for each group (`"n"`), possibly truncated at `nmax`. The third column gives the total number of subjects required (`"N"`), also truncated if `nmax` is truncated.

`rr.sstable` A matrix with the estimated sample sizes based on the robust model variance estimates, or else `NULL` if `model="olsonly"` was specified. See the `ols.sstable` slot description above for the analogous layout of the matrix.

`settings` A list of properties mostly carried as-is from the `data` argument object of class `cgOneFactorData`, with the following additional members:

- `sigmaest` A list with 2 members, `ols`, containing the estimated spread (sigma, standard deviation) from the classical model of `fit`, and `rr`, containing the estimated spread (sigma, standard deviation) from the robust model of `fit`, or `NULL` if the robust model was not fit.
- `planningname` A `character` describing the study or purpose of the sample size analysis. Taken from the `settings$analysisname` of the `fit` object.
- `ngrps` A saved copy of the `ngrps` argument.
- `direction` A saved copy of the `direction` argument.
- `alpha` A saved copy of the `alpha` argument.
- `power` A saved copy of the `power` argument.
- `nmax` A saved copy of the `nmax` argument.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

References

Fleiss, J.L. (1986). *The Design and Analysis of Clinical Experiments*, Appendix A, pages 371 - 376. New York: Wiley.

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

canine.samplesize <- samplesizeTable(canine.fit, direction="increasing",
                                     mmdvec=c(10, 25, 50, 75, 100))

samplesizeTable(canine.fit, direction="decreasing",
                mmdvec=c(25, 50, 75), model="olsonly")
```

`samplesizeTable.cgPairedDifferenceFit`

Estimate Sample Sizes based on a cgPairedDifferenceFit object

Description

Estimate the sample size that would be required to detect a specified difference in a paired difference data study. The estimate is based on the variability that was observed in a previous paired difference data study. A cgPairedDifferenceSampleSizeTable class object is created.

Usage

```
## S4 method for signature 'cgPairedDifferenceFit'
samplesizeTable(fit, direction, mmdvec,
 power = 0.80, alpha = 0.05, nmax = 1000, display = "print", ...)
```

Arguments

`fit`	A `cgPairedDifferenceFit` object from a previous paired difference data study.
`direction`	A `character` value indicating whether the sample size should be estimated to detect an `"increase"` or a `"decrease"`. This only effects the sample size estimates if the previous study in `fit` was analyzed on the log scale, in which case the differences in `mmdvec` are relative differences instead of absolute differences. For detecting relative changes, the sample size required to detect a relative increase of 25% is not the same as the sample size to detect a relative decrease of 25%, for example. But for detecting absolute changes, the sample size required to detect an absolute increase of 25 is the same as the sample size to detect an absolute decrease of 25.
`mmdvec`	A `numeric vector` specifying the minimum meaningful differences to be detected in the future study. If the previous study in `fit` was analyzed on the log scale, then the values in `mmdvec` are assumed to be relative percentage increases or decreases, depending on the value of `direction`. If the previous study in `fit` was not analyzed on the log scale, then the values in `mmdvec` are assumed to be absolute increases or decreases, depending on the value of `direction`. Each value in `mmdvec` needs to be positive.
`power`	The power for the future study, set by default to be `0.80`. This is equivalent to $1 - \beta$, where β is the probability of committing a Type II error: accepting the null hypothesis of no difference when a difference truly exists.
`alpha`	The significance level or alpha for the future study, set by default as `0.05`.
`nmax`	The maximum number of subjects per group. If more subjects are estimated to be required, than the exact number required is not reported, only the fact that more than the maximum number would be required. This is in place to prevent long and likely unnecessary calculations.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `cgPairedDifferenceSamplesizeTable` object, which is a formatted text output of the table(s). `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `cgPairedDifferenceSamplesizeTable` components.
`...`	Additional arguments. Only one is currently valid: `correction` Other than the `NULL` value, only the `"df"` value can be specified. This `"df"` values provides a degrees of freedom correction for using variance estimates based on different degrees of freedom. See details below.

Details

Here, the estimated sample size actually refers to the number of experimental units. Hence the number of observations will always be twice the number of experimental units, due to the paired structure.

This sample size method only works for the classical least squares fitted model, since there is no analogous decomposition of total variance into between-experimental unit and within-experimental unit variance components. Sample sizes are estimated for detecting a minimum difference with the classical least squares t-test / F-test.

The `correction = "df"` argument specifies a method that Fleiss (1986, pages 129-130) attributes to Cochran and Cox (1957) and Fisher. The correction decreases the relative efficiency that is calculated from accounting for correlated paired observations, relative to the unpaired two group design. The adjustment accounts for the different degrees of freedom used for the variance components in the paired design (between-experimental unit, within-experimettal unit, total variability.)

Since the correction reduces the relative efficiency, and the noncentrality parameter is also reduced. The correction is a multiplicative factor bounded below by 0.833 and approaches 1 as the number of experimental units increments from the minimum of `n=2`. The reduction in the noncentrality parameter increases the computed sample size.

Value

Creates an object of class `cgPairedDifferenceSampleSizeTable`, with the following slots:

`ols.sstable` A matrix with the estimated experimental unit sample sizes based on the classical model variance estimates. The matrix has 3 columns and one row for each element of the `mmdvec vector`. The first column specifies the minimum meaningful difference (`"mmd"`). The second column gives the number of experimental units (`"n"`) required, possibly truncated at `nmax`. The third column gives the total number of observations (`"N"`), also possibly truncated at `nmax`. Since this for the paired groups design, `N = n * 2` will always hold.

`settings` A list of properties mostly carried as-is from the `data` argument object of class `cgPairedDifferenceData`, with the following additional members:

`sigmaest` A list with 1 member, `ols`, containing the estimated spread (sigma, standard deviation) variance estimates from the classical model of `fit`. This list component is a vector of length 3, providing the within-experimental unit, between experimental unit, and total variability estimates.

`planningname` A `character` describing the study or purpose of the sample size analysis. Taken from the `settings$analysisname` of the `fit` object.

`direction` A saved copy of the `direction` argument.

`alpha` A saved copy of the `alpha` argument.

`power` A saved copy of the `power` argument.

`nmax` A saved copy of the `nmax` argument.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis, John Oleynick, and Eva Ye

References

Fleiss, J. L. (1986). *The Design and Analysis of Clinical Experiments*, pages 129 - 130. New York: Wiley.

Cochran, W. G. and Cox, G. M. (1957), *Experimental Designs.* Second edition. Wiley.

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1,
                                                  logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

## Recall the interest is in increased weight for the anorexia FT
## (family treatment) group of patients
samplesizeTable(anorexiaFT.fit, direction="increasing",
                mmdvec=c(5, 10, 15, 20))

## and with the adjustment on the noncentrality parameter
samplesizeTable(anorexiaFT.fit, direction="increasing",
                mmdvec=c(5, 10, 15, 20), correction="df")
```

---

show.cgOneFactorComparisonsTable

Show a One Factor Comparisons Table object from the cg package

---

Description

Show a `cgOneFactorComparisonsTable` object, which contains information of comparisons based on a fit in a `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorComparisonsTable'
show(object)
```

Arguments

`object`	A `cgOneFactorComparisonsTable` object, typically created by `comparisonsTable.cgOneFactorFit`.

Details

The object is shown using `showDefault`. See `cgOneFactorComparisonsTable` for details of the object slots.

Value

`show.cgOneFactorComparisonsTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorComparisonsTable`, `showDefault`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)
show(comparisonsTable(canine.fit, display="none"))
```

`show.cgOneFactorDescriptiveTable`

Show a One Factor Descriptive Table object from the cg package

Description

Show a `cgOneFactorDescriptiveTable` object, which contains a table of quantiles and other summary statistics of the data from a `cgOneFactorData` object.

Usage

```
## S4 method for signature 'cgOneFactorDescriptiveTable'
show(object)
```

Arguments

`object`	A `cgOneFactorDescriptiveTable` object, typically created by `descriptiveTable.cgOneFactorData`.

Details

The object is shown using `showDefault`. See `cgOneFactorDescriptiveTable` for details of the object slots.

Value

`show.cgOneFactorDescriptiveTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorDescriptiveTable`, `showDefault`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
show(descriptiveTable(canine.data, display="none"))

## Censored Data
data(gmcsfcens)
gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE)
show(descriptiveTable(gmcsfcens.data, display="none"))
```

`show.cgOneFactorDownweightedTable`

Show a One Factor Downweighted Table object from the cg package

---

Description

Show a `cgOneFactorDownweightedTable` object, which contains information of downweighted observations in a resistant & robust fit from a `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorDownweightedTable'
show(object)
```

Arguments

`object`	A `cgOneFactorDownweightedTable` object, typically created by `downweightedTable.cgOneFactorFit`.

Details

The object is shown using `showDefault`. See `cgOneFactorDownweightedTable` for details of the object slots.

Value

`show.cgOneFactorDownweightedTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorDownweightedTable`, `showDefault`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
```

```
canine.fit <- fit(canine.data)
show(downweightedTable(canine.fit, cutoffwt=0.95, display="none"))
```

`show.cgOneFactorGlobalTest` *Show a Global Test object from the cg package*

Description

Show a `cgOneFactorGlobalTest` object, which contains p-value information from a `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorGlobalTest'
show(object)
```

Arguments

`object`	A `cgOneFactorGlobalTest` object, typically created by `globalTest.cgOneFactorFit`.

Details

The object is shown using `showDefault`. See `cgOneFactorGlobalTest` for details of the object slots.

Value

`show.cgOneFactorGlobalTest` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorGlobalTest`, `showDefault`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)
globalTest(canine.fit, display="show")
show(globalTest(canine.fit, display="none"))
```

`show.cgOneFactorGrpSummaryTable`

Show a One Factor Group Summary Table object from the cg package

---

Description

Show a `cgOneFactorGrpSummaryTable` object, which contains information of group mean and standard error summaries based on a fit in a `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorGrpSummaryTable'
show(object)
```

Arguments

`object`	A `cgOneFactorGrpSummaryTable` object, typically created by `grpSummaryTable.cgOneFactorFit`.

Details

The object is shown using `showDefault`. See `cgOneFactorGrpSummaryTable` for details of the object slots.

Value

`show.cgOneFactorGrpSummaryTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

cgOneFactorGrpSummaryTable, showDefault

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)
show(grpSummaryTable(canine.fit, display="none"))
```

---

show.cgOneFactorSampleSizeTable

Show a One Factor Sample Size Table object from the cg package

Description

Show a `cgOneFactorSampleSizeTable` object, which contains a table of sample size estimates based on a `cgOneFactorFit` object.

Usage

```
## S4 method for signature 'cgOneFactorSampleSizeTable'
show(object)
```

Arguments

`object`	A `cgOneFactorSampleSizeTable` object, typically created by `samplesizeTable.cgOneFactorFit`.

Details

The object is shown using `showDefault`. See `cgOneFactorSampleSizeTable` for details of the object slots.

Value

`show.cgOneFactorSampleSizeTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorSampleSizeTable`, `showDefault`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)
show(samplesizeTable(canine.fit, direction="increasing",
                     mmdvec=c(25, 50, 75, 100), display="none"))
```

`show.cgPairedDifferenceComparisonsTable`

Show a Paired Difference Data Comparisons Table object from the cg package

Description

Show a `cgPairedDifferenceComparisonsTable` object, which contains information of comparisons based on a fit in a `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceComparisonsTable'
show(object)
```

Arguments

`object`	A `cgPairedDifferenceComparisonsTable` object, typically created by `comparisonsTable.cgPairedDifferenceFit`.

Details

The object is shown using `showDefault`. See `cgPairedDifferenceComparisonsTable` for details of the object slots.

Value

`show.cgPairedDifferenceComparisonsTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceComparisonsTable`, `showDefault`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)
show(comparisonsTable(anorexiaFT.fit, display="none"))
```

`show.cgPairedDifferenceCorrelationTable`

Show a Paired Difference Correlation Table object from the cg package

Description

Show a `cgPairedDifferenceCorrelationTable` object, which contains a table of correlations of the data from a `cgPairedDifferenceData` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceCorrelationTable'
show(object)
```

Arguments

`object`	A `cgPairedDifferenceCorrelationTable` object, typically created by `correlationTable.cgPairedDifferenceData`.

Details

The object is shown using `showDefault`. See `cgPairedDifferenceCorrelationTable` for details of the object slots.

Value

`show.cgPairedDifferenceCorrelationTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceCorrelationTable`, `showDefault`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)

## Next two calls are equivalent
correlationTable(anorexiaFT.data, display="show")

show(correlationTable(anorexiaFT.data, display="none"))
```

`show.cgPairedDifferenceDescriptiveTable`
Show a Paired Difference Descriptive Table object from the cg package

---

Description

Show a `cgPairedDifferenceDescriptiveTable` object, which contains a table of quantiles and other summary statistics of the data from a `cgPairedDifferenceData` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceDescriptiveTable'
show(object)
```

Arguments

`object`	A `cgPairedDifferenceDescriptiveTable` object, typically created by `descriptiveTable.cgPairedDifferenceData`.

Details

The object is shown using `showDefault`. See `cgPairedDifferenceDescriptiveTable` for details of the object slots.

Value

`show.cgPairedDifferenceDescriptiveTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceDescriptiveTable`, `showDefault`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)

## Next two calls are equivalent
descriptiveTable(anorexiaFT.data, display="show")

show(descriptiveTable(anorexiaFT.data, display="none"))
```

`show.cgPairedDifferenceDownweightedTable` *Show a Paired Difference Downweighted Table object from the cg package*

Description

Show a `cgPairedDifferenceDownweightedTable` object, which contains information of downweighted observations in a resistant & robust fit from a `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceDownweightedTable'
show(object)
```

Arguments

`object`	A `cgPairedDifferenceDownweightedTable` object, typically created by `downweightedTable.cgPairedDifferenceFit`.

Details

The object is shown using `showDefault`. See `cgPairedDifferenceDownweightedTable` for details of the object slots.

Value

`show.cgPairedDifferenceDownweightedTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceDownweightedTable`, `showDefault`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1,
                                                  logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)
show(downweightedTable(anorexiaFT.fit, cutoffwt=0.25, display="none"))
```

`show.cgPairedDifferenceSampleSizeTable`

Show a Paired Difference Sample Size Table object from the cg package

Description

Show a `cgPairedDifferenceSampleSizeTable` object, which contains a table of sample size estimates based on a `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceSampleSizeTable'
show(object)
```

Arguments

`object`	A `cgPairedDifferenceSampleSizeTable` object, typically created by `samplesizeTable.cgPairedDifferenceFit`.

Details

The object is shown using `showDefault`. See `cgPairedDifferenceSampleSizeTable` for details of the object slots.

Value

`show.cgPairedDifferenceSampleSizeTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceSampleSizeTable`, `showDefault`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

## Recall the interest is in increased weight for the anorexia FT
## (family treatment) group of patients
show(samplesizeTable(anorexiaFT.fit, direction="increasing",
                     mmdvec=c(5, 10, 15, 20), display="none"))
```

`show.cgPairedDifferenceVarianceTable`

Show a Paired Difference Variance Table object from the cg package

---

Description

Show a `cgPairedDifferenceVarianceTable` object, which contains a table of variances from a `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceVarianceTable'
show(object)
```

Arguments

`object`	A `cgPairedDifferenceVarianceTable` object, typically created by `varianceTable.cgPairedDifferenceFit`.

Details

The object is shown using `showDefault`. See `cgPairedDifferenceVarianceTable` for details of the object slots.

Value

`show.cgPairedDifferenceVarianceTable` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceVarianceTable`, `showDefault`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1,
                                                 logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

## Next two calls are equivalent
varianceTable(anorexiaFT.fit, display="show")

show(varianceTable(anorexiaFT.fit, display="none"))
```

`showObj` *Show raw form of an object from the cg package*

Description

Show the raw form of an object from the cg package.

Usage

```
showObj(object)
```

Arguments

`object`	An object created by the **cg** package.

Details

The object raw form is shown using `showDefault`. The name `showObj` is designed for use when the conventional `show` name is an alias for `print` in the cg package.

Value

A method-specific `fit` object is returned. See the specific methods for discussion of return values.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorFit`, `cgPairedDifferenceFit`, `showDefault`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

showObj(canine.fit)
```

`showObj.cgOneFactorFit` *Show a Fitted Model object from the cg package*

Description

Show the raw form of a `cgOneFactorFit` object, which contains model fit information.

Usage

```
## S4 method for signature 'cgOneFactorFit'
showObj(object)
```

Arguments

`object`	A `cgOneFactorFit` object.

Details

The object is shown using `showDefault`. See `cgOneFactorFit` for details of the object slots.

The name `showObj` is designed for use for cases like this when the coventional `show` name is an alias for `print`.

Value

`showObj.cgOneFactorFit` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorFit`, `showDefault`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

showObj(canine.fit)

show(canine.fit) ## alias for print method on the object
```

`showObj.cgPairedDifferenceFit`

Show a Fitted Model object from the cg package

Description

Show the raw form of a `cgPairedDifferenceFit` object, which contains model fit information.

Usage

```
## S4 method for signature 'cgPairedDifferenceFit'
showObj(object)
```

Arguments

`object`	A `cgPairedDifferenceFit` object.

Details

The object is shown using `showDefault`. See `cgPairedDifferenceFit` for details of the object slots.

The name `showObj` is designed for use for cases like this when the coventional `show` name is an alias for `print`.

Value

`showObj.cgPairedDifferenceFit` returns `invisible`. The main purpose is the side effect of printing the whole object to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorFit`, `cgPairedDifferenceFit`, `showDefault`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)
anorexiaFT.fit <- fit(data=anorexiaFT.data, type="rr")

showObj(anorexiaFT.fit)

show(anorexiaFT.fit) ## alias for print method on the object
```

`summary.cgOneFactorFit`

Summary display of a One Factor Model Fit object with some format options

---

Description

Summary printing of a `cgOneFactorFit` object, which contains fitted model information.

Usage

```
## S4 method for signature 'cgOneFactorFit'
summary(object, title = NULL, endptname = NULL, ...)
```

Arguments

`object`	An `cgOneFactorFit` object.
`title`	The title printed out with the summary of the fitted model(s). If `NULL`, it is set to be "Fitted Model Summaries of" the `analysisname` value in the `settings` slot of the `cgOneFactorFit` object.
`endptname`	The endpoint name, printed out with the fitted model information. If `NULL`, it is set to the `endptname` value in the `settings` slot of the `cgOneFactorFit` object.
`...`	Additional arguments. Only one is currently valid: `model` For `cgOneFactorFit` objects that have output derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible: `"both"` Both the ordinary classical least squares and resistant & robust model fit summaries are printed. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `object` argument. `"olsonly"` Only the ordinary classical least squares model fit summary is printed. `"rronly"` Only the resistant & robust model fit summary is printed. For other possible `cgOneFactorFit` components such as accelerated failure time or unequal variance model fits, the `model` argument is not relevant, and the single model fit summary is printed for these model types.

Details

The object summary is printed using a mix of `cat` and `print` calls. See `cgOneFactorFit` for details of the `*fit` and other object slots.

This method simply echoes summary methods for individual fit classes, such as `lm` and `rlm`.

Value

`summary.cgOneFactorFit` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgOneFactorFit`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

summary(canine.fit)
```

`summary.cgPairedDifferenceFit` *Summary display of a Paired Difference Model Fit object with some format options*

Description

Summary printing of a `cgPairedDifferenceFit` object, which contains fitted model information.

Usage

```
## S4 method for signature 'cgPairedDifferenceFit'
summary(object, title = NULL, endptname = NULL, ...)
```

Arguments

`object`	An `cgPairedDifferenceFit` object.
`title`	The title printed out with the summary of the fitted model(s). If `NULL`, it is set to be "Fitted Model Summaries of" the `analysisname` value in the `settings` slot of the `cgPairedDifferenceFit` object.
`endptname`	The endpoint name, printed out with the fitted model information. If `NULL`, it is set to the `endptname` value in the `settings` slot of the `cgPairedDifferenceFit` object.
`...`	Additional arguments. Only one is currently valid: `model` For `cgPairedDifferenceFit` objects that have output derived from classical least squares `lm` or resistant & robust `rlm` fits, the following argument values are possible: `"both"` Both the ordinary classical least squares and resistant & robust model fit summaries are printed. This is the default when both fits are present in the `cgPairedDifferenceFit` object specified in the `object` argument. `"olsonly"` Only the ordinary classical least squares model fit summary is printed. `"rronly"` Only the resistant & robust model fit summary is printed.

Details

The object summary is printed using a mix of `cat` and `print` calls. See `cgPairedDifferenceFit` for details of the `*fit` and other object slots.

This method simply echoes summary methods for individual fit classes, such as `lm` and `rlm`.

Value

`summary.cgPairedDifferenceFit` returns `invisible`. The main purpose is the side effect of printing to the current output connection, which is typically the console.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`cgPairedDifferenceFit`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

summary(anorexiaFT.fit)
```

`varianceGraph` *Variance Graphs*

Description

Create an equal variance assessment graph of the residuals of a fitted object from the **cg** package

Usage

```
varianceGraph(fit, trend = NULL, cgtheme = TRUE, device = "single", ...)
```

Arguments

`fit`	A fit object, typically created by the `fit` generic function.
`trend`	Add a trend line to help assess the trend of the residuals. See specific method written for the `fit` argument.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`device`	Can be one of three values: `"single"` The default, which will put all graphs on the same device page. `"multiple"` Relevant only when multiple fitted models are present in the `fit` object and requested to be plotted. In those cases, a new graphics device is generated to hold each additional plot beyond the first. `"ask"` Relevant only when multiple fitted models are present in the `fit` object and requested to be plotted. In these cases, each plot is portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that the user input confirmation is needed before the graphs are drawn.
`...`	Additional arguments, depending on the specific method written for the object. See the method-specific documentation for additional details.

Details

The graphs plot the square root of the absolute value of the residuals against the fitted value. The notion of using the squared root of the absolute residuals is attributed to John Tukey.

Value

`varianceGraph` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`varianceGraph.cgOneFactorFit`

Examples

```
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)

varianceGraph(canine.fit)

varianceGraph(canine.fit, model="olsonly")
```

`varianceGraph.cgOneFactorFit`

Equal Variance Graphs assessment of cgOneFactorFit object

Description

Graph residuals of a cgOneFactorFit object to assess equal variance assumption

Usage

```
## S4 method for signature 'cgOneFactorFit'
varianceGraph(fit, trend = NULL, cgtheme = TRUE,
 device = "single", ...)
```

Arguments

`fit`	A fit object of class `cgOneFactorFit`.
`trend`	Add a trend line. When set to the default `NULL`, the addition of a trend line depends on the following: When there is no censored data, a trend will be added with the `lowess` smoother. If any censored data residuals are present, then at least 7 groups are needed in the fit for a trend line to be added with the **VGAM**::`vgam` smoothing spline. Otherwise, group means are estimated and connected. The values of `FALSE` and `TRUE` override this. See Details section below.
`cgtheme`	When set to the default `TRUE`, ensures a trellis device is active with limited color scheme. Namely `background`, `strip.shingle`, and `strip.background` are each set to `"white"`.
`device`	Can be one of three values: `"single"` The default, which will put all graphs on the same device page. For example, when resistant & robust and classical least squares are present and `model="both"` (the default), a 2 x 1 paneled graph will be created. `"multiple"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default) or `model="extended"`. In those cases, a new graphics device is generated to hold the resistant & robust version, as a single-paneled graph, and the classical least squares version is on the previous device. If `model="extended"`, then a second new graphics device is generated to hold the unweighted resistant & robust residuals, as another single-paneled graph. `"ask"` Relevant only when resistant & robust and classical least squares are present and `model="both"` (the default) or `model="extended"`. In that case, each are portrayed as a single-paneled graph, with the `ask=TRUE` argument specified in `par` so that the user input confirmation is needed before the graphs are drawn.
`...`	Additional arguments. Two are currently valid: `model` For `cgOneFactorFit` `fit` objects that have classical least squares `lm()` or resistant & robust `rlm()` fits, the following argument values are possible: `"both"` Graphs of the residuals from both the ordinary classical least squares and resistant & robust fits are plotted. This is the default when both fits are present in the `cgOneFactorFit` object specified in the `fit` argument. If the resistant & robust fit is not available, this value is not relevant. `"olsonly"` Only a graph of the residuals from the ordinary classical least squares `olsfit` fit is performed. `"rronly"` **or** `"rrwtdonly"` Only a graph of the weighted residuals from the resistant and robust `rrfit` fit is plotted. `"rrunwtdonly"` Only a graph of the unweighted residuals from the resistant and robust `rrfit` fit is plotted. For other possible `cgOneFactorFit` fit slots such as accelerated failure time or unequal variance models, the `model` argument is not relevant, and the appropriate graph will be plotted for these model types.

Details

The graph plots the square root of the absolute value of the residuals against the fitted value. The square root spacing on the y-axis has tick marks in the fitted scale. The notion of using the squared root of the absolute residuals is attributed to John Tukey.

The values are automatically jittered to minimize overlapping points. For censored data, left-censored values are shown as a shallow "V", which is actually just a rotated downward "<" sign. Similarly, right-censored values are shown as a deeper "^", which is a rotated upward ">" sign.

For the `trend` argument, an added trend line when censored data residuals are present needs to be interpreted cautiously. When there are 7 or more groups, a cubic smoothing spline based on **VGAM**::`vgam` is fit; otherwise, the group means of the residuals are estimated with an accelerated failure time model and then just connected. If "too many" censored data values are present, the line may be withheld and warnings will be issued, or if forced with `line=TRUE`, for example, may appear nonsensical.

The heading for the graph is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `analysisname` argument.

The label for the Y-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `endptname` argument.

The number of decimal places printed in the ticks on the Y-axis is taken from the `cgOneFactorData` object, which `prepareCGOneFactorData` sets from its `digits` argument.

The minimum and maximum values from the range of the absolute valued residuals are respectively labeled in the bottom and top left corners of the graph region.

If group labels along the x-axis seem to overlap in the standard horizontal form, they will be rotated 45 degrees.

Value

`varianceGraph.cgOneFactorFit` returns an invisible `NULL`. The main purpose is the side effect of graphing to the current device.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
## From running examples of cgOneFactorData objects
data(canine)
canine.data <- prepareCGOneFactorData(canine, format="groupcolumns",
                                      analysisname="Canine",
                                      endptname="Prostate Volume",
                                      endptunits=expression(plain(cm)^3),
                                      digits=1, logscale=TRUE, refgrp="CC")
canine.fit <- fit(canine.data)
```

```
varianceGraph(canine.fit)

varianceGraph(canine.fit, model="olsonly")


gmcsfcens.data <- prepareCGOneFactorData(gmcsfcens, format="groupcolumns",
                                         analysisname="cytokine",
                                         endptname="GM-CSF (pg/ml)",
                                         logscale=TRUE,
                                         digits=1)

gmcsfcens.fit <- fit(gmcsfcens.data, type="aft")

varianceGraph(gmcsfcens.fit, trend=TRUE)

varianceGraph(gmcsfcens.fit) ## will yield a warning message why no line
                             ## is graphed

varianceGraph(gmcsfcens.fit, trend=FALSE)
```

`varianceTable` *Compute Variance Components*

Description

Create a table of variances from a **cg** fit object.

Usage

```
varianceTable(fit, display = "print", ...)
```

Arguments

`fit`	A fit object created and prepared (see `prepare`) using the **cg** package.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `varianceTable` object, which is a formatted text output of the table. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the `varianceTable` components.
`...`	Additional arguments. Currently none are valid.

Value

A method-specific `varianceTable` object is returned. See the specific methods for discussion of return values.

Note

Contact `<cg@billpikounis.net>` for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

See Also

`varianceTable.cgPairedDifferenceFit`

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                  analysisname="Anorexia FT",
                                                  endptname="Weight",
                                                  endptunits="lbs",
                                                  expunitname="Patient",
                                                  digits=1,
                                                  logscale=TRUE)
anorexiaFT.fit <- fit(data=anorexiaFT.data)
varianceTable(anorexiaFT.fit)
```

`varianceTable.cgPairedDifferenceFit`

Compute Variances from a cgPairedDifferenceFit object

Description

Create a table of variance component estimates of the data in a `cgPairedDifferenceFit` object.

Usage

```
## S4 method for signature 'cgPairedDifferenceFit'
varianceTable(fit, display = "print", ...)
```

Arguments

`fit`	A `cgPairedDifferenceFit` object, typically created by `fit.cgPairedDifferenceData`.
`display`	One of three valid values: `"print"` The default value; It calls a `print` method for the created `cgPairedDifferenceVarianceTable` object, which is a formatted text output of the table. `"none"` Supresses any printing. Useful, for example, when just assignment of the resulting object is desired. `"show"` Calls the default `showDefault` method, which will just print out the cgPairedDifferenceVatianceTable object components.
`...`	Additional arguments. Currently none are valid.

Details

The returned table contains variance component estimates for the classical least squares fit. There is no analogous decomposition of variance component estimates calculated for the resistant & robust fit.

Value

Creates an object of class `cgPairedDifferenceVarianceTable`, with the following slots:

`contents` The table of variance component estimates. There are two, the "within experimental unit" variance and the "between experimental unit" variance. See below for the data frame structure of the table. The label portion "experimental unit" will be replaced by the `expunitname` component of the `settings` slot of the `cgPairedDifferenceFit` `fit` object, if previously specified.

`efficiency` A table of efficiency estimates, derived from the variance component estimates. The goal is to quantify the reduced number of experimental units needed since a paired difference design was employed, instead of a an unpaired design. See below for the data frame structure of the table.

`settings` A list of settings carried from the `cgPairedDifferenceFit` `fit` object. These are used for the `print.cgPairedDifferenceVarianceTable` method, invoked for example when `display="print"`.

The data frame structure of the variance components table from the classical least squares fit is provided in the `contents` slot. The data frame consists of `row.names` based on the `expunitname` component of the `settings` slot in the `cgPairedDifferenceFit` `fit` object. The first row is for the "within" component, and the second is for the "between" component. The "total" variance is in the third row of the table, the sum of the between and within variance components. The first column of the table is the variance components estimates, and the third column is the square root of the variance components, labeled `Spread(StdDev)`. In the second column is the `Percent` calculation of the two variance components relative to the total sum variance.

The data frame structure of the efficiency table from the classical least squares fit is provided in the `efficiency` slot. There are four rows and one column. All values are derived from the variance components estimates in the `contents` slot described above. The first row of `Relative Efficiency` comes from dividing the total variance by the between experimental unit variance component. The second row expresses the estimated gain in sensitivity by using a paired difference design and analysis over using a unpaired design and analysis. This is equal to the within experimental unit variance component divided by the total variance, and is expressed here as `Percent Reduction`. The third row is the number of experimental units based on the input data set paired structure. The last row contains the estimated number of unpaired design experimental units that would have been needed for the same sensitivity. The label portion "experimental unit" in these last two row names will be replaced by the `expunitname` component of the `settings` slot of the `cgPairedDifferenceFit` `fit` object if previously specified.

Note

Contact <`cg@billpikounis.net`> for bug reports, questions, concerns, and comments.

Author(s)

Bill Pikounis [aut, cre, cph], John Oleynick [aut], Eva Ye [ctb]

Examples

```
data(anorexiaFT)
anorexiaFT.data <- prepareCGPairedDifferenceData(anorexiaFT, format="groupcolumns",
                                                 analysisname="Anorexia FT",
                                                 endptname="Weight",
                                                 endptunits="lbs",
                                                 expunitname="Patient",
                                                 digits=1, logscale=TRUE)
anorexiaFT.fit <- fit(anorexiaFT.data)

varianceTable(anorexiaFT.fit)
```

Index