Package 'HH'

July 2, 2014

Type Package

Title Statistical Analysis and Data Display: Heiberger and Holland

Version 3.0-4

Date 2014-02-22

Author Richard M. Heiberger

Maintainer Richard M. Heiberger <rmh@temple.edu>

Depends R (>= 3.0.2), lattice, stats, grid, latticeExtra, multcomp

Imports reshape2, leaps, vcd, colorspace, RColorBrewer

Suggests Rcmdr, methods, abind, mvtnorm

Description Support software for Statistical Analysis and Data Display (First Edition, Springer, ISBN 0-387-40270-5) and (Second Edition, Springer, ISBN x-xxx-xxxxx-x). This contemporary presentation of statistical methods features extensive use of graphical displays for exploring data and for displaying the analysis. The second edition includes redesigned graphics and additional chapters. The authors emphasize how to construct and interpret graphs, discuss principles of graphical design, and show how accompanying traditional tabular results are used to confirm the visual impressions derived directly from the graphs. Many of the graphical formats are novel and appear here for the first time in print. All chapters have exercises. All functions introduced in the book are in the package. R code for all examples, both graphs and tables, in the book is included in the scripts directory of the package.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-11 15:19:39

R topics documented:

HH-package *Support software for Statistical Analysis and Data Display by Richard M. Heiberger and Burt Holland*

Description

Support software for *Statistical Analysis and Data Display* (Springer, ISBN 0-387-40270-5). This contemporary presentation of statistical methods features extensive use of graphical displays for exploring data and for displaying the analysis. The authors demonstrate how to analyze data—showing code, graphics, and accompanying computer listings—for all the methods they cover. They emphasize how to construct and interpret graphs, discuss principles of graphical design, and show how accompanying traditional tabular results are used to confirm the visual impressions derived directly from the graphs. Many of the graphical formats are novel and appear here for the first time in print. All chapters have exercises.

Details

Package:	HH
Type:	Package
Version:	1.4
Date:	2006-08-21
License:	GPL version 2 or newer

data display, scatterplot matrix, MMC mean–mean multiple comparison plots, interaction plots, ANCOVA plots, regression diagnostics, time series, ARIMA models, boxplots

Author(s)

Richard M. Heiberger

Maintainer: Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

ancova, ci.plot, interaction2wt, ladder, case.lm, norm.curve, hov, resid.squares, MMC, xysplom

Examples

```
## interaction2wt()
## multicomp.mmc()  ## S-Plus
## mmc()       ## R
## ancova()
## xysplom()
## plot.case()
## bwplot() ## with position
## tsacfplots()     ## at this writing, only S-Plus
## tsdiagplot()     ## at this writing, only S-Plus
```

---

ae.dotplot *AE (Adverse Events) dotplot of incidence and relative risk*

Description

A two-panel display of the most frequently occurring AEs in the active arm of a clinical study. The first panel displays their incidence by treatment group, with different symbols for each group. The second panel displays the relative risk of an event on the active arm relative to the placebo arm, with 95% confidence intervals for a 2×2 table. By default, the AEs are ordered by relative risk so that events with the largest increases in risk for the active treatment are prominent at the top of the display. See the Details section for information on changing the sort order.

Usage

```
ae.dotplot(ae, ...)

ae.dotplot.long(xr,
                A.name = levels(xr$RAND)[1], B.name = levels(xr$RAND)[2],
                col.AB = c("red","blue"), pch.AB = c(16, 17),
                main.title = paste("Most Frequent On-Therapy Adverse Events",
                                   "Sorted by Relative Risk"),
                main.cex = 1,
                cex.AB.points = NULL, cex.AB.y.scale = 0.6,
              position.left = c(0, 0, 0.7, 1), position.right = c(0.61, 0, 0.98, 1),
                key.y = -0.2, CI.percent=95)

logrelrisk(ae, A.name, B.name, crit.value=1.96)

panel.ae.leftplot(x, y, groups, col.AB, ...)

panel.ae.rightplot(x, y, ..., lwd=6, lower, upper, cex=.7)
```

```
panel.ae.dotplot(x, y, groups, ..., col.AB, pch.AB, lower, upper) ## R only

aeReshapeToLong(aewide)
```

Arguments

ae	For ae.dotplot, either a data.frame containing the Adverse Event data in long format as described by the detail for xr below, or a data.frame containing the Adverse event data in wide format as described by the detail for aewide below. For logrelrisk, a data.frame containing the first 4 columns of xr described below.
...	For ae.dotplot, all the arguments listed in the calling sequence for ae.ddotplot.long and possibly standard panel function arguments. For the other functions, just standard panel function arguments.
xr	RAND: treatment as randomized (factor).\ PREF: adverse event symptom name (factor).\ SN: number of patients in treatment group.\ SAE: number of patients in each group for whom the event PREF was observed.\ PCT: SAE/SN as a percent.\ relrisk: Relative risk defined as PCT for the B treatment divided by PCT for the A treatment.\ logrelrisk: natural logarithm of relrisk.\ ase.logrelrisk: asymptotic standard error of logrelrisk.\ logrelriskCI.lower, logrelriskCI.upper: confidence interval for logrelrisk. \ relriskCI.lower, relriskCI.upper: back transform of the CI for the log relative risk into the relative risk scale.\
aewide	Event: adverse event symptom name (factor).\ N.A, N.B: number of patients in treatment groups A and B.\ AE.A, AE.B: number of patients in treatment groups A and B for whom the event Event was observed.\ PCT.A, PCT.B: AE.A/N.A and AE.B/N.B as a percent.\ Relative.Risk: Relative risk defined as PCT.B divided by PCT.A.\ logrelrisk: natural logarithm of relrisk.\ ase.logrelrisk: asymptotic standard error of logrelrisk.\ logrelriskCI.lower, logrelriskCI.upper: confidence interval for logrelrisk. \ relriskCI.lower, relriskCI.upper: back transform of the CI for the log relative risk into the relative risk scale.\
A.name, B.name	Names of treatment groups (in x$RAND).
col.AB, pch.AB, cex.AB.points	color, plotting character and character expansion for the individual points on the left plot.
cex.AB.y.scale	Character expansion for the left tick labels (the symptom names).
main.title, main.cex	Main title and character expansion for the combined plot in ae.dotplot.
cex	The character expansion for the points in the left and right plots.
position.left, position.right	position of the left and right plots. This argument is use in S-Plus only, not in R. See the discussion of position in print.trellis,

key.y	Position of the key (legend) in the combined plot. This is the y argument of the key. See the discussion of the key argument to xyplot in xyplot.
crit.value	Critical value used to compute confidence intervals on the log relative risk. Defaults to 1.96. User is responsible for specifying both crit.value and CI.percent consistently.
CI.percent	Confidence percent associated with the crit.value Defaults to 95. User is responsible for specifying both crit.value and CI.percent consistently.
x, y, groups, lwd	standard panel function arguments.
lower, upper	xr$logrelriskCI.lower and xr$logrelriskCI.upper inside the panel functions.

Details

The second panel shows relative risk of an event on the active arm (treatment B) relative to the placebo arm (treatment A), with 95% confidence intervals for a 2×2 table. Confidence intervals on the log relative risk are calculated using the asymptotic standard error formula given as Equation 3.18 in Agresti A., Categorical Data Analysis. Wiley: New York, 1990.

By default the ae.dotplot function sorts the events by relative risk. To change the sort order, you must redefine the ordering of the ordered factor PREF. See the examples below.

Value

logrelrisk takes an input data.frame of the form x described in the argument list and returns a data.frame consisting of the input argument with additional columns as described in the argument xr. The result column of symptom names PREF is an ordered factor, with the order specified by the relative risk.

ae.leftplot returns a "trellis" object containing a horizontal dotplot of the percents against each of the symptom names.

ae.rightplot returns a "trellis" object containing a horizontal plot on the log scale of the relative risk confidence intervals against each of the symptom names.

ae.dotplot calls both ae.leftplot and ae.rightplot and combines their plots into a single display with a single set of left axis labels, a main title, and a key. The value returned invisibly is a list of the full left trellis object and the right trellis object with its left labels blanked out. Printing the value will not usually be interesting as the main title and key are not included. It is better to call ae.dotplot directly, perhaps with a change in some of the positioning arguments.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Ohad Amit, Richard M. Heiberger, and Peter W. Lane. (2008) "Graphical Approaches to the Analysis of Safety Data from Clinical Trials". *Pharmaceutical Statistics*, **7**, 1, 20–35.

http://www3.interscience.wiley.com/journal/114129388/abstract

Examples

```
## variable names in the input data.frame aeanonym
## RAND   treatment as randomized
## PREF   adverse event symptom name
## SN     number of patients in treatment group
## SAE    number of patients  in each group for whom the event PREF was observed
##
## Input sort order is PREF/RAND

data(aeanonym)
head(aeanonym)

## Calculate log relative risk and confidence intervals (95% by default).
## logrelrisk sets the sort order for PREF to match the relative risk.
aeanonymr <- logrelrisk(aeanonym) ## sorts by relative risk
head(aeanonymr)

## construct and print plot on current graphics device
ae.dotplot(aeanonymr,
           A.name="TREATMENT A (N=216)",
           B.name="TREATMENT B (N=431)")
## export.eps(h2("stdt/figure/aerelrisk.eps"))
## This looks great on screen and exports badly to eps.
## We recommend drawing this plot directly to the postscript device:
##
## trellis.device(postscript, color=TRUE, horizontal=TRUE,
##                colors=ps.colors.rgb[
##                  c("black", "blue", "red", "green",
##                    "yellow", "cyan","magenta","brown"),],
##                onefile=FALSE, print.it=FALSE,
##                file=h2("stdt/figure/aerelrisk.ps"))
## ae.dotplot(aeanonymr,
##            A.name="TREATMENT A (N=216)",
##            B.name="TREATMENT B (N=431)")
## dev.off()

## To change the sort order, redefine the PREF factor.
## For this example, to plot alphabetically, use the statement
aeanonymr$PREF <- ordered(aeanonymr$PREF, levels=sort(levels(aeanonymr$PREF)))
ae.dotplot(aeanonymr,
           A.name="TREATMENT A (N=216)",
           B.name="TREATMENT B (N=431)",
           main.title="change the main title to reflect the new sort order")

## Not run:
## to restore the order back to the default, use
relrisk <- aeanonymr[seq(1, nrow(aeanonymr), 2), "relrisk"]
PREF <- unique(aeanonymr$PREF)
aeanonymr$PREF <- ordered(aeanonymr$PREF, levels=PREF[order(relrisk)])
ae.dotplot(aeanonymr,
           A.name="TREATMENT A (N=216)",
           B.name="TREATMENT B (N=431)",
```

```
          main.title="back to the original sort order")

## smaller artifical example with the wide format
aewide <- data.frame(Event=letters[1:6],
                     N.A=c(50,50,50,50,50,50),
                     N.B=c(90,90,90,90,90,90),
                     AE.A=2*(1:6),
                     AE.B=1:6)
aewtol <- aeReshapeToLong(aewide)
xr <- logrelrisk(aewtol)
ae.dotplot(xr)

## End(Not run)
```

AEdotplot *AE (Adverse Events) dotplot of incidence and relative risk*

Description

A three-panel display of the most frequently occurring AEs in the active arm of a clinical study. The first panel displays their incidence by treatment group, with different symbols for each group. The second panel displays the relative risk of an event on the active arm relative to the placebo arm, with 95% confidence intervals for a 2×2 table. By default, the AEs are ordered by relative risk so that events with the largest increases in risk for the active treatment are prominent at the top of the display. By setting the argument `sortbyRelativeRisk=FALSE`, the AEs retain the order specified by the levels of the factor. The third panel displays the numerical values of number of patients for each treatment, number of adverse events for each treatment, and relative risk. The third panel can be suppressed by the `print` method.

Usage

```
  AEdotplot(xr, ...)

  ## S3 method for class 'formula'
AEdotplot(xr, groups=NULL, data=NULL,
          sortbyRelativeRisk=TRUE,
          ...,
          sub=list(deparse(this.call[1:4],
                           width.cutoff=500), cex=.7))
```

Arguments

`xr`	For the formula method, a formula of the form `AE ~ nAE/nTRT | OrgSys`, where the condition variable is optional. For the formula method only, the variable names are not restricted. See `AEdotplot.data.frame` for the support methods.
`groups`	Variable containing the treatment levels.

`data` `data.frame` containing at least four variables: containing the AE name, the treatment level, the number of observed AE in that treatment level, the number of patients in that treatment group. It may also contain a fifth variable containing a condition variable used to split the `data.frame` into partitions. It may be used to partition the plot, for example by organ system or by gender. The treatment factor must have exactly two levels. Each AE name must appear exactly once for each level of the treatment.

`sortbyRelativeRisk` logical. If `TRUE`, then make the Adverse Events an ordered factor ordering by relative risk. If `FALSE`, then make the Adverse Events an ordered factor retaining the order of the input levels.

`sub` Subtitle for the plot. The default value is the command that generates the plot.

`...` Any of the arguments listed in the calling sequence for the methods documented in `AEdotplot.data.frame`.

Details

The first panel is an ordinary dotplot of the percent of AE observed for each treatment by AE.

The second panel shows relative risk of an event on the Treatment B arm (usually the active compound) relative to the Treatment A arm (usually the placebo), with 95% confidence intervals for a 2×2 table. Confidence intervals on the log relative risk are calculated using the asymptotic standard error formula given as Equation 3.18 in Agresti A., *Categorical Data Analysis.* Wiley: New York, 1990.

By default the `AEdotplot` function sorts the events by relative risk. To retain the sort order implied by the `levels` of the AE factor, specify the argument `sortbyRelativeRisk=FALSE`. To control the sort order, make the AE factor in the input dataset an `ordered` factor and specify the levels in the order you want.

The third panel shows the numerical values of the number and percent of observed events on each arm and the relative risk. The display of third panel can be suppressed by specifying the `panel.widths` argument. See the discussion of the `panel.widths` in `AEdotplot.data.frame`.

Value

The primary interest is in the display of the plot.

The function returns an `AEdotplot` object which is a list of three `trellis` objects, one for the the Percent plot, one for the Relative Risk plot, and one for the Text plot containing the table of input values. The object has attributes

1. `main` and `sub` hold the main and subtitles. Each must be a list containing the text in the first component.
2. `ae.key` is a key as described in `xyplot`.
3. `n.events` is a vector containing the number of events in each subpanel.
4. `panel.widths` is a vector of relative widths of the three components of the graph. The numbers must sum to one. Zero values are permitted. The first width includes the left axis and the Percent plot. The second is the Relative Risk plot, and the third is the plot of the table values.
5. `AEtable` is a table containing the data plotted on its row.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Ohad Amit, Richard M. Heiberger, and Peter W. Lane. (2008) "Graphical Approaches to the Analysis of Safety Data from Clinical Trials". *Pharmaceutical Statistics*, **7**, 1, 20–35.

http://www3.interscience.wiley.com/journal/114129388/abstract

See Also

AEdotplot.data.frame

Examples

```
## formula method.  See ?AEdotplot.data.frame for other methods.
data(AEdata)
head(AEdata)

AEdotplot(AE ~ nAE/nTRT, groups = TRT, data = AEdata)
AEdotplot(AE ~ nAE/nTRT | OrgSys, groups = TRT, data = AEdata)

## Not run:


AEdotplot(AE ~ nAE/nTRT | OrgSys, groups = TRT,
          data = AEdata[c(AEdata$OrgSys %in% c("GI","Resp")),])

## test sortbyRelativeRisk=FALSE
ABCD.12345 <- AEdata[1:12,]
head(ABCD.12345)
AEdotplot(AE ~ nAE/nTRT | OrgSys, groups=TRT, data=ABCD.12345)
AEdotplot(AE ~ nAE/nTRT | OrgSys, groups=TRT, data=ABCD.12345, sort=FALSE)

## suppress third panel
tmp <- AEdotplot(AE ~ nAE/nTRT, groups = TRT, data = AEdata)
print(tmp, AEtable=FALSE)

## End(Not run)
```

AEdotplot.data.frame *AE (Adverse Events) dotplot of incidence and relative risk, support functions*

Description

Support functions for the AEdotplot.

Usage

```
  ## S3 method for class 'data.frame'
AEdotplot(xr, ...,
          conditionVariable=NULL,
          conditionName=deparse(substitute(xr)),
          useCondition=!is.null(conditionVariable),
          sub=list(conditionName, cex=.7))

  ## S3 method for class 'AElogrelrisk'
AEdotplot(xr,
          A.name=paste(levels(xr$RAND)[1], " (n=", xr$SN[1], ")", sep=""),
          B.name=paste(levels(xr$RAND)[2], " (n=", xr$SN[2], ")", sep=""),
          col.AB=c("red","blue"), pch.AB=c(16,17),
           main=if (sortbyRelativeRisk)
           list("Most Frequent On-Therapy Adverse Events Sorted by Relative Risk",
                  cex=1)
           else
              list("Most Frequent On-Therapy Adverse Events", cex=1),
          cex.AB.points=NULL, cex.AB.y.scale=.6, cex.x.scale=.6,
          panel.widths=c(.55, .22, .23),
          key.y=-.2, CI.percent=95,
          conditionName=deparse(substitute(xr)),
          sortbyRelativeRisk=TRUE,
          ...,
          sub=list(conditionName, cex=.7))

  ## S3 method for class 'AEtable'
AEdotplot(xr, ..., useCondition=TRUE,
                             sub="sub for AEsecond")

  ## S3 method for class 'AEdotplot'
print(x, ...,
      main=attr(x, "main"),
      sub=attr(x,"sub"),
      ae.key=attr(x, "ae.key"),
      panel.widths=attr(x,"panel.widths"),
      AEtable=TRUE)

  ## S3 method for class 'AEdotplot'
c(..., panel.widths=attr(aedp[[1]], "panel.widths"))

  AElogrelrisk(ae,
               A.name=levels(ae$RAND)[1],
               B.name=levels(ae$RAND)[2],
               crit.value=1.96,
               sortbyRelativeRisk=TRUE, ...)

  AEmatchSortorder(AEstandard,
```

```
    AEsecond,
    AEsecond.AEtable=attr(AEsecond, "AEtable"),
    levels.order=
       lapply(attr(AEstandard,"AEtable"),
              function(AEsubtable) levels(AEsubtable$PREF)),
  main.second=list(paste("Most Frequent On-Therapy Adverse Events",
                         "Sorted to Match First Table"),
                   cex=1))

## S3 method for class 'AEdotplot'
update(object, ...)
```

Arguments

ae	For AElogrelrisk, a data.frame containing at least the first 4 columns of xr.
xr	For the formula method documented in AEdotplot, a formula of the form AE ~ nAE/nTRT | OrgSys, where the condition variable is optional. For the formula method only, the variable names are not restricted. For the other methods, xr is a data.frame containing the Adverse Event data in long format. It must have variables named RAND: treatment as randomized (factor with exactly two levels). PREF: adverse event symptom name (factor). SN: number of patients in treatment group. SAE: number of patients in each group for whom the event PREF was observed. If the xr object is a AElogrelrisk object, then it must also have variables PCT: SAE/SN as a percent. relrisk: Relative risk defined as PCT for the B treatment divided by PCT for the A treatment. logrelrisk: natural logarithm of relrisk. ase.logrelrisk: asymptotic standard error of logrelrisk. logrelriskCI.lower, logrelriskCI.upper: confidence interval for logrelrisk. relriskCI.lower, relriskCI.upper: back transform of the CI for the log relative risk into the relative risk scale.
sortbyRelativeRisk	logical. If TRUE, then make the Adverse Events an ordered factor ordering by relative risk. If FALSE, then make the Adverse Events an ordered factor retaining the order of the input levels.
conditionVariable	Vector of same length as number of rows in xr, it may be one of the columns in xr in which case its full name in the form xr$varname must be used. It will be used to split the data.frame into partitions. It may be used to partition the plot, for example by organ system or by gender.
conditionName	Character. Name to be used in left.strip.
useCondition	logical. If FALSE, then a non-NULL ConditionVariable won't be used.
x	object to be printed.
panel.widths	Vector of three non-negative numerics that sum to 1. These are the widths of each of the three panels in the output plot. The left panel contains the AE names

as y-tick labels and the Percent plot. The middle panel contains the Relative Risk plot. The right panel contains a table of the numerical values of number of patients for each treatment, number of adverse events for each treatment, and relative risk. Setting the third value to 0 suppresses the table of numerical values from the display.

`AEtable` logical. For the `print.AEdotplot` function. If `TRUE` (the default), display all three panels. If `FALSE`, then display only the Percent and Relative Risk plots.

`main, sub` Main title and subtitle for the combined plot in `AEdotplot`.

`main.second` Main title for second plot whose sort order has been changed to match the first plot.

`A.name, B.name` Names of treatment groups (in `x$RAND`).

`col.AB, pch.AB, cex.AB.points`
color, plotting character and character expansion for the individual points on the left plot.

`cex.AB.y.scale` Character expansion for the left tick labels (the Adverse Effects names).

`cex.x.scale` Character expansion for the x-axis tick labels.

`key.y` Position of the key (legend) in the combined plot. This is the `y` argument of the `key`. See the discussion of the `key` argument to `xyplot` in `xyplot`.

`ae.key` is a key as described in `xyplot`.

`AEstandard, AEsecond, AEsecond.AEtable, levels.order`
Arguments that force the Adverse Events in the panels of `AEsecond` to have the same sort order `levels.order` of `PREF` as the panels of `AEstandard`. `AEstandard` and `AEsecond` are two `"AEdotplot"` objects with the same set of panels and the same Adverse Events in corresponding panels. `AEsecond.AEtable` is the `AEtable` object from `AEsecond`. `levels.order` is the new order for `AEsecond`; normally the same order as in `AEprimary`.

`crit.value` Critical value used to compute confidence intervals on the log relative risk. Defaults to 1.96. User is responsible for specifying both `crit.value` and `CI.percent` consistently.

`CI.percent` Confidence percent associated with the `crit.value` Defaults to 95. User is responsible for specifying both `crit.value` and `CI.percent` consistently.

`...` For `AEdotplot` and `AEdotplot.data.frame`, all the arguments listed in the calling sequence for `AEddotplot.AErelrisk`.. For `c.AEdotplot`, one or more `"AEdotplot"` objects. For `print.AEdotplot`, the `...` arguments are ignored.

`object` An `AEdotplot` object. The `update` method updates the components of each of the constituent `trellis` objects. It does not update the `"main"` and `"sub"` attributes (nor any other attribute) of the `AEdotplot` object.

Details

The first panel is an ordinary dotplot of the percent of AE observed for each treatment by AE.

The second panel shows relative risk of an event on the Treatment B arm (usually the active compound) relative to the Treatment A arm (usually the placebo), with 95% confidence intervals for a 2×2 table. Confidence intervals on the log relative risk are calculated using the asymptotic standard

error formula given as Equation 3.18 in Agresti A., *Categorical Data Analysis.* Wiley: New York, 1990.

By default the `AEdotplot` function sorts the events by relative risk. To retain the sort order implied by the `levels` of the AE factor, specify the argument `sortbyRelativeRisk=FALSE`. To control the sort order, make the AE factor in the input dataset an `ordered` factor and specify the levels in the order you want.

The third panel shows the numerical values of the number and percent of observed events on each arm and the relative risk. The display of third panel can be suppressed by specifying the `panel.widths` argument.

Value

The primary interest is in the display of the plot.

The function returns an `AEdotplot` object which is a list of three `trellis` objects, one for the the Percent plot, one for the Relative Risk plot, and one for the Text plot containing the table of input values. The object has attributes

1. `main` and `sub` hold the main and subtitles. Each must be a list containing the text in the first component.
2. `ae.key` is a key as described in `xyplot`.
3. `n.events` is a vector containing the number of events in each subpanel.
4. `panel.widths` is a vector of relative widths of the three components of the graph. The numbers must sum to one. Zero values are permitted. The first width includes the left axis and the Percent plot. The second is the Relative Risk plot, and the third is the plot of the table values.
5. `AEtable` is a table containing the data plotted on its row.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Ohad Amit, Richard M. Heiberger, and Peter W. Lane. (2008) "Graphical Approaches to the Analysis of Safety Data from Clinical Trials". *Pharmaceutical Statistics*, **7**, 1, 20–35.

http://www3.interscience.wiley.com/journal/114129388/abstract

See Also

`AEdotplot`

Examples

```
## Not run:
## variable names in the input data.frame aeanonym
## RAND   treatment as randomized
## PREF   adverse event symptom name
## SN     number of patients in treatment group
## SAE    number of patients  in each group for whom the event PREF was observed
```

```
## OrgSys Organ System
##
## Input sort order is PREF/RAND

data(aeanonym)
head(aeanonym)

## variable names are hard-wired in the program
## names(aeanonym) <- c("RAND", "PREF", "SAE", "SN", "OrgSys")


## Calculate log relative risk and confidence intervals (95
## AElogrelrisk sets the sort order for PREF to match the relative risk.
aeanonymr <- AElogrelrisk(aeanonym) ## PREF sorted by relative risk
head(aeanonymr)
class(aeanonymr$PREF)
levels(aeanonymr$PREF)

AEdotplot(aeanonym)
\dontrun{
AEdotplot(aeanonym, sort=FALSE)
}
AEdotplot(aeanonym, conditionVariable=aeanonym$OrgSys)

aefake <- rbind(cbind(aeanonym, group="ABC"), cbind(aeanonym, group="DEF"))
aefake$SAE[67:132] <- sample(aefake$SAE[67:132])
aefake$OrgSys.group <- with(aefake, interaction(OrgSys, group))

## fake 2
KEEP <- aefake$OrgSys %in% c("GI","Resp")

AEfakeGR <- AEdotplot(aefake[KEEP,], conditionVariable=aefake$OrgSys.group[KEEP],
            sub=list("ABC and DEF have different sort orders for PREF", cex=.7))
AEfakeGR ## ABC and DEF have different sort orders for PREF

AEfakeGR1 <- AEdotplot(aefake[KEEP & (1:132) <= 66,],
                       conditionVariable=aefake$OrgSys.group[KEEP & (1:132) <= 66])
AEfakeGR2 <- AEdotplot(aefake[KEEP & (1:132) >= 67,],
                       conditionVariable=aefake$OrgSys.group[KEEP & (1:132) >= 67])

AEfakeGR1
AEfakeGR2

AEfakeMatched <- AEmatchSortorder(AEfakeGR1, AEfakeGR2)
update(do.call(c, AEfakeMatched),
       main="ABC sorted by Relative Risk; DEF matches ABC order")

## End(Not run)
## Please see  ?AEdotplot  for examples using the formula method
##
## Many more examples are in demo("AEdotplotManyExamples")
```

ancova *Compute and plot oneway analysis of covariance*

Description

Compute and plot oneway analysis of covariance. The result object is an ancova object which consists of an ordinary aov object with an additional `trellis` attribute. The `trellis` attribute is a `trellis` object consisting of a series of plots of `y ~ x`. The left set of panels is conditioned on the levels of the factor groups. The right panel is a superpose of all the groups.

Usage

```
ancova(formula, data.in = NULL, ...,
       x, groups, transpose = FALSE,
       display.plot.command = FALSE,
       superpose.level.name = "superpose",
       ignore.groups = FALSE, ignore.groups.name = "ignore.groups",
       blocks, blocks.pch = letters[seq(levels(blocks))],
       layout, between, main)

panel.ancova(x, y, subscripts, groups,
 transpose = FALSE, ...,
 coef, contrasts, classes,
 ignore.groups, blocks, blocks.pch, blocks.cex)

## The following are ancova methods for generic functions.
## S3 method for class 'ancova'
anova(object, ...)

## S3 method for class 'ancova'
predict(object, ...)

## S3 method for class 'ancova'
print(x, ...) ## prints the anova(x) and the trellis attribute

## S3 method for class 'ancova'
model.frame(formula, ...)

## S3 method for class 'ancova'
summary(object, ...)

## S3 method for class 'ancova'
plot(x, y, ...) ## standard lm plot.  y is always ignored.

## S3 method for class 'ancova'
coef(object, ...)
```

Arguments

formula	A formula specifying the model.
data.in	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.
...	Arguments to be passed to aov, such as subset or na.action.
x	Covariate in ancova, needed for plotting when the formula does not include x. "aov" object in print.ancova, to match the argument of the print generic function. Variable to plotted in "panel.ancova".
groups	Factor. Needed for plotting when the formula does not include groups after the conditioning bar "|".
transpose	S-Plus: The axes in each panel of the plot are transposed. The analysis is identical, just the axes displaying it have been interchanged. R: no effect.
display.plot.command	The default setting is usually what the user wants. The alternate value TRUE prints on the console the command that draws the graph. This is strictly for debugging the ancova command.
superpose.level.name	Name used in strip label for superposed panel.
ignore.groups	When TRUE, an additional panel showing all groups together with a common regression line is displayed.
ignore.groups.name	Name used in strip label for ignore.groups panel.
blocks	Additional factor used to label points in the panels.
blocks.pch	Alternate set of labels used when a blocks factor is specified.
blocks.cex	Alternate set of cex used when a blocks factor is specified.
layout	The layout of multiple panels. The default is a single row. See details.
between	Space between the panels for the individual group levels and the superpose panel including all groups.
main	Character with a main header title to be done on the top of each page.
y,subscripts	In "panel.ancova", see panel.xyplot.
object	An "aov"object. The functions using this argument are methods for the similarly named generic functions.
coef, contrasts, classes	Internal variables used to communicate between ancova and panel.ancova. They keep track of the constant or different slopes and intercepts in each panel of the plot.

Details

The ancova function does two things. It passes its arguments directly to the aov function and returns the entire aov object. It also rearranges the data and formula in its argument and passes that to the xyplot function. The trellis attribute is a trellis object consisting of a series of plots of y ~ x. The left set of panels is conditioned on the levels of the factor groups. The right panel is a superpose of all the groups.

Value

The result object is an ancova object which consists of an ordinary aov object with an additional trellis attribute. The default print method is to print both the anova of the object and the trellis attribute.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

ancova-class aov xyplot

Examples

```
data(hotdog)

## y ~ x                     ## constant line across all groups
ancova(Sodium ~ Calories,     data=hotdog, groups=Type)

## y ~ a                     ## different horizontal line in each group
ancova(Sodium ~            Type, data=hotdog, x=Calories)

## This is the usual usage
## y ~ x + a  or  y ~ a + x  ## constant slope, different intercepts
ancova(Sodium ~ Calories + Type, data=hotdog)
ancova(Sodium ~ Type + Calories, data=hotdog)

## y ~ x * a  or  y ~ a * x  ## different slopes, and different intercepts
ancova(Sodium ~ Calories * Type, data=hotdog)
ancova(Sodium ~ Type * Calories, data=hotdog)

## y ~ a * x ## save the object and print the trellis graph
hotdog.ancova <- ancova(Sodium ~ Type * Calories, data=hotdog)
attr(hotdog.ancova, "trellis")
```

```
## label points in the panels by the value of the block factor
data(apple)
ancova(yield ~ treat + pre, data=apple, blocks=block)

## Please see
##      demo("ancova")
## for a composite graph illustrating the four models listed above.
```

---

ancova-class *Class "ancova" Analysis of Covariance*

Description

Analysis of Covariance. The class is an extension of "aov" and "lm". It is identical to the "aov" for a single factor and a single covariate plus an attribute which contains a `"trellis"` object. Four different models are included in the class. See ancova for the examples.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class `"aov"`, directly. Class `"lm"`, by class "aov", distance 2. Class `"mlm"`, by class "aov", distance 2, with explicit test and coerce. Class `"oldClass"`, by class "aov", distance 3. Class `"oldClass"`, by class "aov", distance 4, with explicit test and coerce.

Methods

No methods defined with class "ancova" in the signature. S3-type methods are `"anova.ancova"`, `"coef.ancova"`, `"coefficients.ancova"`, `"model.frame.ancova"`, `"plot.ancova"`, `"predict.ancova"`, `"print.ancova"`, `"summary.ancova"`. `"plot.ancova(x)"` plots a standard lm plot of `x`. `"print.ancova(x)"` prints the `anova(x)` and the trellis attribute. The remaining methods use `NextMethod`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

ancova

anovaMean *ANOVA table from the group sample sizes, means, and standard deviations.*

Description

Oneway ANOVA table from the summary information consisting of group sample sizes, means, and standard deviations. The full dataset is not needed.

Usage

```
anovaMean(object, n, ybar, s, ..., ylabel = "ylabel")
```

Arguments

object	level names
n	sample size for each level
ybar	sample mean for each level
s	sample standard deviation for each level
...	other arguments (not used)
ylabel	name of response variable

Value

Analysis of variance table, identical to the ANOVA table that would have been produced by `anova.lm` if the original data, rather than the summary data, had been available.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`anova.lm`, `plot.mmc.multicomp`

Examples

```
## pulmonary data used in Hsu and Peruggia paper defining the mean-mean plot
## See ?plot.mmc.multicomp for details on the dataset.

data(pulmonary)

anovaMean(pulmonary$smoker,
          pulmonary$n,
          pulmonary$FVC,
          pulmonary$s,
          ylabel="pulmonary")
```

aovSufficient *Analysis of variance from sufficient statistics for groups.*

Description

Analysis of variance from sufficient statistics for groups. For each group, we need the factor level, the response mean, the within-group standard deviation, and the sample size. The correct ANOVA table is produced. The residuals are fake. The generic vcov and summary.lm don't work for the variance of the regression coefficients in this case. Use vcovSufficient.

Usage

```
aovSufficient(formula, data = NULL,
              projections = FALSE, qr = TRUE, contrasts = NULL,
              weights = data$n, sd = data$s,
              ...)

vcovSufficient(object, ...)
```

Arguments

formula, data, projections, qr, contrasts, ...	See aov.
weights	See lm.
sd	vector of within-group standard deviations.
object	"aov" object constructed by aovSufficient. It also works with regular aov objects.

Value

For aovSufficient, an object of class c("aov", "lm"). For vcovSufficient, a function that returns the covariance matrix of the regression coefficients.

Note

The residuals are fake. They are all identical and equal to the MLE standard error (sqrt(SumSq.res/df.tot)). They give the right ANOVA table. They may cause confusion or warnings in other programs. The standard errors and t-tests of the coefficients are not calculated by summary.lm. Using the aov object from aovSufficient in glht requires the vcov. and df arguments.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

MMC and

aov.

Examples

```
## This example is from Hsu and Peruggia

## This is the R version
## See ?mmc.mean for S-Plus

if.R(s={},
r={

data(pulmonary)
pulmonary
pulmonary.aov <- aovSufficient(FVC ~ smoker,
                               data=pulmonary)
summary(pulmonary.aov)

## Not run:
pulmonary.mmc <- mmc(pulmonary.aov,
                     linfct=mcp(smoker="Tukey"),
                     df=pulmonary.aov$df.residual,
                     vcov.=vcovSufficient)
old.omd <- par(omd=c(.0,.95,0,1))
plot(pulmonary.mmc, ry=c(2.5, 3.4), x.offset=.05)

## tiebreaker plot, with contrasts ordered to match MMC plot,
## with all contrasts forced positive and with names also reversed,
## and with matched x-scale.
plotMatchMMC(pulmonary.mmc$mca)

## orthogonal contrasts
pulm.lmat <- cbind("npnl-mh"=c( 1, 1, 1, 1,-2,-2), ## not.much vs lots
                   "n-pnl"  =c( 3,-1,-1,-1, 0, 0), ## none vs light
                   "p-nl"   =c( 0, 2,-1,-1, 0, 0), ## {} arbitrary 2 df
                   "n-l"    =c( 0, 0, 1,-1, 0, 0), ## {} for 3 types of light
                   "m-h"    =c( 0, 0, 0, 0, 1,-1)) ## moderate vs heavy
dimnames(pulm.lmat)[[1]] <- row.names(pulmonary)
pulm.lmat

pulmonary.mmc <- mmc(pulmonary.aov,
                     linfct=mcp(smoker="Tukey"),
                     df=pulmonary.aov$df.residual,
                     vcov.=vcovSufficient,
                     focus.lmat=pulm.lmat)

plot(pulmonary.mmc, ry=c(2.5, 3.4), x.offset=.05)
plotMatchMMC(pulmonary.mmc$lmat, col.signif='blue')

## pairwise and orthogonal contrasts on the same plot
```

```
plot(pulmonary.mmc, ry=c(2.5, 3.4), x.offset=.05,
     print.mca=TRUE, print.lmat=TRUE)
par(old.omd)

## End(Not run)
})
```

`arima.diag.hh` *Repair design error in S-Plus arima.diag*

Description

Repair design error in S-Plus `arima.diag`.

Usage

```
arima.diag.hh(z, acf.resid = TRUE,
              lag.max = round(max(gof.lag + n.parms + 1, 10 * log10(n))),
              gof.lag = 15, resid = FALSE,
              std.resid = TRUE, plot = TRUE, type = "h", ...,
              x=eval(parse(text = series.name)))
```

Arguments

`z,acf.resid,lag.max,gof.lag,resid,std.resid,plot,type,...`	This function is a no-op in R. The arguments are not used.
`x`	The time series. This must be specified when arima.diag is called from inside another function.

Details

Repairs design flaw in S-Plus `arima.diag`. The location of the time series is hardwired one level up, so it can't be found when `arima.diag` is not one level down from the top.

This function is a no-op in R.

Value

This function is a no-op in R. It returns `NA`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`tsdiagplot` in both systems and

`arima.diag` in S-Plus.

`arma.loop` *Loop through a series of ARIMA models and display coordinated tables and diagnoastic graphs.*

Description

Loop through a series of ARIMA models and display coordinated tables and diagnostic graphs. The complete example from the Heiberger and Teles article, also included in the Heiberger and Holland book, is illustrated.

Usage

```
arma.loop(x,
          model,             ## S-Plus
          order, seasonal,   ## R
          series=deparse(substitute(x)), ...)

diag.arma.loop(z,
               x=stop("The time series x is needed in S-Plus when p=q=0."),
               lag.max = 36, gof.lag = lag.max)

rearrange.diag.arma.loop(z)
```

Arguments

`x`	Time series vector. In S-Plus, `x` must be an `"rts"`.
`model`	A valid S-Plus `model` for `arima.mle`.
`order, seasonal`	A valid R `order` and `seasonal` for `arima`.
`series`	Character string describing the time series.
`...`	Additional arguments for `arima.mle` or `arima`.
`z`	For `diag.arma.loop`, an "arma.loop" object. For `rearrange.diag.arma.loop`, an "diag.arma.loop" object.
`lag.max`	Maximum lag for the acf and pacf plots.
`gof.lag`	Maximum lag for the gof plots.

Details

S-Plus and R have different functions, with different input argument names and different components in their value.

Value

`arma.loop`: `"arma.loop"` object which is a matrix of lists, each containing an arima model.

`diag.arma.loop`: `"diag.arma.loop"` object which is a matrix of lists, each containing the standard diagnostics for one arima model.

`rearrange.diag.arma.loop`: List of matrices, each containing all the values for a specific diagnostic measure collected from the set of arima models.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

References

"Displays for Direct Comparison of ARIMA Models" The American Statistician, May 2002, Vol. 56, No. 2, pp. 131-138. Richard M. Heiberger, Temple University, and Paulo Teles, Faculdade de Economia do Porto.

Richard M. Heiberger and Burt Holland (2004), Statistical Analysis and Data Display, Springer, ISBN 0-387-40270-5

See Also

`tsdiagplot`

Examples

```
## see tsdiagplot for the example
```

---

as.likert *Support functions for diverging stacked barcharts for Likert, semantic differential, and rating scale data.*

Description

Constructs class="likert" objects to be used by the plot.likert methods.

Usage

```
is.likert(x)

as.likert(x, ...)
## Default S3 method:
as.likert(x, ...)
## S3 method for class 'data.frame'
as.likert(x, ...)
## S3 method for class 'formula'
```

```
as.likert(x, ...)  ## doesn't work yet
## S3 method for class 'ftable'
as.likert(x, ...)
## S3 method for class 'table'
as.likert(x, ...)
## S3 method for class 'matrix'
as.likert(x,
                ReferenceZero=NULL,
                ...,
                rowlabel=NULL, collabel=NULL,
                xlimEqualLeftRight=FALSE,
                xTickLabelsPositive=TRUE,
                padding=FALSE,
                reverse.left=TRUE)
## S3 method for class 'listOfNamedMatrices'
as.likert(x, ...)
## S3 method for class 'array'
as.likert(x, ...)


## S3 method for class 'likert'
rev(x)

is.likertCapable(x, ...)
```

Arguments

x	For the as.likert methods, a numeric object stored as a vector, matrix, two-dimensional table, two-dimensional ftable, two-dimensional structable (as defined in the vcd package), or list of named matrices. For is.likert and is.likertCapable, any object. This is the only required argument.
rowlabel, collabel	names(dimnames(x)), where x is the argument to the as.likert functions. These will become the xlab and ylab of the likert plot.
...	other arguments. They will be ignored by the as.likert method.
ReferenceZero	Please see discussion of this argument in likert.
xlimEqualLeftRight	Logical. The default is FALSE. If TRUE, then the left and right x limits are set to negative and positive of the larger of the absolute value of the original x limits.
xTickLabelsPositive	Logical. The default is TRUE. If TRUE, then the tick labels on the negative side are displayed as positive values.
padding, reverse.left	padding is FALSE for likert and TRUE for likertMosaic. reverse.left is TRUE for likert and FALSE for likertMosaic. likert is based on barchart and requires that the sequencing of negative values be reversed. likertMosiac is based on mosaic and needs padding on left and right to fill the rectangle implied by the convex hull of the plot.

Details

Please see likert for information on the plot for which as.likert prepares the data.

Value

For the as.likert methods, a likert object, which is a matrix with additional attributes that are needed to make the barchart method used by the plot.likert methods work with the data. Columns for respondents who disagree have negated values. The column of the original data for respondents who neither agree nor disagree is split into two columns, each containing halved values—one positive and one negative. Negative columns come first in the sequence of "No Opinion"(negative)–"Strongly Disagree", followed by "No Opinion"(positive)–"Strongly Agree". There are four attributes: "even.col" indicating whether there were originally an even number of columns, "n.levels" the original number of levels, "levels" the original levels in the original order, "positive.order" The sequence in which to display the rows in order to make the right hand sides progress with high values on top.

is.likert returns a TRUE or FALSE value.

is.likertCapable returns a TRUE or FALSE value if the argument can used as an argument to one of the plot.likert methods.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Richard M. Heiberger, Naomi B. Robbins (2014)., "Design of Diverging Stacked Bar Charts for Likert Scales and Other Applications", Journal of Statistical Software, 57(5), 1–32, http://www.jstatsoft.org/v57/i05/.

Naomi Robbins <naomi@nbr-graphs.com>, "Visualizing Data: Challenges to Presentation of Quality Graphics—and Solutions", Amstat News, September 2011, 28–30.

Naomi B. Robbins and Richard M. Heiberger (2011). Plotting Likert and Other Rating Scales. In JSM Proceedings, Section on Survey Research Methods. Alexandria, VA: American Statistical Association.

Luo, Amy and Tim Keyes (2005). "Second Set of Results in from the Career Track Member Survey," Amstat News. Arlington, VA: American Statistical Association.

See Also

likert

Examples

```
## Please see ?likert to see these functions used in context.

tmp2 <- array(1:12, dim=c(3,4), dimnames=list(B=LETTERS[3:5], C=letters[6:9]))
as.likert(tmp2)  ## even number of levels.

is.likert(tmp2)
```

```
is.likert(as.likert(tmp2))
```

`as.matrix.listOfNamedMatrices` *Convert a list of matrices to a single matrix*

Description

Convert a list of matrices to a single matrix. This function is used to improve legibility of the printed object. The `as.matrix.listOfNamedMatrices` display is easier to read when the rownames are very long, as in the example illustrated here. Because the default print of the matrix repeats the rownames several times, with only a few columns of the data shown in each repetition, the actual matrix structure of the data values is obscured.

Usage

```
## S3 method for class 'listOfNamedMatrices'
as.matrix(x, abbreviate = TRUE, minlength = 4, ...)
is.listOfNamedMatrices(x,  xName=deparse(substitute(x)))
## S3 method for class 'listOfNamedMatrices'
as.data.frame(x, ...)
as.listOfNamedMatrices(x,  xName=deparse(substitute(x)), ...)
## S3 method for class 'listOfNamedMatrices'
x[...]
## S3 method for class 'array'
as.listOfNamedMatrices(x, xName=deparse(substitute(x)), ...)
## S3 method for class 'list'
as.listOfNamedMatrices(x, xName=deparse(substitute(x)), ...)
## S3 method for class 'MatrixList'
as.listOfNamedMatrices(x, xName=deparse(substitute(x)), ...)
## S3 method for class 'listOfNamedMatrices'
print(x, ...)

as.MatrixList(x)
## S3 method for class 'array'
as.MatrixList(x)
## S3 method for class 'MatrixList'
print(x, ...)

as.likertDataFrame(x, xName=deparse(substitute(x)))
## S3 method for class 'listOfNamedMatrices'
as.likertDataFrame(x, xName=deparse(substitute(x)))
## S3 method for class 'array'
as.likertDataFrame(x, xName=deparse(substitute(x)))
```

Arguments

x	Named list of matrices. All matrices in the list should have the same number of columns and the same column names. The list item names will normally be long. The row names will normally be long. The number of rows and their names will normally differ across the matrices. Each named item in the list may be a vector, matrix, array, data.frame, two-dimensional table, two-dimensional ftable, or two-dimensional structable For the `as.MatrixList` methods, an array.
...	Other arguments. Not used.
abbreviate	Logical. If `TRUE`, then use the `abbreviate` function on the item names and row names.
minlength	the minimum length of the abbreviations.
xName	Name of the argument in its original environment.

Value

The result of `as.listOfNamedMatrices` is a list with `class=c("listOfNamedMatrices", "list")`.

The result of `as.matrix.listOfNamedMatrices` is an `rbind` of the individual matrices in the argument list `x`. The rownames of the result matrix are constructed by pasting the abbreviation of the list item names with the abbreviation of the individual matrix rownames. The original names are retained as the `"Subtables.Rows"` attribute.

The result of `is.listOfNamedMatrices` is logical value.

`print.listOfNamedMatrices` prints `as.matrix.listOfNamedMatrices` of its argument and returns the original argument.

`as.data.frame.listOfNamedMatrices(x, ...)` is an unfortunate kluge. The result is the original `x` that has NOT been transformed to a `data.frame`. A `warning` message is generated that states that the conversion has not taken place. This kluge is needed to use `"listOfNamedMatrices"` objects with the `Commander` package because `Rcmdr` follows its calls to the R `data` function with an attempt, futile in this case, to force the resulting object to be a `data.frame`.

The `as.MatrixList` methods construct a list of matrices from an array. Each matrix has the first two dimensions of the array. The result list is itself an array defined by all but the first two dimensions of the argument array.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`likert`

Examples

```
data(ProfChal)

tmp <- data.matrix(ProfChal[,1:5])
rownames(tmp) <- ProfChal$Question
```

```
ProfChal.list <- split.data.frame(tmp, ProfChal$Subtable)

## Original list of matrices is difficult to read because
## it is displayed on too many lines.
ProfChal.list[2:3]

## Single matrix with long list item names and long row names
## of argument list retained as an attribute.
as.listOfNamedMatrices(ProfChal.list[2:3], minlength=6)

tmp3 <- array(1:24, dim=c(2,3,4), dimnames=list(A=letters[1:2], B=LETTERS[3:5], C=letters[6:9]))
tmp3
as.MatrixList(tmp3)

## Not run:
sapply(as.MatrixList(tmp3), as.likert, simplify=FALSE)  ## odd number of levels.

data(NZScienceTeaching)
likert(Question ~ ., NZScienceTeaching)
likert(Question ~ . | Subtable, data=NZScienceTeaching)
likert(Question ~ . | Subtable, data=NZScienceTeaching,
       layout=c(1,2), scales=list(y=list(relation="free")))

## End(Not run)
```

`as.multicomp` *Support functions in R for MMC (mean–mean multiple comparisons) plots.*

Description

MMC plots: In R, functions used to interface the `glht` in R to the MMC functions designed with S-Plus `multicomp` notation. These are all internal functions that the user doesn't see.

Usage

```
## S3 method for class 'mmc.multicomp'
print(x, ..., width.cutoff=options()$width-5)

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

## print.multicomp.hh(x, digits = 4, ..., height=T) ## S-Plus only

## S3 method for class 'multicomp.hh'
print(x, ...) ## R only
```

```
as.multicomp(x, ...)

## S3 method for class 'glht'
as.multicomp(x,       ## glht object
           focus,   ## currently required
           ylabel=deparse(terms(x$model)[[2]]),
           means=model.tables(x$model, type="means",
                              cterm=focus)$tables[[focus]],
           height,
           lmat=t(x$linfct),
           lmat.rows=lmatRows(x, focus),
           lmat.scale.abs2=TRUE,
           estimate.sign=1,
           order.contrasts=TRUE,
           contrasts.none=FALSE,
           level=0.95,
           calpha=NULL,
           method=x$type,
           df,
           vcov.,
           ...
           )

as.glht(x, ...)

## S3 method for class 'multicomp'
as.glht(x, ...)
```

Arguments

x	"glht" object for as.multicomp. A "mmc.multicomp" object for print.mmc.multicomp. A "multicomp" object for as.glht and print.multicomp.
...	other arguments.
focus	name of focus factor.
ylabel	response variable name on the graph.
means	means of the response variable on the focus factor.
lmat, lmat.rows	mmc
lmat.scale.abs2	logical, almost always TRUE. If it is not TRUE, then the contrasts will not be properly placed on the MMC plot.
estimate.sign	numeric. 1: force all contrasts to be positive by reversing negative contrasts. -1: force all contrasts to be negative by reversing positive contrasts. Leave contrasts as they are constructed by glht.
order.contrasts, height	logical. If TRUE, order contrasts by height (see mmc).

contrasts.none	logical. This is an internal detail. The "contrasts" for the group means are not real contrasts in the sense they don't compare anything. mmc.glht sets this argument to TRUE for the none component.
level	Confidence level. Defaults to 0.95.
calpha	R only. User-specified critical point. See confint.glht.
df, vcov.	R only. Arguments forwarded through glht to modelparm.
method	R only. See type in confint.glht.
width.cutoff	See deparse.

Details

The mmc.multicomp print method displays the confidence intervals and heights on the MMC plot for each component of the mmc.multicomp object.

print.multicomp displays the confidence intervals and heights for a single component.

Value

as.multicomp is a generic function to change its argument to a "multicomp" object.

as.multicomp.glht changes an "glht" object to a "multicomp" object. If the model component of the argument "x" is an "aov" object then the standard error is taken from the anova(x$model) table, otherwise from the summary(x). With a large number of levels for the focus factor, the summary(x) function is exceedingly slow (80 minutes for 30 levels on 1.5GHz Windows XP). For the same example, the anova(x$model) takes a fraction of a second.

Note

The multiple comparisons calculations in R and S-Plus use completely different libraries. MMC plots in R are based on

glht. MMC plots in S-Plus are based on

multicomp. The MMC plot is the same in both systems. The details of gettting the plot differ.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Heiberger, R.~M. and Holland, B. (2006). "Mean–mean multiple comparison displays for families of linear contrasts." *Journal of Computational and Graphical Statistics*, 15:937–955.

See Also

mmc,

glht.

`axis.i2wt` *specialized axis function for interaction2wt.*

Description

Labels the bottom axis with the x-factor name for each column. Labels the right axis with the response variable name in all rows.

Usage

```
axis.i2wt(side, scales, ...)
```

Arguments

`side, scales, ...`
See `axis.default`.

Author(s)

Richard M. Heiberger, with asssistance from Deepayan Sarkar.

See Also

`interaction2wt`

`ci.plot` *Plot confidence and prediction intervals for simple linear regression*

Description

The data, the least squares line, the confidence interval lines, and the prediction interval lines for a simple linear regression (`lm(y ~ x)`) are displayed. Tick marks are placed at the location of xbar, the x-value of the narrowest interval.

Usage

```
ci.plot(lm.object, ...)

## S3 method for class 'lm'
ci.plot(lm.object,
        xlim=range(data[, x.name]),
        newdata,
        conf.level=.95,
        data=model.frame(lm.object),
        newfit,
        ylim,
        pch=16,
        main.cex=1,
        main=list(paste(100*conf.level,
          "% confidence and prediction intervals for ",
          substitute(lm.object), sep=""), cex=main.cex), ...
        )
```

Arguments

lm.object	Linear model for one y and one x variable.
xlim	xlim for plot. Default is based on data from which lm.object was constructed.
newdata	data.frame containing data for which predictions are wanted. The variable name of the column must be identical to the name of the predictor variable in the model object. Defaults to a data.frame containing a vector spanning the range of observed data. User-specified values are appended to the default vector.
conf.level	Confidence level for intervals, defaults to .95
data	data extracted from the lm.object
newfit	Constructed data.frame containing the predictions,confidence interval, and prediction interval for the newdata.
ylim	ylim for plot. Default is based on the constructed prediction interval.
pch	Plotting character for observed points.
main.cex	Font size for main title.
main	Main title for plot
...	Additional arguments to be passed to panel function.

Value

"trellis" object containing the plot.

Note

The predict.lm functions in S-Plus and R differ. The S-Plus function can produce both confidence and prediction intervals with a single call. The R function produces only one of them in a single call. Therefore the default calculation of newfit within the function depends on the system.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

lm, predict.lm

Examples

```
tmp <- data.frame(x=rnorm(20), y=rnorm(20))
tmp.lm <- lm(y ~ x, data=tmp)
ci.plot(tmp.lm)
```

---

col.hh *Initializing Trellis Displays*

Description

Initialization of an R display device with the graphical parameters that rmh prefers.

Usage

```
col.hh()
```

Value

List of graphical parameters to be used in the theme argument to the trellis.device or trellis.par.set functions.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

trellis.device, trellis.par.get

Examples

```
## Not run:
if.R(r={
trellis.device(theme="col.hh") ## Open a device with the theme

trellis.device(theme=col.hh()) ## Open a device with the theme

trellis.par.set(theme=col.hh())## Add theme to already open device
},s={})

## End(Not run)
```

`cp.calc` *Rearranges and improves the legibility of the output from the stepwise function in S-Plus.*

Description

Rearranges and improves the legibility of the output from the

`stepwise` function in S-Plus. The output can be used for the Cp plot. `cp.calc` works only in S-Plus. Use

`regsubsets` in R. The example below works in both languages.

Usage

```
cp.calc(sw, data, y.name)

## S3 method for class 'cp.object'
print(x, ...)

## S3 method for class 'cp.object'
x[..., drop = TRUE]
```

Arguments

`sw`	Output from the S-Plus `stepwise` function.
`data`	Dataset name from which `"sw"` was calculated.
`y.name`	Name of response variable for which `"sw"` was calculated.
`x`	Object of class `"cp.object"`.
`...`	Additional arguments to `"["` or `"print"`.
`drop`	Argument to the `print` function.

Value

`"cp.object"`, which is a data.frame containing information about each model that was attempted with additional attributes: `tss` total sum of squares, `n` number of observations, `y.name` response variable, `full.i` row name of full model. The columns are

`p`	number of parameters in the model
`cp`	Cp statistic
`aic`	AIC statistic
`rss`	Residual sum of squares
`r2`	R^2
`r2.adj`	Adjusted R^2
`xvars`	X variables
`sw.names`	Model name produced by `stepwise`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Examples

```
## This example is from Section 9.15 of Heiberger and Holland (2004).
data(usair)
if.R(s={usair <- usair}, r={})

splom(~usair, main="U.S. Air Pollution Data with SO2 response", cex=.5)
## export.eps(hh("regb/figure/regb.f1.usair.eps"))

usair$lnSO2 <- log(usair$SO2)
usair$lnmfg <- log(usair$mfgfirms)
usair$lnpopn <- log(usair$popn)

usair[1:3,]   ## lnSO2 is in position 8, SO2 is in position 1
              ## lnmfg is in position 9, lnpopn is in position 10

splom(~usair[, c(8,2,9,10,5:7)],
              main="U.S. Air Pollution Data with 3 log-transformed variables",
              cex=.5)
## export.eps(hh("regb/figure/regb.f2.usair.eps"))

if.R(s={
  usair.step <- stepwise(y=usair$lnSO2,
                         x=usair[, c(2,9,10,5:7)],
                         method="exhaustive",
                         plot=FALSE, nbest=2)
  ## print for pedagogical purposes only.  The plot of cp ~ p is more useful.
  ## The line with rss=1e35 is a stepwise() bug, that we reported to S-Plus.
  print(usair.step, digits=4)
  usair.cp <- cp.calc(usair.step, usair, "lnSO2")
  ## print for pedagogical purposes only.  The plot of cp ~ p is more useful.
  usair.cp
  tmp <- (usair.cp$cp <= 10)
  usair.cp[tmp,]

  old.par <- par(mar=par()$mar+c(0,1,0,0))
  tmp <- (usair.cp$cp <= 10)
  plot(cp ~ p, data=usair.cp[tmp,], ylim=c(0,10), type="n", cex=1.3)
  abline(b=1)
  text(x=usair.cp$p[tmp], y=usair.cp$cp[tmp],
       row.names(usair.cp)[tmp], cex=1.3)
  title(main="Cp plot for usair.dat, Cp<10")
  par(old.par)
```

```
## export.eps(hh("regb/figure/regb.f3.usair.eps"))
},r={
  usair.regsubset <- leaps::regsubsets(lnSO2~lnmfg+lnpopn+precip+raindays+temp+wind,
                                       data=usair, nbest=2)
  usair.subsets.Summary <- summaryHH(usair.regsubset)
  tmp <- (usair.subsets.Summary$cp <= 10)
  usair.subsets.Summary[tmp,]
  plot(usair.subsets.Summary[tmp,], statistic='cp', legend=FALSE)

  usair.lm7 <- lm.regsubsets(usair.regsubset, 7)
  anova(usair.lm7)
  summary(usair.lm7)
})

vif(lnSO2 ~ temp + lnmfg + lnpopn + wind + precip + raindays, data=usair)

vif(lnSO2 ~ temp + lnmfg + wind + precip, data=usair)

usair.lm <- lm(lnSO2 ~ temp + lnmfg + wind + precip, data=usair)
anova(usair.lm)
summary(usair.lm, corr=FALSE)
```

`datasets` *Datasets for Statistical Analysis and Data Display, Heiberger and Holland*

Description

Most of the datasets are described in the book *Statistical Analysis and Data Display*.

For `ProfChal`, see `plot.likert`.

`AudiencePercent` is from personal communication by the market researcher who did the study.

`SFF8121` is student evaluations of my class compared to the average of all graduate classes in the Spring 2010 semester. Personal communication from the Temple University Office of the Provost to me.

`ProfDiv` is "Profit-and-Dividend Status of 348 Corportations in the United States for the period from 1929 to 1935" from Brinton WC (1939), *Graphic Presentation*. Brinton Associates. http://www.archive.org/details/graphicpresentat00brinrich.

`NZScienceTeaching` is from New Zealand Ministry of Research Science and Technology(2006), "Staying in Science." http://www.morst.govt.nz/Documents/publications/researchreports/Staying-in-Science-summary.pdf.

`PoorChildren` is from "Poor Children, Working Parents", Analysis of data from the CensusBureau's American Community Survey. Comparison of Census areas of 100,000 or more people, based on samples from 2005 to 2009.

Source: Data from the U.S. Census Bureau's American Community Survey; analysis by Andrew A. Beveridge, QueensCollege. Copyright 2011 The New York Times Company

http://www.nytimes.com/imagepages/2011/12/03/opinion/03blow-ch.html?ref=opinion

http://www.nytimes.com/2011/12/03/opinion/blow-newts-war-on-poor-children.html?_r=1

Naomi Robbins and I discuss the PoorChildren example in the Forbes online column: http://www.forbes.com/sites/naomirobbins/2011/12/20/alternative-to-charles-blows-figure-in-newts-war-on-poor-

```
demo(PoorChildren, package="HH")
```

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

`dchisq.intermediate` *Intermediate f and chisq functions to simplify writing for both R and S-Plus.*

Description

Intermediate f and chisq functions to simplify writing for both R and S-Plus.

Usage

```
dchisq.intermediate(x, df, ncp=0, log=FALSE)
pchisq.intermediate(q, df, ncp=0, lower.tail=TRUE, log.p=FALSE)
qchisq.intermediate(p, df, ncp=0, lower.tail=TRUE, log.p=FALSE)
df.intermediate(x, df1, df2, ncp=0, log=FALSE)
pf.intermediate(q, df1, df2, ncp=0, lower.tail=TRUE, log.p=FALSE)
qf.intermediate(p, df1, df2, ncp=0, lower.tail=TRUE, log.p=FALSE)
```

Arguments

`x,p,q, df,df1,df2, ncp, log,log.p, lower.tail`
: See `pchisq` and `pf`. Some arguments don't exist in S-Plus. That is why these functions are needed.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

defunct *Defunct Functions in Package 'HH'*

Description

The function names listed here are no longer part of the HH package. Their task has been assigned to different function names.

Usage

```
anova.mean(...)               ## anovaMean
vcov.sufficient(...)          ## vcovSufficient
aov.sufficient(...)           ## aovSufficient
print.glht.mmc.multicomp(...) ## print.mmc.multicomp
coef.arima.HH(...)            ## coefArimaHH
glht.mmc(...)                 ## mmc
odds.ratio(...)               ## OddsRatio
plot.odds.ratio(...)          ## plotOddsRatio
persp.plane(...)              ## perspPlane
persp.floor(...)              ## perspFloor
persp.back.wall.x(...)        ## perspBack.wall.x
persp.back.wall.y(...)        ## perspBack.wall.y
persp.setup(...)              ## not used in R, S-Plus only
plot.hov(...)                 ## hovPlot
plot.hov.bf(...)              ## hovPlot.bf
plot.matchMMC(...)            ## plotMatchMMC
seqplot.forecast(...)         ## seqplotForecast
lm.case(...)                  ## case.lm
```

Arguments

...	other arguments.

Details

Some of these function names have been replaced by using them as methods. Some have had their spelling changed to remove the '.' character.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

diag.maybe.null *Returns a value for the diagonal of NA and NULL arguments.*

Description

Returns the argument for the diagonal of NA and NULL arguments. For all other arguments, it calls the regular diag function.

Usage

```
diag.maybe.null(x, ...)
```

Arguments

x	matrix, vector, NA,
...	Other arguments to diag.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

diag.

Examples

```
diag.maybe.null(NULL)
diag.maybe.null(NA)
diag.maybe.null(1:5)
```

do.formula.trellis.xysplom *Interprets model formulas for xysplom and extended bwplots*

Description

Interprets a model formula in the context of its data.frame.

Usage

```
do.formula.trellis.xysplom(formula, data, na.action = na.pass)
```

Arguments

formula	model formula
data	data.frame
na.action	see na.action

Value

A list containing three data.frames and three formula, one for each.

x	data.frame containing the variables on the right-hand side of the model formula.
y	data.frame containing the variables on the left-hand side of the model formula.
g	data.frame containing the variables, if any, after the conditioning bar | of the model formula.
x.formula	formula containing the right-hand side of the model formula.
y.formula	formula containing the left-hand side of the model formula.
g.formula	formula containing the formula after the conditioning bar | of the model formula.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

formula, na.action

Examples

```
tmp <- data.frame(y=1, x=2, z=3, g=4)
do.formula.trellis.xysplom( y ~ x + z | g, data=tmp)
```

---

emptyMainLeftAxisLeftStripBottomLegend

Remove main title, left axis tick labels, left strip, bottom legend from plot and keep the vertical spacing allocated to those items.

Description

Remove main title, left axis tick labels, left strip, bottom legend from plot and keep the vertical spacing allocated to those items. This function is used to prepare a trellis object to be placed next to another trellis object. The current object will have much of its annotation removed with the intent of sharing annotation with the other object. This is motivated by the ProfChal example in plot.likert.

Usage

```
emptyMainLeftAxisLeftStripBottomLegend(x)
```

Arguments

x	A "trellis" object.

Details

We manipulate the items inside the trellis object.

Value

A "trellis" object with the stated items replaced by non-printing values. The vertical spacing of the original object is retained.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

The manipulations are similar to those in the c.trellis and related functions in the latticeExtra package.

See Also

plot.likert

Examples

```
## This is a small example.
## See ?plot.likert for the complete example including motivation.
##
require(grid)
require(lattice)
require(latticeExtra)
require(HH)

data(ProfChal)

tmp <- data.matrix(ProfChal[,1:5])
rownames(tmp) <- ProfChal$Question
ProfChal.list <- split.data.frame(tmp, ProfChal$Subtable)

Empl <- ProfChal.list[[2]]

pct   <- likert(Empl, as.percent="noRightAxis", xlab="Percent")
pct
count <- likert(Empl, rightAxis=TRUE,
                xlab="Count", ylab.right="Row Count Totals",
```

```
                scales=list(x=list(at=c(0, 100, 200))))
count
countEmptied <- HH:::emptyMainLeftAxisLeftStripBottomLegend(count)
countEmptied

tmp <- update(resizePanels(c(pct, countEmptied, y.same=TRUE, layout=c(2,1)), w=c(.8, .2)),
              scales=list(y=list(alternating=3, limits=count$y.limits),
                          x=list(at=list(pct$x.scales$at, count$x.scales$at),
                                 labels=list(pct$x.scales$labels,
                                             count$x.scales$labels))),
              xlab=c(" ", pct$xlab, " ", count$xlab),
              between=list(x=1))
tmp
```

`export.eps` *Exports a graph to an EPS file.*

Description

Exports a graph from the current device in R, or the `graphsheet` in S-Plus, to an `EPS` file.

Usage

```
export.eps(FileName.in, Name.in="GSD2", ...)
```

Arguments

`FileName.in`	name of file to be created.
`Name.in`	Name of graphsheet in S-Plus, ignored in R.
`...`	other arguments in R, ignored in S-Plus.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`dev2`.

Examples

```
## Not run:
trellis.device()
plot(1:10)
export.eps("abcd.eps")

## End(Not run)
```

extra *Miscellaneous functions that I wish were in or consistent between S-Plus and R.*

Description

Miscellaneous functions that I wish were in or consistent between S-Plus and R.

Usage

```
as.rts(x, ...)

## S3 method for class 'ts'
units(x)

title.trellis(main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
    line = NA, outer = FALSE, axes=NULL, ...)

title.grob(main=NULL, y=.985, gp=gpar(cex=1.5))

## S3 method for class 'arima.model'
as.character(x, ...)

arima.model(x)

coefArimaHH(object, ...)

.arima.info.names.not.ordered (model)
```

Arguments

x	vector or time series
...	Additional arguments.
main, sub, xlab, ylab, line, outer, axes	See title.
model	A time series model specification in the S-Plus notation.
object	"arima" object in S-Plus.
y, gp	See grid.text in R.

Value

The result object of arima.model has class "arima.model"

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

arma.loop

F.curve *plot a chisquare or a F-curve.*

Description

Plot a chisquare or a F-curve. Shade a region for rejection region or do-not-reject region. F.observed and chisq.observed plots a vertical line with arrowhead markers at the location of the observed xbar and outlines the area corresponding to the p-value.

Usage

```
F.setup(df1=1,
        df2=Inf,
        ncp=0,
        log.p=FALSE,
        xlim.in=c(0, 5),
        ylim.in=range(c(0, 1.1*df.intermediate(x=seq(.5,1.5,.01),
                       df1=df1, df2=df2, ncp=ncp, log=log.p))),
        main.in=main.calc,
        ...)

F.curve(df1=1,
        df2=Inf,
        ncp=0,
        log.p=FALSE,
        alpha=.05,
        critical.values=f.alpha,
        f=seq(0, par()$usr[2], length=109),
        shade="right", col=par("col"),
        axis.name="f",
        ...)

F.observed(f.obs, col="green",
           df1=1,
           df2=Inf,
           ncp=0,
           log.p=FALSE,
           axis.name="f",
           shade="right",
           shaded.area=0,
           display.obs=TRUE)

chisq.setup(df=1,
```

```
            ncp=0,
            log.p=FALSE,
          xlim.in=c(0, qchisq.intermediate(p=1-.01, df=df, ncp=ncp, log.p=log.p)),
          ylim.in=range(c(0, 1.1*dchisq.intermediate(x=seq(max(0.5,df-2),df+2,.01),
                          df=df, ncp=ncp, log=log.p))),
            main.in=main.calc,
            ...)

chisq.curve(df=1,
            ncp=0,
            log.p=FALSE,
            alpha=.05,
            critical.values=chisq.alpha,
            chisq=seq(0, par()$usr[2], length=109),
            shade="right", col=par("col"),
            axis.name="chisq",
            ...)

chisq.observed(chisq.obs, col="green",
               df=1,
               ncp=0,
               log.p=FALSE,
               axis.name="chisq",
               shade="right",
               shaded.area=0,
               display.obs=TRUE)
```

Arguments

`xlim.in, ylim.in`	Initial settings for `xlim, ylim`. The defaults are calculated for the degrees of freedom.
`df, df1, df2, ncp, log.p`	Degrees of freedom, non-centrality parameter, probabilities are given as log(p). See `pchisq` and `pf`.
`alpha`	Probability of a Type I error. `alpha` is a vector of one or two values. If one value, it is the right alpha. If two values, they are the `c(left.alpha, right.alpha)`.
`critical.values`	Critical values. Initial values correspond to the specified `alpha` levels. A scalar value implies a one-sided test on the right side. A vector of two values implies a two-sided test.
`main.in`	Main title.
`shade`	Valid values for shade are "right", "left", "inside", "outside", "none". Default is "right" for one-sided critical.values and "outside" for two-sided critical values.
`col`	color of the shaded region and the area of the shaded region.

shaded.area	Numerical value of the area. This value may be cumulated over two calls to the function (one call for left, one call for right). The shaded.area is the return value of the function. The calling program is responsible for the cumulation.
display.obs	Logical. If TRUE, print the numerical value of the observed value, plot a vertical abline at the value, and use it for showing the p-value. If FALSE, don't print or plot the observed value; just use it for showing the p-value.
f,chisq	Values used to draw curve. Replace them if more resolution is needed.
f.obs, chisq.obs	Observed values of statistic. p-values are calculated for these values.
axis.name	Axis name.
...	Other arguments which are ignored.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

```
old.omd <- par(omd=c(.05,.88, .05,1))
chisq.setup(df=12)
chisq.curve(df=12, col='blue')
chisq.observed(22, df=12)
par(old.omd)

old.omd <- par(omd=c(.05,.88, .05,1))
chisq.setup(df=12)
chisq.curve(df=12, col='blue', alpha=c(.05, .05))
par(old.omd)

old.omd <- par(omd=c(.05,.88, .05,1))
F.setup(df1=5, df2=30)
F.curve(df1=5, df2=30, col='blue')
F.observed(3, df1=5, df2=30)
par(old.omd)

old.omd <- par(omd=c(.05,.88, .05,1))
F.setup(df1=5, df2=30)
F.curve(df1=5, df2=30, col='blue', alpha=c(.05, .05))
par(old.omd)
```

---

glhtWithMCP.993 *Retain averaging behavior that was previously available in glht.*

Description

For some ANOVA models with two or more factors, we need to average over interaction terms. These functions use an older version of glht.mcp and mcp2matrix to do that averaging.

Usage

```
glhtWithMCP.993(model, linfct, ...)
mcp2matrix.993(model, linfct)
```

Arguments

model, linfct, ...	See glht

Details

mcp2matrix is taken from from multcomp_0.993-2.tar.gz/R/mcp.R.

glhtWithMCP.993 is based on glht.mcp in multcomp_1.0-0/R/glht.R with the call to mcp2matrix replaced by a call to mcp2matrix.993.

Value

See glht

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

MMC

---

gof.calculation *Calculate Box–Ljung Goodness of Fit for ARIMA models in S-Plus.*

Description

Calculate Box–Ljung Goodness of Fit for ARIMA models in S-Plus. In R we use the Box.test function.

Usage

```
gof.calculation(acf.list, gof.lag, n, n.parms)
```

Arguments

acf.list	An "acf" object.
gof.lag	The number of model parameters is the number of lags to use for computing the Portmanteau goodness of fit statistic
n	Number of residuals in model.
n.parms	Number of AR and MA parameters in the model.

Details

This function is isolated from the S-Plus `arima.diag` function. It is used only in S-Plus.

Value

See the `gof` value described in

`arima.diag` in S-Plus.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

`arima.diag` in S-Plus.

Examples

```
if.R(s={
co2.arima <- arima.mle(co2, list(list(order=c(0,1,1)),
                                 list(order=c(0,1,1), period=12)))
co2.acf <- acf(resid(co2.arima), plot=FALSE, lag=40)
co2.gof <- gof.calculation(co2.acf, 36, length(co2), 2)
xyplot(p.value ~ lag, data=co2.gof, panel=panel.gof,
       ylim=range(0, co2.gof$p.value))
},r={})
```

---

grid.yaxis.hh *make x- and y-axis labels*

Description

uses modified older version of grid functions. Includes optional specification of the axis labels.

Usage

```
grid.yaxis.hh(at = NULL, label = TRUE, main = TRUE, gp = gpar(),
              draw = TRUE, vp = NULL, labels)

make.yaxis.hh.labels(at, main, labels = at)

grid.xaxis.hh(at = NULL, label = TRUE, main = TRUE, gp = gpar(),
              draw = TRUE, vp = NULL, labels)

make.xaxis.hh.labels(at, main, labels = at)
```

Arguments

at, label, main, gp, draw, vp	See link[grid]{grid.xaxis}.
labels	label values if you don't want the defaults

Value

See link[grid]{grid.xaxis}.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

link[grid]{grid.xaxis}

GSremove — *Remove selected GraphSheetPages in the S-Plus Windows GUI Graphsheet*

Description

Remove selected GraphSheetPages in the S-Plus Windows GUI Graphsheet. This does the same task as right-click/delete on the tabs of the GraphSheet.

Usage

```
GSremove(pages, sheet = "GSD2$Page")
```

Arguments

pages	Page numbers in the tabs at the bottom of the Graphsheet.
sheet	Defaults to GSD2, the first name that is used when the graphsheet or trellis.device function is used.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

graphsheet in S-Plus.

Examples

```
## Not run:
trellis.device()
plot(1:10); plot(11:20); plot(21:30)
GSremove(c(1,3))

## End(Not run)
```

---

hh *Resolve filenames relative to the HH directory.*

Description

The pathnames in the HH package for all the datasets referenced in Heiberger and Holland (2004) are given relative to the `options()$HH.ROOT.DIR` directory. The pathnames for all the executable files in the online files accompanying Heiberger and Holland (2004) are given relative to the `options()$HHfile.ROOT.DIR` directory.

Usage

```
hh(file)     ## gives message about change in usage beginning with HH_2.3-17
hh.old(file) ## new name for function hh() prior to HH_2.3-17
hh.file(file)
hh.file.DOS(file, displayForCutAndPaste=TRUE)
```

Arguments

`file` Character string giving the pathname of a file.

For `hh`, the file is in the HH package and the pathname is relative to `options()$HH.ROOT.DIR`, the installed location of the HH package. The option is set automatically when the HH package is loaded by `library` or `require`. This function is normally used to access the datasets that come with the HH book and package.

For `hh.file` and `hh.file.DOS`, the file is in the HH online files which need to be independently downloaded from the HH book's website http://springeronline.com/0-387-40270-5. The pathname is relative to `options()$HHfile.ROOT.DIR`, which must be set by the user to correspond to the location where the HH online files are stored on the specific computer. The `file` argument to these two functions is the relative pathname exactly as specified in the captions of figures and tables in Heiberger and Holland (2004).

`displayForCutAndPaste`

Logical value. When `TRUE` (the default) the function `hh.file.DOS` prints the full pathname with the '"\"' file separator convention so it can be picked up and pasted into an editor that uses the MS DOS convention for file paths. The function `hh.file.DOS` always returns a value with the full pathname using the '"\\"' convention so it could be used as an argument to an R or S-Plus function.

Details

Beginning with HH_2.3-17, access to datasets with the notation `hh("abcde <- datasets/abcde.dat")` is defunct. Instead, use the notation `data(abcde)`. The old notation will generate an error with a message to use the new notation.

The files accessed with the notation `hh("chaptername/code/normpdf.r")` are defunct. Instead use the new files accessed with the notation `hh.old("scripts/Chxx-chaptername.r")`. The old notation will generate an error with a message to use the new notation.

The datasets from the Heiberger and Holland (2004) online files are all given paths relative to the beginning of the `hh` directory, which must be stored as `options()$HH.ROOT.DIR`. If you installed the HH datasets as part of the R or S-Plus HH package, then option `options()$HH.ROOT.DIR` is set automatically.

The code listings in Heiberger and Holland (2004) are all given paths relative to the beginning of the `hh.file` directory, which must be stored as `options()$HHfile.ROOT.DIR`. Most of the files are designed to be entered at the command line, and are not designed to be sourced. The primary use of the `hh.file` functions is to display the pathname of the file so it can be opened for use in an editor. The online files must be independently downloaded from the book's web site http://springeronline.com/0-387-40270-5. The `HHfile.ROOT.DIR` option must be set by the user to correspond to the location where the files are stored. The book recommends `options(HHfile.ROOT.DIR="c:/HOME/hh")` in Windows or `options(HHfile.ROOT.DIR="/usr/users/hh")` in Unix. See Appendix B of Heiberger and Holland (2004) for further details. The `options(HHfile.ROOT.DIR="something"` statement may need to be modified to match the location of the online files directory on your machine. If you use more than one computer, you may need a different value for the `HHfile.ROOT.DIR` option on each machine. This is the only change you will need to make in order to run any of our software or examples. The `hh`, `hh.file`, and `hh.file.DOS` functions are not changed.

Value

Fully expanded, absolute pathname for the input filename. `hh` and `hh.file` use the separator convention of the `file.path` function. `hh.file.DOS` returns the pathname using the '`"\\"`' separator convention. When `displayForCutAndPaste` is `TRUE`, `hh.file.DOS` prints the full pathname with the '`"\"`' convention, so it can be picked up and pasted into an editor that uses the MS DOS convention for file paths.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Examples

```
## hotdog <- read.table(hh.old("datasets/hotdog.dat"), header=TRUE)
## This form of data input for files from the text has been replaced by
```

```
## the alternate form
##     data(hotdog)


## Not run:
## Define the HHfile.ROOT.DIR option first.
## Define
##   options(HHfile.ROOT.DIR="c:/HOME/hh")  ## value recommended in Appendix B
## before using the hh.file() functions.
hh.file("relativefilepath")
hh.file.DOS("relativefilepath")
hh.file.DOS("relativefilepath", displayForCutAndPaste=FALSE)

## End(Not run)
```

HH.regsubsets *Display tabular results for Best Subsets Regression.*

Description

Print a tabular display of the results of Best Subsets Regression. This is an alternate display for the object from the regsubsets function. This function is based on regsubsets. The functions described here are designed for the HH package in R and use the leaps package in R. The leaps package is not in S-Plus, hence these functions do not work in the HH package for S-Plus.

Usage

```
`summaryHH`(object, ...)

## S3 method for class 'regsubsets'
summaryHH(object,
          names = abbreviate(dimnames(incidence)[[2]], minlength = abbrev),
          abbrev = 1, min.size = 1, max.size = dim(sumry$which)[2],
          statistic = c("bic", "cp", "adjr2", "rsq", "rss", "stderr"),
          las = par("las"),
          cex.subsets = 1, ..., main=statistic)

## S3 method for class 'summaryHH.regsubsets'
plot(x, ...,
          statistic="adjr2", legend=FALSE,
          col="darkgray", cex=1, pch=16,
          col.text="black", cex.text=1, col.abline="darkgray")
```

Arguments

object	An object of class "regsubsets".
x	An object of class "summaryHH.regsubsets".
statistic	Name of statistic to be plotted for each model.
...	Other arguments to be passed down to subsets.regsubsets and plot.
names	Abbreviations of variable names.
abbrev	minimum number of letters in each abbreviation.
min.size	minimum size subset to plot; default is 1.
max.size	maximum size subset to plot; default is number of predictors.
legend	logical variable, TRUE if the legend should be printed. If the legend is printed, the execution halts until the user clicks an empty space in the graph where the legend should be placed.
las	Orientation for model names on graph.
cex.subsets	can be used to change the relative size of the characters used to plot the regression subsets; default is 1.
main	"main" title for graph.
col, cex, pch	par values for dot locating statistic.
col.text, cex.text	par values for abbreviations of models on plot.
col.abline	par parameters for abline when the statistic is cp.

Value

summaryHH produces a table of models, with p, rsq, rss, adjr2, cp, bic, stderr for each. plot.summaryHH.regsubsets plots the specified statistic from the summary. All the others are support functions.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

regsubsets

hov *Homogeneity of Variance*

Description

Oneway analysis of variance makes the assumption that the variances of the groups are equal. Brown and Forsyth, 1974 present the recommended test of this assumption. The Brown and Forsyth test statistic is the F statistic resulting from an ordinary one-way analysis of variance on the absolute deviations from the median.

Usage

```
hov(x, data = sys.parent(), method = "bf") ## x is a formula

## users will normally use the formula above and will not call the
## method directly.
hov.bf(x, group,  ## x is the response variable
       y.name = deparse(substitute(x)),
       group.name = deparse(substitute(group)))
```

Arguments

x	Formula appropriate for oneway anova in hov. Response variable in hov.bf.
data	data.frame
method	Character string defining method. At this time the only recognized method is "bf" for the Brown–Forsyth method.
group	factor.
y.name	name of response variable, defaults to variable name in formula.
group.name	name of factor, defaults to variable name in formula.

Value

"htest" object for the hov test.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Brown, M.~B. and Forsyth, A.~B. (1974). *Robust tests for equality of variances*. *Journal of the American Statistical Association*, 69:364–367.

See Also

aov, hovPlot

Examples

```
data(turkey)

hov(wt.gain ~ diet, data=turkey)
hovPlot(wt.gain ~ diet, data=turkey)
```

---

if.R *Conditional Execution for R or S-Plus*

Description

if.R uses the is.R function to determine whether to execute the expression in the r argument or the expression in the s argument.

Usage

```
if.R(r, s)
```

Arguments

r	Any R expression, including a group of expressions nested in braces. Assignments made in this expression are available to the enclosing function.
s	Any S-Plus expression, including a group of expressions nested in braces. Assignments made in this expression are available to the enclosing function.

Details

Not all functions are in both implementations of the S language. In particular, panel functions for lattice in R (based on grid graphics) are very different from panel functions for trellis (based on the older graphics technology) in S-Plus.

Value

The result of the executed expression.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

is.R

Examples

```
if.R(r={"This is R."},
     s={"This is S-Plus"})
```

`interaction.positioned` *interaction method for positioned factors.*

Description

This is intended to be a method for `interaction` for positioned factors. Since `interaction` is not currently implemented as a generic, `interaction.positioned` is a standalone function. The result is assigned a `position`. The position for each interaction level is the position of the corresponding `a` factor plus a scaled level of the `b` factor. The default scale is .1.

Usage

```
interaction.positioned(...,  ## exactly two factors
             drop = FALSE, sep = ".",
             b.offset=0,
             b.scale=.1)
```

Arguments

`...`	exactly two factors. The first factor `a` is used as the major factor in sort order. The second factor `b` is used as minor factor in sort order.
`b.offset`	amount added to `position(b)` to adjust appearance.
`b.scale`	scale to relate units of `position(a)` to units of `position(b)`.
`drop, sep`	See `factor`.

Value

`"positioned"` object containing the ordinary interaction with a `"position"` attribute.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`positioned`.

Examples

```
a <- positioned(letters[c(1,2,3,1,2,3)], value=c(1,4,9))
b <- positioned(LETTERS[c(4,4,4,5,5,5)], value=c(1,2))
a.b <- interaction.positioned(a, b)
a.b.2 <- interaction.positioned(a, b, b.scale=.2)
b.a <- interaction.positioned(b, a)
```

`interaction2wt` *Plot all main effects and twoway interactions in a multifactor design*

Description

The main diagonal displays boxplots for the main effects of each factor. The off-diagonals show the interaction plots for each pair of factors. The `i,j` panel shows the same factors as the `j,i` but with the trace- and x-factor roles interchanged.

Usage

```
interaction2wt(x, ...)

## S3 method for class 'formula'
interaction2wt(x, data = sys.parent(), responselab, ...)

## Default S3 method:
interaction2wt(x,
               response.var,
               responselab = deparse(substitute(response.var)),
               responselab.expression = responselab,
               relation = list(x = "same", y = "same"),
               x.relation = relation$x,
               y.relation = relation$y,
               digits = 3,
               x.between=if (label.as.interaction.formula) 0 else 1,
               y.between=if (label.as.interaction.formula) 0 else 1,
               between,
               cex = 0.75,
               rot=c(0,0),
               panel.input = panel.interaction2wt,
               strip.input =
                if (label.as.interaction.formula) strip.default
                else strip.interaction2wt,
               par.strip.text.input = trellis.par.get()$add.text,
               scales.additional,
               main.in =
                 paste(responselab,
                       ": main effects and 2-way interactions",
                       sep = ""),
               xlab = "",
               ylab = "",
               simple=FALSE,
               box.ratio=if (simple) .32 else 1,
               label.as.interaction.formula=TRUE,
               ...,
               main.cex,
```

```
                key.cex.title=trellis.par.get()$par.xlab.text$cex,
                key.cex.text=trellis.par.get()$axis.text$cex,
                factor.expressions=names.x
)
```

Arguments

	Arguments when x is a formula.
	The object on which method dispatch is carried out.
	For the "formula" method, a formula describing the response variable and factors. The formula is generally of the form y ~ g1 + g2 + ... There may be one or more factors in the formula.
	For the "default" method, data.frame of factors. This is usually constructed by formula method from the input data and the input formula.
data	For the formula method, a data frame containing values for any variables in the formula. In the R version, if not found in data, or if data is unspecified, the variables are looked for in the environment of the formula.
responselab	Character name of response variable, defaults to the name of the response variable in the formula.
responselab.expression	plotmath or character name of response variable, defaults to responselab.
...	additional arguments, primarily trellis arguments.
response.var	For the "default" method, the response variable. This is usually constructed by formula method from the input data and the input formula.
simple	logical. TRUE if simple effects are to be displayed. Arguments simple.offset, simple.scale, and col.by.row may also be needed. See panel.interaction2wt for details.
box.ratio	xyplot.
	Trellis/Lattice arguments. Default values are set by the the formula method. The user may override the defaults. See also
	xyplot.
relation	trellis argument.
x.relation	x value of relation argument.
y.relation	y value of relation argument.
digits	doesn't do anything at the moment
x.between	x value of between argument.
y.between	y value of between argument.
between	trellis/lattice between argument. If used, between has precedence over both the x.between and y.between arguments.
cex	S-Plus: changes the size of the median dot in the boxplots. R: doesn't do anything.
panel.input	panel function. Default is panel.interaction2wt.

`label.as.interaction.formula`
: logical. If `TRUE`, each panel has a single strip label of the form `y ~ a | b`. If `FALSE`, each panel has a pair of strip labels, one for the `trace` factor and one for the `x` factor.

`strip.input`
: strip function. Default depends on the value of `label.as.interaction.formula`.

`par.strip.text.input`
: `par.strip.text` argument.

`scales.additional`
: additional arguments to `scales` argument of `interaction.positioned`.

`main.in`
: Text of main title.

`xlab`
: No effect.

`ylab`
: No effect.

`main.cex`
: `cex` for main title.

`key.cex.title`
: `cex` key title. Defaults to `cex` for `xlab`.

`key.cex.text`
: `cex` group names in key. Defaults to `cex` for `axis.text`.

`factor.expressions`
: Expressions for titles of keys and `xlab` for each column. Defaults to the names of the factors in the input formula.

`rot`
: Rotation of x tick labels and y tick labels. Only 0 and 90 will look good.

Value

`"trellis"` object containing the plot.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

`panel.interaction2wt`

Examples

```
data(vulcan)
if.R(s={old.omd=par(omd=c(.1,1,.03,1))}, r={})
if.R(r=
     interaction2wt(wear ~ filler + pretreat + raw, data=vulcan,
                    par.strip.text=list(cex=.8))
     ,s=
     interaction2wt(wear ~ filler + pretreat + raw, data=vulcan,
```

```
                    par.strip.text=list(cex=.8),
                    key.in=list(x=-3.5))
)
interaction2wt(wear ~ filler + raw, data=vulcan,
               simple=TRUE)
interaction2wt(wear ~ filler + raw, data=vulcan,
               simple=TRUE, simple.scale=c(filler=.15, raw=.2),
               xlim=c(.3, 5.6))
if.R(s={par(old.omd)}, r={})

if.R(r={
  ToothGrowth <- ToothGrowth  ## local copy
  ToothGrowth$dose <- positioned(ToothGrowth$dose) ## modify local copy
print(
  anova(aov(len ~ supp*dose, data=ToothGrowth))
)
print(
  interaction2wt(len ~ supp + dose, data=ToothGrowth)
)

esoph=esoph
esoph$rate=with(esoph, ncases/ncontrols)

position(esoph$alcgp) <- 2:5
position(esoph$tobgp) <- 2:5

print(
interaction2wt(rate ~ agegp + alcgp + tobgp, esoph, rot=c(90,0),
               par.strip.text=list(cex=.8))
)

old.cex <- trellis.par.set(par.xlab.text=list(cex=.8))
print(
interaction2wt(rate ~ agegp + alcgp + tobgp, esoph, rot=c(90,0),
               par.strip.text=list(cex=.8),
               factor.expressions=c(
                  agegp=expression(Age~~(years)),
                  alcgp=expression(Alcohol~
                                   bgroup("(",scriptstyle(frac(gm, day)),")")),
                  tobgp=expression(Tobacco~
                                   bgroup("(",scriptstyle(frac(gm, day)),")"))),
               responselab.expression="Cancer\nRate",
           main.in="Esophogeal Cancer Rate ~ Alcohol Consumption + Tobacco Consumption",
               main.cex=1.2)
)

par(old.cex)

esoph.aov <- aov(rate ~ agegp + alcgp + tobgp, data=esoph)
print(
anova(esoph.aov)
)
}, s={})
```

intxplot *Interaction plot, with an option to print standard error bars.*

Description

Interaction plot, with an option to print standard error bars. There is an option to offset group lines to prevent the bars from overprinting.

Usage

```
intxplot(x, data=sys.parent[1], groups.in,
          scales,
          key.length=1,
          key.lines,
          key=TRUE,
          trace.factor.name=deparse(substitute(groups.in)),
          x.factor.name=x.factor,
          xlab=x.factor.name,
          main=list(main.title, cex=main.cex),
          condition.name="condition",
          panel="panel.intxplot",
          summary.function="sufficient",
          se,
          ...,
          data.is.summary=FALSE,
          main.title=paste(
            "Interactions of", trace.factor.name, "and",
            x.factor.name,
            if (length(x[[3]]) > 1)
            paste("|", condition.name.to.use)),
          main.cex=1.5)

panel.intxplot(x, y, subscripts, groups, type = "l", ..., se, cv=1.96,
               offset.use=(!missing(groups) && !missing(se)),
               offset.scale=2*max(as.numeric.positioned(groups)),
               offset=
               as.numeric.positioned(groups[match(levels(groups),
                                     groups)]) / offset.scale,
               rug.use=offset.use)
```

Arguments

x	For intxplot, a formula with a factor as the predictor variable. For panel.intxplot, standard argument for panel functions.
data	data.frame, as used in xyplot.

groups.in	groups.in, as used in xyplot.
scales	Optional, additional arguments for the standard scales in xyplot.
key.length	Number of columns in the key.
key.lines	default value for the lines argument of key.
key	logical. If TRUE, draw the key.
trace.factor.name	Name of the grouping variable.
x.factor.name	name of the dependent variable.
xlab	as in xyplot, defaults to the name of the predictor variable from the formula.
main	as in xyplot. Defaults to the main.title argument.
panel	as in xyplot. Defaults to the "panel.intxplot".
condition.name	name of the conditioning variable.
summary.function	The default sufficient finds the mean, standard deviation, and sample size of the response variable for each level of the conditioning factor. See sufficient.
se	standard errors to be passed to panel.intxplot. se Missing, logical, or a numeric vector. If missing or FALSE, standard errors are not plotted. If se=TRUE in intxplot, the standard errors are calculated from the sufficient statistics for each group as the group's standard deviation divided by the square root of the group's observation count. If se is numeric vector, it is evaluated in the environment of the sufficient statistics. the se argument to panel.intxplot must be numeric. ,
...	In intxplot, arguments for panel.intxplot. In panel.intxplot, arguments for panel.superpose.
data.is.summary	logical, defaults to FALSE under the assumption that the input data.frame is the original data and the intxplot function will generate the summary information (primarily standard deviation sd and number of observations nobs for each group). When TRUE, the standard error calculation assumes variables sd and nobs are in the dataset.
main.title	Default main title for plot.
main.cex	Default character expansion for main title.
y, subscripts, groups, type	Standard arguments for panel functions.
cv	critical value for confidence intervals. Defaults to 1.96.
offset.use	logical. If TRUE, offset the endpoints of each group.
offset.scale	Scale number indicating how far apart the ends of the groups will be placed. Larger numbers make them closer together.
offset	Actual numbers by which the end of the groups are offset from their nominal location which is the as.numeric of the group levels.
rug.use	logical. If TRUE, display a rug for the endpoints of each group.

Value

"trellis" object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

sufficient

Examples

```
## This uses the same data as the HH Section 12.13 rhizobium example.

data(rhiz.clover)

## interaction plot, no se
intxplot(Npg ~ strain, groups=comb, data=rhiz.clover)

## interaction plot, individual se for each treatment combination
intxplot(Npg ~ strain, groups=comb, data=rhiz.clover, se=TRUE)

## Rescaled to allow the CI bars to stay within the plot region
intxplot(Npg ~ strain, groups=comb, data=rhiz.clover, se=TRUE,
         ylim=range(rhiz.clover$Npg))

## interaction plot, common se based on ANOVA table
intxplot(Npg ~ strain, groups=comb, data=rhiz.clover,
         se=sqrt(sum((nobs-1)*sd^2)/(sum(nobs-1)))/sqrt(5))

## Rescaled to allow the CI bars to stay within the plot region
intxplot(Npg ~ strain, groups=comb, data=rhiz.clover,
         se=sqrt(sum((nobs-1)*sd^2)/(sum(nobs-1)))/sqrt(5),
         ylim=range(rhiz.clover$Npg))

## change distance between endpoints
intxplot(Npg ~ strain, groups=comb, data=rhiz.clover,
         se=TRUE, offset.scale=20)

## When data includes the nobs and sd variables, data.is.summary=TRUE is needed.
intxplot(Npg ~ strain, groups=comb,
         se=sqrt(sum((nobs-1)*sd^2)/(sum(nobs-1)))/sqrt(5),
         data=sufficient(rhiz.clover, y="Npg", c("strain","comb")),
         data.is.summary=TRUE,
         ylim=range(rhiz.clover$Npg))
```

ladder *Draw a "ladder of powers" plot, plotting each of several powers of y against the same powers of x.*

Description

Draw a "ladder of powers" plot, plotting each of several powers of y against the same powers of x. The powers are

```
result <- data.frame(-1/x, -1/sqrt(x), log(x), sqrt(x), x, x^2)
names(result) <- c(-1, -.5, 0, .5, 1, 2)
```

Usage

```
ladder(formula.in, data=sys.parent(),
       main.in="Ladders of Powers",
       panel.in=panel.cartesian,
       xlab=deparse(formula.in[[3]]),
       ylab=deparse(formula.in[[2]]),
       scales=list(alternating=if.R(s=TRUE, r=FALSE),
         labels=FALSE, ticks=FALSE, cex=.6),
       par.strip.text=list(cex=.6),
       cex=.5, pch=16, between=list(x=.3, y=.3),
       dsx=xlab,
       dsy=ylab,
       ladder.function=ladder.f,
       strip.number=if.R(r=2, s=1),
       strip.names,
       strip.style=1,
       strip,
       oma=c(0,0,0,0),  ## S-Plus
       axis3.line=.61,
       layout=c(length(tmp$x.power), length(tmp$y.power)),
       axis.key.padding = 10, ## R right axis
       key.axis.padding = 10, ## R top axis
       useOuter=TRUE, ## R useOuterStrips(combineLimits(result))
       ...)

ladder3(x, y,
        dsx=deparse(substitute(x)),
        dsy=deparse(substitute(y)),
        ladder.function=ladder.f)

ladder.f(x)

ladder.fstar(x)
```

```
strip.ladder(which.given,
             which.panel,
             var.name,
             factor.levels,
             shingle.intervals,
             par.strip.text=trellis.par.get("add.text"),
             strip.names=c(TRUE,TRUE),
             style=1,
             ...)
```

Arguments

formula.in	A formula with exactly one variable on each side.
data	data.frame
main.in	main title for xyplot
panel.in	panel.cartesian has many arguments in addition to the arguments in panel.xyplot. Any replacement panel function must have those argument names, even if it doesn't do anything with them.
xlab, ylab	Trellis arguments, default to right- and left-sides of the formula.in.
strip	Strip function. Our default is strip.ladder (see below). The other viable argument value is FALSE.
cex, pch, between, scales, layout	arguments for xyplot.
dsx, dsy	Names to be used as level names in ladder.function for the generated factor distinguishing the powers. They default to xlab, ylab. For long variable names, an abbreviated name here will decrease clutter in the ladder of powers plot. These names are not visible in the plot when strip=FALSE.
ladder.function	function to use to create data.frame of powers of input variable.
strip.number	Number of strip labels in each panel of the display. 0: no strip labels; 1: one strip label of the form y^p ~ x^q; 2: two strip labels of the form ylab: y^p and xlab: x^q, where p and q are the powers returned by ladders; y and x are the arguments dsy and dsx.
useOuter	logical, defaults to TRUE. In R, this implies that strip.number is forced to 2 and that the resulting "trellis" object will be sent through useOuterStrips(combineLimits(result)). This argument is ignored by S-Plus.
strip.style	style argument to strip.
oma	argument to par in S-Plus.
...	other arguments to xyplot.
axis3.line	extra space to make the top axis align with the top of the top row of panels. Trial and error to choose a good value.
axis.key.padding	Extra space on right of set of panels in R.

`key.axis.padding`
: Extra space on top of set of panels in R.

`x, y`
: variables.

`which.given, which.panel, var.name, factor.levels, shingle.intervals, par.strip.text`
: See `strip.default`.

`strip.names, style`
: We always print the `strip.names` in `style=1`. Multicolored styles are too busy.

Details

The `ladder` function uses `panel.cartesian` which is defined differently in R (using grid graphics) and S-Plus (using traditional graphics). Therefore the fine control over appearance uses different arguments or different values for the same arguments.

Value

`ladder` returns a `"trellis"` object.

The functions `ladder.fstar` and `ladder.f` take an input vector `x` of non-negative values and construct a data.frame by taking the input to the powers `c(-1, -.5, 0, .5, 1, 2)`, one column per power. `ladder.f` uses the simple powers and `ladder.fstar` uses the scaled Box–Cox transformation.

ladder.fstar	ladder.fstar	notation
(x^p - 1)/p	(x^p - 1)/p	p
(1/x - 1)/(-1)	(1/x - 1)/(-1)	-1.0
(1/sqrt(x)-1)/(-.5)	(1/sqrt(x)-1)/(-.5)	-0.5
log(x)	log(x)	0.0
((sqrt(x)-1)/.5)	((sqrt(x)-1)/.5)	0.5
x-1	x-1	1.0
(x^2 - 1)/2	(x^2 - 1)/2	2.0

`ladder3` takes two vectors as arguments. It returns a `data.frame` with five columns:

`X, Y`: data to be plotted. The column `X` contains the data from the input `x` taken to all the powers and aligned with the similarly expanded column `Y`.

`x, y`: symbolic labeling of the power corresponding to `X,Y`.

`group`: result from pasting the labels in `x, y` with `*` between them.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Hoaglin, D.~C., Mosteller, F., and Tukey, J.~W., editors (1983). *Understanding Robust and Exploratory Data Analysis*. Wiley.

Box, G. E.~P. and Cox, D.~R. (1964). An analysis of transformations. *J. Royal Statist Soc B*, 26:211–252.

See Also

panel.cartesian

Examples

```
data(tv)

## default
## R: outer strip labels
## S-Plus: one strip label per panel (too full for this example, see below)
ladder(life.exp ~ ppl.per.phys, data=tv,
       main="Ladder of Powers for Life Expectancy and People per Physician")

## Not run:
## one strip label
if.R(r=ladder(life.exp ~ ppl.per.phys, data=tv, strip.number=1, useOuter=FALSE,
              dsx="ppp", dsy="le"),
     s=ladder(life.exp ~ ppl.per.phys, data=tv, strip.number=1,
              dsx="ppp", dsy="le")) ## S-Plus default

## two strip labels
if.R(r=ladder(life.exp ~ ppl.per.phys, data=tv, strip.number=2, useOuter=FALSE),
     s=ladder(life.exp ~ ppl.per.phys, data=tv, strip.number=2,
              axis3.line=1.2))

## outer strip labels
if.R(r=ladder(life.exp ~ ppl.per.phys, data=tv, useOuter=TRUE), ## R default
     s={}) ## S-Plus not available

## no strip labels (probably silly, but possible)
if.R(r=ladder(life.exp ~ ppl.per.phys, data=tv, strip.number=0, useOuter=FALSE),
     s=ladder(life.exp ~ ppl.per.phys, data=tv, strip.number=0,
              axis3.line=0))

## End(Not run)
```

---

legendGrob2wt *place separate keys to the left of each row of a trellis*

Description

Each key is created and then inserted into a single grob.

Usage

```
legendGrob2wt(...)
```

Arguments

...	key1, key2, etc. Each key will normally be the result of a draw.key with draw=FALSE.

Value

A Grid frame object (that inherits from 'grob').

Author(s)

Richard M. Heiberger, with asssistance from Deepayan Sarkar.

See Also

interaction2wt

---

likert *Diverging stacked barcharts for Likert, semantic differential, rating scale data, and population pyramids.*

Description

Constructs and plots diverging stacked barcharts for Likert, semantic differential, rating scale data, and population pyramids.

Usage

```
likert(x, ...)
likertplot(x, ...)
## S3 method for class 'likert'
plot(x, ...)

## S3 method for class 'formula'
plot.likert(x, data, ReferenceZero=NULL, value, levelsName="",
            scales.in=NULL,   ## use scales=
            between=list(x=1 + (horizontal), y=.5 + 2*(!horizontal)),
            auto.key.in=NULL, ## use auto.key=
            panel.in=NULL,    ## use panel=
            horizontal=TRUE,
            par.settings.in=NULL, ## use par.settings=
            ...,
            as.percent = FALSE,
            ## titles
```

```
              ylab= if (horizontal) {
                if (length(x)==3)
                  deparse(x[[2]])
                else
                  "Question"
              }
              else
              if (as.percent != FALSE) "Percent" else "Count",
              xlab= if (!horizontal) {
                if (length(x)==3)
                  deparse(x[[2]])
                else
                  "Question"
              }
              else
              if (as.percent != FALSE) "Percent" else "Count",
              main = x.sys.call,
              ## right axis
              rightAxisLabels = rowSums(data.list$Nums),
              rightAxis = !missing(rightAxisLabels),
              ylab.right = if (rightAxis) "Row Count Totals" else NULL,
              xlab.top = NULL,

              ## scales
              xscale.components = xscale.components.top.HH,
              yscale.components = yscale.components.right.HH,
              xlimEqualLeftRight = FALSE,
              xTickLabelsPositive = TRUE,
              ## row sequencing
              as.table=TRUE,
              positive.order=FALSE,
              data.order=FALSE,
              reverse=ifelse(horizontal, as.table, FALSE),
              ## resizePanels arguments
              h.resizePanels=sapply(result$y.used.at, length),
              w.resizePanels=sapply(result$x.used.at, length),
              ## color options
              reference.line.col="gray65",
              key.border.white=TRUE,
              col=likertColor(Nums.attr$nlevels,
                ReferenceZero=ReferenceZero,
                colorFunction=colorFunction,
                colorFunctionOption=colorFunctionOption),
              colorFunction="diverge_hcl",
              colorFunctionOption="lighter"
              )
## Default S3 method:
plot.likert(x,
```

```
              positive.order=FALSE,
              ylab=names(dimnames(x)[1]),
              xlab=if (as.percent != FALSE) "Percent" else "Count",
              main=xName,
              reference.line.col="gray65",
              col.strip.background="gray97",
              col=likertColor(attr(x, "nlevels"),
                ReferenceZero=ReferenceZero,
                colorFunction=colorFunction,
                colorFunctionOption=colorFunctionOption),
              colorFunction="diverge_hcl",
              colorFunctionOption="lighter",
              as.percent=FALSE,
              par.settings.in=NULL,
              horizontal=TRUE,
              ReferenceZero=NULL,
              ...,
              key.border.white=TRUE,
              xName=deparse(substitute(x)),
              rightAxisLabels=rowSums(abs(x)),
              rightAxis=!missing(rightAxisLabels),
              ylab.right=if (rightAxis) "Row Count Totals" else NULL,
              panel=panel.barchart,
              xscale.components=xscale.components.top.HH,
              yscale.components=yscale.components.right.HH,
              xlimEqualLeftRight=FALSE,
              xTickLabelsPositive=TRUE,
              reverse=FALSE)

    ## S3 method for class 'array'
    plot.likert(x,
                condlevelsName=paste("names(dimnames(", xName, "))[-(1:2)]",
                                     sep=""),
                xName=deparse(substitute(x)),
                main=paste("layers of", xName, "by", condlevelsName),
                ...)

    ## S3 method for class 'likert'
    plot.likert(x, ...)  ## See Details

    ## S3 method for class 'list'
    plot.likert(x,  ## named list of matrices, 2D tables,
                                  ## 2D ftables, or 2D structables,
                                  ## or all-numeric data.frames
                condlevelsName="ListNames",
                xName=deparse(substitute(x)),
                main=paste("List items of", xName, "by", condlevelsName),
                layout=if (length(dim.x) > 1) dim.x else {
```

```
                        if (horizontal) c(1, length(x)) else c(length(x), 1)},
    positive.order=FALSE,
    strip=!horizontal,
    strip.left=horizontal,
    strip.left.values=names(x),
    strip.values=names(x),
    strip.par=list(cex=1, lines=1),
    strip.left.par=list(cex=1, lines=1),
    horizontal=TRUE,
    ...,
  rightAxisLabels=sapply(x, function(x) rowSums(abs(x)), simplify = FALSE),
    rightAxis=!missing(rightAxisLabels),
    resize.height.tuning=-.5,
    resize.height=if (missing(layout) || length(dim.x) != 2) {
      c("nrow","rowSums")
    } else {
      rep(1, layout[2])
    },
    resize.width=if (missing(layout)) {1 } else {
      rep(1, layout[1])
    },
    box.ratio=if (
      length(resize.height)==1 &&
      resize.height == "rowSums") 1000 else 2,
    xscale.components=xscale.components.top.HH,
    yscale.components=yscale.components.right.HH)

## S3 method for class 'table'
plot.likert(x, ..., xName=deparse(substitute(x)))
## S3 method for class 'ftable'
plot.likert(x, ..., xName=deparse(substitute(x)))
## S3 method for class 'structable'
plot.likert(x, ..., xName=deparse(substitute(x)))

## S3 method for class 'data.frame'
plot.likert(x, ..., xName=deparse(substitute(x)))


xscale.components.top.HH(...)
yscale.components.right.HH(...)
```

Arguments

x	For the formula method, a model formula. All terms in the formula must be the names of columns in the data.frame argument `data` or the special abbreviation `.` only on the right-hand-side. Functions of the names will not work. The right-hand-side must be either `.` or the sum of the names of numeric variables in `data`. Non-syntactic names must be in quotes (single ' or double "), but not backticks `` ` ``. The `.` on the right-hand-side is expanded to the formula

containing the sum of all remaining (after the response and the conditioning variables) numeric columns in `data`. An empty left-hand-side is interpreted as the `rownames(data)`. See the examples for all possible forms of formula recognized by the `likert` function.

Otherwise, any numeric object stored as a vector, matrix, array, data.frame, table, ftable, structable (as defined in the `vcd` package), or as a list of named two-dimensional objects. This is the only required argument. See the Details section for restrictions on the form of data.frame, list, ftable, and structable arguments.

`data` For the formula method, a `data.frame`. Do not use variable names `".value"` or `".variable"`.

`ReferenceZero` Numeric scalar or `NULL`. The position in the range `seq(0, attr(x, "nlevels")+.5, .5)` where the reference line at 0 will be placed. `attr(x, "nlevels")` is the number of columns of the original argument `x`, *before* it has been coerced to a `"likert"` object. The default `NULL` corresponds to the middle level if there are an odd number of levels, and to halfway between the two middle levels if there are an even number of levels. This argument is used when the number of positive levels and the number of negative levels are not the same. For example, with 4 levels `c("Disagree", "Neutral", "Weak Agree", "Strong Agree")`, the argument would be specified `ReferenceZero=2` indicating that the graphical split would be in the middle of the second group with label `"Neutral"`.

`value` Name of the numeric variable containing the data when the formula method is used with the long data form. The predictor in the formula will be a factor name. The name of the predictor will be used as the title in the key.

`levelsName` (optional) Name of the implied factor distinguishing the columns of the response variables when the formula method is used with the wide data form. This name will be used as the title in the key.

`positive.order` If `FALSE`, the default value, the original order of the rows is retained. This is necessary for arrays, because each panel has the same rownames. If `TRUE`, rows are ordered within each panel with the row whose bar goes farthest to the right at the top of a panel of horizontal bars or at the left of a panel of vertical bars. `positive.order` is frequently set to `TRUE` for lists.

`data.order` `formula` method only. If `positive.order` is `TRUE`, this `data.order` variable is ignored. If `FALSE`, the default value, and the rows are specified by a factor, then they are ordered by their levels. If `TRUE`, then the rows are ordered by their order in the input `data.frame`.

`as.percent` When `as.percent==TRUE` or `as.percent=="noRightAxis"`, then the values in each row are rescaled to row percents. When `as.percent==TRUE` the original row totals are used as `rightAxisLabels`, `rightAxis` is set to `TRUE`, the `ylab.right` is by default set to `"Row Count Totals"` (the user can change its value in the calling sequence). When `as.percent=="noRightAxis"`, then `rightAxis` will be set to `FALSE`.

`as.table` Standard `lattice` argument. See `barchart`.

`par.settings.in, scales.in, auto.key.in, panel.in`

These are placeholders for `lattice` arguments that lets the user specify some lattice `par.settings` and still retain the ones that are prespecified in the `plot.likert.default`.

`ylab, xlab, ylab.right, xlab.top, main`
: Standard `lattice` graph labels in `barchart`.

`between`
: Standard `lattice` argument.

`col`
: Vector of color names for the levels of the agreement factor. Although the colors can be specified as an arbitrary vector of color names, for example, `col=c('red','blue','#4AB3F2')`, usually specifying one of the diverging palettes from `diverge_hcl` or sequential palettes from `sequential_hcl` will suffice. For less intense colors, you can use the middle colors from a larger set of colors; e.g., `col=sequential_hcl(11)[5:2]`. See the last `AudiencePercent` example below for this usage.

`colorFunction, colorFunctionOption`
: See `likertColor`.

`reference.line.col`
: Color for reference line at zero.

`col.strip.background`
: Background color for the strip labels.

`key.border.white`
: Logical. If `TRUE`, then place a white border around the `rect` in the key, else use the `col` of the rect itself.

`horizontal`
: Logical, with default `TRUE` indicating horizontal bars, will be passed to the `barchart` function by the `plot.likert` method. In addition, it interchanges the meaning of `resize.height` and `resize.width` arguments to the `likert` functions applied to arrays and lists.

`...`
: other arguments. These will be passed to the `barchart` function by the `plot.likert` method. Arguments to the `lattice` `auto.key=list()` argument (described in `barchart`) will be used in the legend. See the examples.

`strip.left, strip`
: Logical. The default `strip.left=TRUE` places the strip labels on the left of each panel as in the first professional challenges example. The alternative `strip.left=FALSE` puts the strip labels on the top of each panel, the traditional lattice strip label position.

`condlevelsName, strip.left.values, strip.values, strip.par, strip.left.par, layout`
: Arguments which will be passed to `ResizeEtc`.

`xName`
: Name of the argument in its original environment.

`rightAxis`
: logical. Should right axis values be displayed? Defaults to `FALSE` unless `rightAxisLabels` are specified.

`rightAxisLabels`
: Values to be displayed on the right axis. The default values are the row totals. These are sensible for tables of counts. When the data is rescaled to percents by the `as.percent=TRUE` argument, then the `rightAxisLabels` are still defaulted to the row totals for the counts. We illustrate this usage in the `ProfChal` example.

`resize.height.tuning`
: Tuning parameter used to adjust the space between bars as specified by the `resize.height` argument to the `ResizeEtc` function.

h.resizePanels, resize.height
: Either character scalar or numeric vector. If "nrow", then the panels heights are proportional to the number of bars in each panel. If "rowSums" and there is exactly one bar per panel, then the panels heights are proportional to the total count in each bar, and see the discussion of the box.ratio argument. If a numeric vector, the panel heights are proportional to the numbers in the argument.

w.resizePanels, resize.width
: Numeric vector. The panel widths are proportional to the numbers in the argument.

box.ratio
: If there are more than one bar in any panel, then this defaults to the trellis standard value of 2. If there is exactly one bar in a panel, then the value is 1000, with the intent to minimize the white space in the panel. In this way, when as.percent==TRUE, the bar total area is the count and the bar widths are all equal at 100%. See the example below.

panel
: panel function eventually to be used by barchart.

xscale.components, yscale.components
: See yscale.components.default. xscale.components.top.HH constructs the top x-axis labels, when needed, as the names of the bottom x-axis labels. yscale.components.right.HH constructs the right y-axis labels, when needed, as the names of the left y-axis labels. The names are placed automatically by the plot.likert methods based on the value of the arguments as.percent, rightAxis, and rightAxisLabels. By default, when rightAxis != FALSE the layout.widths are set to list(ylab.right=5, right.padding=0). Otherwise, those arguments are left at their default values. They may be adjusted with an argument of the form par.settings.in= list(layout.widths=list(ylab.right=5, right.padding=0)). Similarly, spacing for the top labels can be adjusted with an argument of the form par.settings.in=list(layout.heights=list(key.axis.padding=6)).

xlimEqualLeftRight
: Logical. The default is FALSE. If TRUE and at and labels are not explicitly specified, then the left and right x limits are set to negative and positive of the larger of the absolute value of the original x limits. When !horizontal, this argument applies to the y coordinate.

xTickLabelsPositive
: Logical. The default is TRUE. If TRUE and at and labels are not explicitly specified, then the tick labels on the negative side are displayed as positive values. When !horizontal, this argument applies to the y coordinate.

reverse
: Logical. The default is FALSE. If TRUE, the rows of the input matrix are reversed. The default is to plot the rows from top-to-bottom for horizontal bars and from left-to-write for vertical bars. reverse, positive.order, and horizontal are independent. All eight combinations are possible. See the Eight sequences and orientations section in the example for all eight.

Details

The counts (or percentages) of respondents on each row who agree with the statement are shown to the right of the zero line; the counts (or percentages) who disagree are shown to the left. The

counts (or percentages) for respondents who neither agree nor disagree are split down the middle and are shown in a neutral color. The neutral category is omitted when the scale has an even number of choices. It is difficult to compare lengths without a common baseline. In this situation, we are primarily interested in the total count (or percent) to the right or left of the zero line; the breakdown into strongly or not is of lesser interest so that the primary comparisons do have a common baseline of zero. The rows within each panel are displayed in their original order by default. If the argument `positive.order=TRUE` is specified, the rows are ordered by the counts (or percentages) who agree.

Diverging stacked barcharts are also called "two-directional stacked barcharts". Some authors use the term "floating barcharts" for vertical diverging stacked barcharts and the term "sliding barcharts" for horizontal diverging stacked barcharts.

All items in a list of named two-dimensional objects must have the same number of columns. If the items have different column names, the column names of the last item in the list will be used in the key. If the dimnames of the matrices are named, the names will be used in the plot. It is possible to produce a likert plot with a list of objects with different numbers of columns, but not with the `plot.likert.list` method. These must be done manually by using the `ResizeEtc` function on each of the individual likert plots. The difficulty is that the legend is based on the last item in the list and will have the wrong number of values for some of the panels.

A single data.frame `x` will be plotted as `data.matrix(x[sapply(x, is.numeric)])`. The subscripting on the class of the columns is there to remove columns of characters (which would otherwise be coerced to NA) and factor columns (which would otherwise be coerced to integers). A data.frame with only numeric columns will work in a named list. A list of data.frame with factors or characters will be plotted by automatically removing columns that are not numeric.

`ftable` and `structable` arguments `x` will be plotted as `as.table(x)`. This changes the display sequence. Therefore the user will probably want to use `aperm` on the `ftable` or `structable` before using `plot.likert`.

The `likert` method is designed for use with `"likert"` objects created with the independent **likert** package. It is not recommended that the **HH** package and the `likert` package both be loaded at the same time, as they have incompatible usage of the exported function names `likert` and `plot.likert`. If the **likert** package is installed, it can be run without loading by using the function calls `likert::likert()` and `likert:::plot.likert()`.

Value

A `"trellis"` object containing the plot. The plot will be automatically displayed unless the result is assigned to an object.

Note

The current version of the `likert` function uses the default diverging palette from `diverge_hcl` as the default. Previous versions used the `RColorBrewer` palette `"RdBu"` as the default color palette. The previous color palette is still available with an explicit call to `likertColorBrewer`, for example

```
col=likertColorBrewer(nc, ReferenceZero=ReferenceZero,
BrewerPaletteName="RdBu", middle.color="gray90")
```

Note

Documentation note: Most of the plots drawn by `plot.likert` have a long left-axis tick label. They therefore require a wider window than R's default of a nominal 7in $\times$ 7in window. The

comments with the examples suggest aesthetic window sizes.

Technical note: There are three (almost) equivalent calling sequences for likert plots.

1. `likert(x)  ## recommended`
   `likert` is an alias for `plot.likert()`.
2. `plot.likert(x)`
   `plot.likert` is both a method of `plot` for `"likert"` objects, and a generic function in its own right. There are methods of `plot.likert` for `"formula"`, `"matrix"`, `"array"`, `"table"`, and several other classes of input objects.
3. `plot(as.likert(x))`
   Both `likert` and `plot.likert` work by calling the `as.likert` function on their argument `x`. Once `as.likert` has converted its argument to a `"likert"` object, the method dispatch technology for the generic `plot.likert` is in play. The user can make the explicit call `as.likert(x)` to see what a `"likert"` object looks like, but is very unlikely to want to look a second time.

Author(s)

Richard M. Heiberger, with contributions from Naomi B. Robbins <naomi@nbr-graphs.com>.

Maintainer: Richard M. Heiberger <rmh@temple.edu>

References

Richard M. Heiberger, Naomi B. Robbins (2014)., "Design of Diverging Stacked Bar Charts for Likert Scales and Other Applications", Journal of Statistical Software, 57(5), 1–32, http://www.jstatsoft.org/v57/i05/.

Richard Heiberger and Naomi Robbins (2011), "Alternative to Charles Blow's Figure in \"Newt's War on Poor Children\"", Forbes OnLine, December 20, 2011. http://www.forbes.com/sites/naomirobbins/2011/12/20/alternative-to-charles-blows-figure-in-newts-war-on-poor-children-2/

Naomi Robbins (2011), "Visualizing Data: Challenges to Presentation of Quality Graphics—and Solutions", Amstat News, September 2011, 28–30. http://magazine.amstat.org/blog/2011/09/01/visualizingdata/

Naomi B. Robbins and Richard M. Heiberger (2011). Plotting Likert and Other Rating Scales. In JSM Proceedings, Section on Survey Research Methods. Alexandria, VA: American Statistical Association, 1058–1066.
https://www.amstat.org/membersonly/proceedings/2011/papers/300784_64164.pdf

Luo, Amy and Tim Keyes (2005). "Second Set of Results in from the Career Track Member Survey," Amstat News. Arlington, VA: American Statistical Association.

See Also

`barchart`, `ResizeEtc`, `as.likert`, `as.matrix.listOfNamedMatrices`

Examples

```
## See file HH/demo/likert-paper.r for a complete set of examples using
## the formula method into the underlying lattice:::barchart plotting
```

```
## technology.  See file HH/demo/likert-paper-noFormula.r for the same
## set of examples using the matrix and list of matrices methods.  See
## file HH/demo/likertMosaic-paper.r for the same set of examples using
## the still experimental functions built on the vcd:::mosaic as the
## underlying plotting technology

require(grid)
require(lattice)
require(latticeExtra)
require(HH)

data(ProfChal)  ## ProfChal is a data.frame.
## See below for discussion of the dataset.


## Count plot
likert(Question ~ . , ProfChal[ProfChal$Subtable=="Employment sector",],
       main='Is your job professionally challenging?',
       ylab=NULL,
       sub="This plot looks better in a 9in x 4in window.")

## Percent plot calculated automatically from Count data
likert(Question ~ . , ProfChal[ProfChal$Subtable=="Employment sector",],
       as.percent=TRUE,
       main='Is your job professionally challenging?',
       ylab=NULL,
       sub="This plot looks better in a 9in x 4in window.")

## formula method
data(NZScienceTeaching)
likert(Question ~ . | Subtable, data=NZScienceTeaching,
       ylab=NULL,
       scales=list(y=list(relation="free")), layout=c(1,2))

## Not run:
## formula notation with expanded right-hand-side
likert(Question ~
       "Strongly disagree" + Disagree + Neutral + Agree + "Strongly agree" |
       Subtable, data=NZScienceTeaching,
       ylab=NULL,
       scales=list(y=list(relation="free")), layout=c(1,2))

## End(Not run)

## Not run:
## formula notation with long data arrangement
NZScienceTeachingLong <- melt(NZScienceTeaching, id.vars=c("Question", "Subtable"))
names(NZScienceTeachingLong)[3] <- "Agreement"
head(NZScienceTeachingLong)

likert(Question ~ Agreement | Subtable, value="value", data=NZScienceTeachingLong,
       ylab=NULL,
       scales=list(y=list(relation="free")), layout=c(1,2))
```

```
## End(Not run)

## Examples with higher-dimensional arrays.
tmp3 <- array(1:24, dim=c(2,3,4),
              dimnames=list(A=letters[1:2], B=LETTERS[3:5], C=letters[6:9]))

## positive.order=FALSE is the default.  With arrays
## the rownames within each item of an array are identical.

## likert(tmp3)
likert(tmp3, layout=c(1,4))
likert(tmp3, layout=c(2,2), resize.height=c(2,1), resize.width=c(3,4))


## plot.likert interprets vectors as single-row matrices.
## http://survey.cvent.com/blog/customer-insights-2/box-scores-are-not-just-for-baseball
Responses <- c(15, 13, 12, 25, 35)
names(Responses) <- c("Strongly Disagree", "Disagree", "No Opinion",
                      "Agree", "Strongly Agree")
## Not run:
likert(Responses, main="Retail-R-Us offers the best everyday prices.",
       sub="This plot looks better in a 9in x 2.6in window.")

## End(Not run)
## reverse=TRUE  is needed for a single-column key with
## horizontal=FALSE and with space="right"
likert(Responses, horizontal=FALSE,
       aspect=1.5,
       main="Retail-R-Us offers the best everyday prices.",
       auto.key=list(space="right", columns=1,
                     reverse=TRUE, padding.text=2),
       sub="This plot looks better in a 4in x 3in window.")


## Not run:
## Since age is always positive and increases in a single direction,
## this example uses colors from a sequential palette for the age
## groups.  In this example we do not use a diverging palette that is
## appropriate when groups are defined by a characteristic, such as
## strength of agreement or disagreement, that can increase in two directions.

## Initially we use the default Blue palette in the sequential_hcl function.
likert(AudiencePercent,
       auto.key=list(between=1, between.columns=2),
       xlab=paste("Percentage of audience younger than 35 (left of zero)",
                  "and older than 35 (right of zero)"),
       main="Target Audience",
       col=rev(sequential_hcl(4)),
       sub="This plot looks better in a 7in x 3.5in window.")

## The really light colors in the previous example are too light.
## Therefore we use the col argument directly.  We chose to use an
```

```
## intermediate set of Blue colors selected from a longer Blue palette.
likert(AudiencePercent,
       positive.order=TRUE,
       auto.key=list(between=1, between.columns=2),
       xlab=paste("Percentage of audience younger than 35",
         "(left of zero) and older than 35 (right of zero)"),
       main="Brand A has the most even distribution of ages",
       col=sequential_hcl(11)[5:2],
       scales=list(x=list(at=seq(-90,60,10),
                     labels=as.vector(rbind("",seq(-80,60,20))))),
      sub="This plot looks better in a 7in x 3.5in window.")

## End(Not run)


## Not run:
## See the ?as.pyramidLikert help page for these examples
## Population Pyramid
data(USAge.table)
USA79 <- USAge.table[75:1, 2:1, "1979"]/1000000
PL <- likert(USA79,
             main="Population of United States 1979 (ages 0-74)",
             xlab="Count in Millions",
             ylab="Age",
             scales=list(
               y=list(
                 limits=c(0,77),
                 at=seq(1,76,5),
                 labels=seq(0,75,5),
                 tck=.5))
             )
PL
as.pyramidLikert(PL)

likert(USAge.table[75:1, 2:1, c("1939","1959","1979")]/1000000,
       main="Population of United States 1939,1959,1979 (ages 0-74)",
       sub="Look for the Baby Boom",
       xlab="Count in Millions",
       ylab="Age",
       scales=list(
         y=list(
           limits=c(0,77),
           at=seq(1,76,5),
           labels=seq(0,75,5),
           tck=.5)),
       strip.left=FALSE, strip=TRUE,
       layout=c(3,1), between=list(x=.5))

## End(Not run)


Pop <- rbind(a=c(3,2,4,9), b=c(6,10,12,10))
dimnames(Pop)[[2]] <- c("Very Low", "Low", "High", "Very High")
```

```
likert(as.listOfNamedMatrices(Pop),
            as.percent=TRUE,
            resize.height="rowSums",
            strip=FALSE,
            strip.left=FALSE,
            main=paste("Area and Height are proportional to 'Row Count Totals'.",
                       "Width is exactly 100%.", sep="\n"))


## Professional Challenges example.
##
## The data for this example is a list of related likert scales, with
## each item in the list consisting of differently named rows.  The data
## is from a questionnaire analyzed in a recent Amstat News article.
## The study population was partitioned in several ways.  Data from one
## of the partitions (Employment sector) was used in the first example
## in this help file.  The examples here show various options for
## displaying all partitions on the same plot.
##
data(ProfChal)
levels(ProfChal$Subtable)[6] <- "Prof Recog" ## reduce length of label

## 1. Plot counts with rows in each panel sorted by positive counts.
##
## Not run:
likert(Question ~ . | Subtable, ProfChal,
       positive.order=TRUE,
       main="This works, but needs more specified arguments to look good")

likert(Question ~ . | Subtable, ProfChal,
       scales=list(y=list(relation="free")), layout=c(1,6),
       positive.order=TRUE,
       between=list(y=0),
       strip=FALSE, strip.left=strip.custom(bg="gray97"),
       par.strip.text=list(cex=.6, lines=5),
       main="Is your job professionally challenging?",
       ylab=NULL,
       sub="This looks better in a 10inx7in window")

## End(Not run)

ProfChalCountsPlot <-
likert(Question ~ . | Subtable, ProfChal,
       scales=list(y=list(relation="free")), layout=c(1,6),
       positive.order=TRUE,
       box.width=unit(.4,"cm"),
       between=list(y=0),
       strip=FALSE, strip.left=strip.custom(bg="gray97"),
       par.strip.text=list(cex=.6, lines=5),
       main="Is your job professionally challenging?",
       rightAxis=TRUE,  ## display Row Count Totals
       ylab=NULL,
       sub="This looks better in a 10inx7in window")
```

```
ProfChalCountsPlot


## Not run:
## 2. Plot percents with rows in each panel sorted by positive percents.
##    This is a different sequence than the counts.  Row Count Totals are
##    displayed on the right axis.
ProfChalPctPlot <-
likert(Question ~ . | Subtable, ProfChal,
       as.percent=TRUE,    ## implies display Row Count Totals
       scales=list(y=list(relation="free")), layout=c(1,6),
       positive.order=TRUE,
       box.width=unit(.4,"cm"),
       between=list(y=0),
       strip=FALSE, strip.left=strip.custom(bg="gray97"),
       par.strip.text=list(cex=.6, lines=5),
       main="Is your job professionally challenging?",
       rightAxis=TRUE,  ## display Row Count Totals
       ylab=NULL,
       sub="This looks better in a 10inx7in window")
ProfChalPctPlot

## 3. Putting both percents and counts on the same plot, both in
##    the order of the positive percents.

LikertPercentCountColumns(Question ~ . | Subtable, ProfChal,
                          layout=c(1,6), scales=list(y=list(relation="free")),
                          ylab=NULL, between=list(y=0),
                          strip.left=strip.custom(bg="gray97"), strip=FALSE,
                          par.strip.text=list(cex=.7),
                          positive.order=TRUE,
                          main="Is your job professionally challenging?")


## Restore original name
## levels(ProfChal$Subtable)[6] <- "Attitude\ntoward\nProfessional\nRecognition"

## End(Not run)

## Not run:
## 4. All possible forms of formula for the likert formula method:
data(ProfChal)
row.names(ProfChal) <- abbreviate(ProfChal$Question, 8)

likert( Question ~ . | Subtable,
       data=ProfChal, scales=list(y=list(relation="free")), layout=c(1,6))

likert( Question ~
        "Strongly Disagree" + Disagree + "No Opinion" + Agree + "Strongly Agree" | Subtable,
       data=ProfChal, scales=list(y=list(relation="free")), layout=c(1,6))

likert( Question ~ . ,
       data=ProfChal)
```

```
likert( Question ~ "Strongly Disagree" + Disagree + "No Opinion" + Agree + "Strongly Agree",
       data=ProfChal)

likert( ~ . | Subtable,
       data=ProfChal, scales=list(y=list(relation="free")), layout=c(1,6))

likert( ~ "Strongly Disagree" + Disagree + "No Opinion" + Agree + "Strongly Agree" | Subtable,
       data=ProfChal, scales=list(y=list(relation="free")), layout=c(1,6))

likert( ~ . ,
       data=ProfChal)

likert( ~ "Strongly Disagree" + Disagree + "No Opinion" + Agree + "Strongly Agree",
       data=ProfChal)


## End(Not run)

## Not run:
## 5. putting the x-axis tick labels on top for horizontal plots
##    putting the y-axis tick lables on right for vertical plots
##
## This non-standard specification is a consequence of using the right
## axis labels for different values than appear on the left axis labels
## with horizontal plots, and using the top axis labels for different
## values than appear on the bottom axis labels with vertical plots.

## Percent plot calculated automatically from Count data

tmph <-
likert(Question ~ . , ProfChal[ProfChal$Subtable=="Employment sector",],
       as.percent=TRUE,
       main='Is your job professionally challenging?',
       ylab=NULL,
       sub="This plot looks better in a 9in x 4in window.")
tmph$x.scales$labels
names(tmph$x.scales$labels) <- tmph$x.scales$labels
update(tmph, scales=list(x=list(alternating=2)), xlab=NULL, xlab.top="Percent")

tmpv <-
likert(Question ~ . , ProfChal[ProfChal$Subtable=="Employment sector",],
       as.percent=TRUE,
       main='Is your job professionally challenging?',
       sub="likert plots with long Question names look better horizontally.
With effort they can be made to look adequate vertically.",
       horizontal=FALSE,
       scales=list(y=list(alternating=2), x=list(rot=c(90, 0))),
       ylab.right="Percent",
       ylab=NULL,
       xlab.top="Column Count Totals",
       par.settings=list(
         layout.heights=list(key.axis.padding=5),
```

```
         layout.widths=list(key.right=1.5, right.padding=0))
)
tmpv$y.scales$labels
names(tmpv$y.scales$labels) <- tmpv$y.scales$labels
tmpv
tmpv$x.limits <- abbreviate(tmpv$x.limits,8)
tmpv$x.scales$rot=c(0, 0)
tmpv


## End(Not run)

## The ProfChal data is done again with explicit use of ResizeEtc
## in ?HH:::ResizeEtc
```

`likertColor` *Selection of colors for Likert plots.*

Description

Selection of colors for Likert plots.

Usage

```
ColorSet(nc, ReferenceZero=NULL)
likertColor(nc, ReferenceZero=NULL,
                colorFunction=c("diverge_hcl","sequential_hcl"),
                colorFunctionOption=c("lighter","flatter","default"),
                colorFunctionArgs=
                 likertColorFunctionArgs[[colorFunctionOption, colorFunction]],
                ...)
likertColorBrewer(nc, ReferenceZero=NULL,
                  BrewerPaletteName="RdBu", middle.color="gray90")

brewer.pal.likert(n, name, middle.color)
```

Arguments

`n, nc`	Number of colors in the palette. If there are more levels than `RColorBrewer` normally handles, we automatically interpolate with `colorRampPalette`.
`ReferenceZero`	Numeric scalar or `NULL`. The position in the range `seq(0, attr(x, "nlevels")+.5, .5)` where the reference line at 0 will be placed. `attr(x, "nlevels")` is the number of columns of the original argument `x`, *before* it has been coerced to a `"likert"` object. The default `NULL` corresponds to the middle level if there are an odd number of levels, and to halfway between the two middle levels if there are an even number of levels. This argument is used when the number of positive levels and the number of negative

levels are not the same. For example, with 4 levels `c("Disagee", "Neutral", "Weak Agree", "Strong Agree")`, the argument would be specified `ReferenceZero=2` indicating that the graphical split would be in the middle of the second group with label `"Neutral"`.

`colorFunction` Function name from the **colorspace** package, either `"diverge_hcl"` or `"sequential_hcl"`.

`colorFunctionOption`
Name of a list item defined inside the `likertColor` function. The item contains a list of parameters to the function identified in the `colorFunction` argument.

`colorFunctionArgs`
list of arguments to the **colorspace** function. The default selects the values by indexing into a list defined in the `likertColor` function using the values of the two arguments `colorFunction` and `colorFunctionOption`. For non-default usage, see the BlueOrange example in this help page.

`...` Other arguments are ignored.

`BrewerPaletteName, name`
RColorBrewer palette names. We default to the diverging palette `RdBu`. Diverging palettes are usually appropriate for two-directional scales (Agree–Disagree). Sequential palettes are often appropriate for one-directional scales (Age Ranges). Qualitative palettes are usually not appropriate for likert plots.

`middle.color` Darker middle color than the default `"#F7F7F7"` in the `RdBu` scheme.

Details

These are support functions for the `plot.likert` function. Please see `plot.likert` for details.

`likertColor` uses by default the `diverge_hcl` diverging palette defined by the argument `colorFunctionOption="lighter"`.

`likertColorBrewer` by default uses the `"RdBu"` diverging palette from RColorBrewer.

Value

`ColorSet` returns a vector of integers, one per each level, corresponding to the strength of the levels from Disagree to Agree. For balanced levels, such as `c("Disagree Strongly", "Disagree Weakly", "Agree Weakly", "Agree Strongly")`, corresponding to `nc=4, ReferenceZero=2.5`, it returns `-2 -1 1 2`. For unbalanced levels, such as `c("Disagree", "Neutral", "Agree Weakly", "Agree Strongly")`, corresponding to `nc=4, ReferenceZero=2`, it returns `-1 0 1 2`.

`likertColor` returns a subset of a palette constructed by either `diverge_hcl` or `sequential_hcl` in the **colorspace** package. The subset corresponds to the levels specified by `ColorSet`.

`brewer.pal.likert` returns a RColorBrewer palette.

`likertColorBrewer` returns a subset of a palette constructed by `brewer.pal.likert`. The subset corresponds to the levels specified by `ColorSet`.

Author(s)

Richard M. Heiberger, with contributions from Naomi B. Robbins <naomi@nbr-graphs.com>.

Maintainer: Richard M. Heiberger <rmh@temple.edu>

See Also

plot.likert

Examples

```
brewer.pal.likert(4, "RdBu")
brewer.pal.likert(5, "RdBu")
ColorSet(4)
ColorSet(4, 2)
likertColor(4)
likertColor(4, 2.5) ## same as above
likertColor(4, 2)   ## one negative level and two positive levels: default
likertColor(5, 3)[-2] ## one negative level and two positive levels: stronger negative

## Not run:
  ## Examples illustrating the six predefined likertColor palettes, and how
  ## to define additional hcl color palettes for use with the likert functions.

  data(ProfDiv)
  ProfDiv.df <- data.frame(ProfDiv)

  likert( ~ . , ProfDiv.df, horizontal=FALSE, positive.order=FALSE)
  likert( ~ . , ProfDiv.df, horizontal=FALSE, positive.order=FALSE,
         colorFunctionOption="default")
  likert( ~ . , ProfDiv.df, horizontal=FALSE, positive.order=FALSE,
         colorFunctionOption="flatter")
  likert( ~ . , ProfDiv.df, horizontal=FALSE, positive.order=FALSE,
         colorFunction="sequential_hcl")
  likert( ~ . , ProfDiv.df, horizontal=FALSE, positive.order=FALSE,
         colorFunction="sequential_hcl", colorFunctionOption="default")
  likert( ~ . , ProfDiv.df, horizontal=FALSE, positive.order=FALSE,
         colorFunction="sequential_hcl", colorFunctionOption="flatter")

  likert(ProfDiv, horizontal=FALSE, positive.order=FALSE)
  likert(ProfDiv, horizontal=FALSE, positive.order=FALSE,
         colorFunctionOption="default")
  likert(ProfDiv, horizontal=FALSE, positive.order=FALSE,
         colorFunctionOption="flatter")
  likert(ProfDiv, horizontal=FALSE, positive.order=FALSE,
         colorFunction="sequential_hcl")
  likert(ProfDiv, horizontal=FALSE, positive.order=FALSE,
         colorFunction="sequential_hcl", colorFunctionOption="default")
  likert(ProfDiv, horizontal=FALSE, positive.order=FALSE,
         colorFunction="sequential_hcl", colorFunctionOption="flatter")

  likertMosaic(ProfDiv.df)
  likertMosaic(ProfDiv.df, colorFunctionOption="default")
  likertMosaic(ProfDiv.df, colorFunctionOption="flatter")
  likertMosaic(ProfDiv.df, colorFunction="sequential_hcl")
  likertMosaic(ProfDiv.df, colorFunction="sequential_hcl",
               colorFunctionOption="default")
  likertMosaic(ProfDiv.df, colorFunction="sequential_hcl",
```

```
              colorFunctionOption="flatter")

  ## specify an hcl palette for use with the likert functions.
  BlueOrange <- likertColor(nc=4, ReferenceZero=NULL,
                            colorFunction="diverge_hcl",
                            colorFunctionArgs=
                              list(h=c(246, 40), c=96, l=c(65,90), power=1.5))
  likert( ~ . , ProfDiv.df, horizontal=FALSE, positive.order=FALSE, col=BlueOrange)

## End(Not run)
```

---

likertMosaic *Diverging stacked barcharts for Likert, semantic differential, rating scale data, and population pyramids based on mosaic as the plotting style.*

Description

Constructs and plots diverging stacked barcharts for Likert, semantic differential, rating scale data, and population pyramids, .based on `mosaic` as the plotting style.

Usage

```
likertMosaic(x, ...)

## S3 method for class 'formula'
likertMosaic(x, data, ReferenceZero = NULL, spacing=NULL,
                   ..., between.y = c(1.2, 0.3))

## S3 method for class 'array'
likertMosaic(x, ReferenceZero = NULL, col = NULL, main = NULL,
             ...,
             as.percent = FALSE, variable.width = NULL, positive.order = FALSE,
             Conditions = NULL,
             x.legend = list(text = list(dimnames(x)[[ndim]]),
                             columns = x.dim[ndim],
                             space = "bottom",
                             size = 2,
                             cex = 0.8,
                             between = 0.6,
                             rect= list(col = col, border = "white")),
             legend.y = 0.05,
             spacing = spacing_highlighting,
             split_vertical = c(TRUE, FALSE),
             margins = c(3, 2, 4, 22),
             keep_aspect = FALSE,
             rot_labels = c(0, 0, 90, 0),
             just_labels = c("center", "center", "center", "right"),
```

```
              labels = c(TRUE, TRUE, FALSE, TRUE),
              varnames = FALSE,
              zero_size = 0,
              gp = gpar(fill = col.extended, col = 0),
              colorFunction="diverge_hcl",
              colorFunctionOption="lighter")

## S3 method for class 'data.frame'
likertMosaic(x, ...)

## Default S3 method:
likertMosaic(x, ...) ## most likely for a vector

## S3 method for class 'list'
likertMosaic(x, ...)

## S3 method for class 'matrix'
likertMosaic(x, ...,
 split_vertical = c(FALSE, TRUE),
 rot_labels = c(90, 0, 0, 0),
 just_labels = c("left", "center", "center", "right"),
 labels = c(TRUE, FALSE))
```

Arguments

x	For the formula method, a model formula. Otherwise, any numeric object stored as a vector, matrix, array, data.frame, table, ftable, structable (as defined in the vcd package), or as a list of named two-dimensional objects. This is the only required argument. See the Details section for restrictions on the form of data.frame, list, ftable, and structable arguments.
data	For the formula method, a `data.frame`.
ReferenceZero	Numeric scalar or NULL. The position in the range `seq(0, attr(x, "nlevels")+.5, .5)` where the reference line at 0 will be placed. `attr(x, "nlevels")` is the number of columns of the original argument x, *before* it has been coerced to a `"likert"` object. The default NULL corresponds to the middle level if there are an odd number of levels, and to half-way between the two middle levels if there are an even number of levels. This argument is used when the number of positive levels and the number of negative levels are not the same. For example, with 4 levels `c("Disagee", "Neutral", "Weak Agree", "Strong Agree")`, the argument would be specified `ReferenceZero=2` indicating that the graphical split would be in the middle of the second group with label `"Neutral"`.
positive.order	If FALSE, the default value, the original order of the rows is retained. This is necessary for arrays, because each panel has the same rownames. If TRUE, rows are ordered within each panel with the row whose bar goes farthest to the right at the top of a panel of horizontal bars or at the left of a panel of vertical bars. `positive.order` is frequently set to TRUE for lists.

as.percent When as.percent==TRUE or as.percent=="noRightAxis", then the values in each row are rescaled to row percents.

variable.width When TRUE and as.percent==TRUE, then the area of the bars (percent along the length times the width) is proportional to the counts.

col Colors for the bars. With the default value NULL, the colors are chosen from the default diverge_hcl diverging palette. Any color specification that R understands can be used here.

colorFunction, colorFunctionOption
See likertColor.

main main title for the plot.

... Additional arguments, passed to the next method and possibly all the way to strucplot.

Conditions Factor used to divide the rows of the plot into sets of rows corresponding to levels of Condition. In the formula method, the conditions are the factors appearing after the | symbol.

between.y vertical spacing between bars. between.y[1] is used between levels of conditioning factors, and between.y[2] is used between bars within the same level of the conditioning factor.

x.legend Description of legend using the terminology and conventions of the lattice package.

legend.y Adjust vertical location of legend.

spacing, split_vertical, margins, keep_aspect, rot_labels, just_labels, labels
Please see strucplot for details.

varnames, zero_size, gp
Please see strucplot for details.

Details

The counts (or percentages) of respondents on each row who agree with the statement are shown to the right of the zero line; the counts (or percentages) who disagree are shown to the left. The counts (or percentages) for respondents who neither agree nor disagree are split down the middle and are shown in a neutral color. The neutral category is omitted when the scale has an even number of choices. It is difficult to compare lengths without a common baseline. In this situation, we are primarily interested in the total count (or percent) to the right or left of the zero line; the breakdown into strongly or not is of lesser interest so that the primary comparisons do have a common baseline of zero. The rows within each panel are displayed in their original order by default. If the argument positive.order=TRUE is specified, the rows are ordered by the counts (or percentages) who agree.

Diverging stacked barcharts are also called "two-directional stacked barcharts". Some authors use the term "floating barcharts" for vertical diverging stacked barcharts and the term "sliding barcharts" for horizontal diverging stacked barcharts.

All items in a list of named two-dimensional objects must have the same number of columns. If the items have different column names, the column names of the last item in the list will be used in the key. If the dimnames of the matrices are named, the names will be used in the plot. It is possible to produce a likert plot with a list of objects with different numbers of columns, but not with the plot.likert.list method. These must be done manually by using the ResizeEtc function on

each of the individual likert plots. The difficulty is that the legend is based on the last item in the list and will have the wrong number of values for some of the panels.

A single data.frame `x` will be plotted as `data.matrix(x)`; therefore factor columns will be converted to integers and character columns will become `NA` and will be plotted as if they had value 0. A data.frame with only numeric columns will work in a named list. A data.frame with factors or characters won't work in a named list.

ftable and structable arguments `x` will be plotted as `as.table(x)`. This changes the display sequence. Therefore the user will probably want to use `aperm` on the ftable or structable before using `plot.likert`.

Value

Please see `strucplot` for a description of the returned object.

Note

The functions described here are currently missing the following features:

1. no axis ticks, number, nor axis label for the x axis
2. no zero reference line
3. no right-axis labels for Row Count Totals
4. no strip.left labels for grouping by Conditions
5. In Figure 8 and 9 (`HH/demo/likertMosaic-paper.r`), no control of the thickness of the bars
6. All bars are horizontal.
7. No borders on the overall plot nor on the panels in plots with grouping by Conditions
8. No control of `between=list(x=number)`
9. `cex` for labeling
10. border on empty boxes
11. I am using a lattice legend, not a native strucplot legend

Author(s)

Richard M. Heiberger, with contributions from Naomi B. Robbins <naomi@nbr-graphs.com>.

Maintainer: Richard M. Heiberger <rmh@temple.edu>

References

Richard M. Heiberger, Naomi B. Robbins (2014)., "Design of Diverging Stacked Bar Charts for Likert Scales and Other Applications", Journal of Statistical Software, 57(5), 1–32, http://www.jstatsoft.org/v57/i05/.

Richard Heiberger and Naomi Robbins (2011), "Alternative to Charles Blow's Figure in \"Newt's War on Poor Children\"", Forbes OnLine, December 20, 2011. http://www.forbes.com/sites/naomirobbins/2011/12/20/alternative-to-charles-blows-figure-in-newts-war-on-poor-children-2/

Naomi Robbins (2011), "Visualizing Data: Challenges to Presentation of Quality Graphics—and Solutions", Amstat News, September 2011, 28–30. http://magazine.amstat.org/blog/2011/09/01/visualizingdata/

Naomi B. Robbins and Richard M. Heiberger (2011). Plotting Likert and Other Rating Scales. In JSM Proceedings, Section on Survey Research Methods. Alexandria, VA: American Statistical Association, 1058–1066. https://www.amstat.org/membersonly/proceedings/2011/papers/300784_64164.pdf

Luo, Amy and Tim Keyes (2005). "Second Set of Results in from the Career Track Member Survey," Amstat News. Arlington, VA: American Statistical Association.

See Also

likert, mosaic

Examples

```
## See file HH/demo/likertMosaic-paper.r for a complete set of examples.
## Not run:
  require(vcd)
  data(ProfChal)
  likertMosaic(Question ~ . | Subtable, ProfChal,
               main="Is your job professionally challenging?")
  likertMosaic(Question ~ . | Subtable, ProfChal,
               main="Is your job professionally challenging?", as.percent=TRUE)
  likertMosaic(Question ~ . | Subtable, ProfChal,
               main="Is your job professionally challenging?", as.percent=TRUE,
               positive.order=TRUE)
  likertMosaic(Question ~ . | Subtable, ProfChal,
               main="Is your job professionally challenging?", as.percent=TRUE,
               variable.width=TRUE)

  EmpRows <- ProfChal$Subtable == "Employment sector"
  ProfChal2 <- ProfChal[EmpRows, 1:5]
  rownames(ProfChal2) <- substr(ProfChal[EmpRows, "Question"], 1, 5)

  likertMosaic(ProfChal2)
  likertMosaic(ProfChal2, main="Employment")
  likertMosaic(ProfChal2, main="Employment", ReferenceZero=0)
  likertMosaic(ProfChal2, main="Employment", ReferenceZero=3.5)
  likertMosaic(ProfChal2, main="Employment", ReferenceZero=4)
  likertMosaic(ProfChal2, main="Employment", ReferenceZero=6)
  likertMosaic(ProfChal2, main="Employment", positive.order=TRUE)
  likertMosaic(ProfChal2, main="Employment", variable.width=TRUE)

  likertMosaic(~ ., data.frame(ProfChal2), main="Employment", positive.order=TRUE)

  likertMosaic(~ ., data.frame(ProfChal2), main="Employment", variable.width=TRUE)
  likert(~ ., data.frame(ProfChal2), main="Employment", variable.width=TRUE)

  data(SFF8121)
  likertMosaic(aperm(SFF8121, c(3,1,2)))


## End(Not run)
```

LikertPercentCountColumns

Display likert plots with percents in the first column of panels and counts in the second column of panels.

Description

Display likert plots with percents in the first column of panels and counts in the second column of panels. Order the rows either in their original order or by the positive order of the percent display.

Usage

```
LikertPercentCountColumns(
           x, data,
           px=list( ## defaults designed for long QuestionName values
             LL=c(.00,  .50), ## and 7in x 7in window
             LP=c(.50,  .70),
            ML=c(.50,  .51),  ## arbitrary, visually center the labels and legend
             RP=c(.71,  .87),
             RL=c(.87, 1.00)),
           ...,
           QuestionName="Question",
           as.percent="Capture and then ignore this argument",
           positive.order=FALSE)
```

Arguments

x, data, positive.order	formula, data.frame, Logical. See likert.
...	other arguments that can be used for likert.
px	See as.TwoTrellisColumns5.
as.percent	Capture this argument and ignore it. The as.percent argument of likert will be TRUE in the left (Percent) column of the resulting "TwoTrellisColumns5" object and FALSE in the right (Count) column.
QuestionName	Character string containing the name of the column in data containing the values of the response variable.

Value

A "TwoTrellisColumns5" object, consisting of a list containing the constructed left, middle, and right trellis objects, and an attribute containing the px value. See as.TwoTrellisColumns5 for details.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert

Examples

```
library(HH)

## These are based on the Professional Challenges example in ?likert
data(ProfChal)

levels(ProfChal$Subtable)[6] <- "Prof Recog" ## reduce length of label

## See ?print.TwoTrellisColumns for this example using the original ordering

## Order both the plot of the count plot and the percent plot by the
## positive.order of the percent plot.

LikertPercentCountColumns(Question ~ . | Subtable, ProfChal,
                          layout=c(1,6), scales=list(y=list(relation="free")),
                          ylab=NULL, between=list(y=0),
                          strip.left=strip.custom(bg="gray97"), strip=FALSE,
                          par.strip.text=list(cex=.7),
                          positive.order=TRUE,
                          main="Is your job professionally challenging?")
## Not run:
## Retain original order of the Question variable

LikertPercentCountColumns(Question ~ . | Subtable, ProfChal,
                          layout=c(1,6), scales=list(y=list(relation="free")),
                          ylab=NULL, between=list(y=0),
                          strip.left=strip.custom(bg="gray97"), strip=FALSE,
                          par.strip.text=list(cex=.7),
                          main="Is your job professionally challenging?")

## Order both the plot of the count plot and the percent plot by the
## positive.order of the percent plot.
## Just the "Employment sector".
LPCCEs <-
LikertPercentCountColumns(Question ~ . ,
                          ProfChal[ProfChal$Subtable == "Employment sector", -7],
                          ylab=NULL, between=list(y=0),
                          par.strip.text=list(cex=.7),
                          positive.order=TRUE,
                        main="Is your job professionally challenging?\nEmployment sector",
                          px=list( ## defaults designed for long QuestionName values
                            LL=c(.00,  .50), ## and 7in x 7in window
                            LP=c(.49,  .70),
                        ML=c(.50,  .51),  ## arbitrary, visually center the labels and legend
                            RP=c(.71,  .84),
                            RL=c(.87, 1.00)))
LPCCEs$RP$x.scales$at <- c(0,100,200)
LPCCEs$RP$x.scales$labels <- c(0,100,200)
```

```
LPCCEs

## End(Not run)
```

`lm.case` *case statistics for regression analysis*

Description

Case statistics for regression analysis. `case.lm` calculates the statistics. `plot.case` plots the cases, one statistic per panel, and illustrates and itemizes all observations for which the standard thresholds are exceeded. `plot.case` returns a `"trellis"` object containing the plot and also places the row.names of the flagged observations in the variable `.lm.case.large`. `panel.case` is a panel function for `plot.case`.

Usage

```
case(fit, ...)
## S3 method for class 'lm'
case(fit, lms = summary.lm(fit), lmi = lm.influence(fit), ...)

## S3 method for class 'case'
plot(x, fit,
          which=c("stu.res","si","h","cook","dffits",
            dimnames(x)[[2]][-(1:8)]),  ##DFBETAS
          between.in=list(y=4, x=9),
          oma=c(0,0,0,4), cex.threshold=if.R(r=1.2, s=1.6),
          main.in=list(
            paste(deparse(fit$call), collapse=""),
            cex=main.cex),
          sigma.in=summary.lm(fit)$sigma,
          p.in=summary.lm(fit)$df[1]-1,
          obs.large=".lm.case.large",
          obs.large.env=if.R(r=globalenv(), s=0),
          main.cex=NULL,
          ...)

panel.case(x, y, subscripts, rownames, group.names,
           nn, pp, ss, cex.threshold,
           par.settings, ## R only. S-Plus ignores this argument
           obs.large, obs.large.env,
           ...)
```

Arguments

`fit`	`"lm"` object computed with `x=TRUE`
`lms`	`summary.lm(fit)`

lmi	lm.influence(fit)
x	In plot.case, the matrix output from case.lm containing case diagnostics on each observation in the original dataset. In panel.case, the x variable to be plotted
which	In plot.case, the names of the columns of x that are to be graphed.
between.in	between trellis/lattice argument.
oma	In S-Plus, change par()$oma to make room for the threshold values. A warning is printed when par()$oma is changed as the delayed printing of trellis objects implies we can't return it to the original value automatically. In R, this argument is ignored. Instead, we use the par.settings argument to xyplot inside plot.case. The par.settings becomes one component of the "trellis" object that is the value of plot.case and is therefore automatically applied every time the object is printed.
cex.threshold	Multiplier for cex for the threshold values.
main.in	main title for xyplot. The default main title displays the linear model formula from fit.
sigma.in	standard error for the fit.
p.in	The number of degrees of freedom associated with the fitted model.
obs.large	Object name where the names of all observations for which the standard thresholds are exceeded will be stored. The default name is .lm.case.large.
obs.large.env	Frame in S-Plus (defaults to 0) and environment in R (defaults to globalenv()) where obs.large will be stored.
main.cex	cex for main title.
...	other arguments to xyplot
y	the y variable to be plotted.
nn	number of rows in original dataset.
pp	The number of degrees of freedom associated with the fitted model.
ss	standard error for the fit.
subscripts	trellis/lattice argument, position in the reshaped dataset constructed by plot.case before calling xyplot.
rownames	row name in the original data.frame.
group.names	names of the individual statistics.
par.settings	Used in R as part of the call to xyplot. Although this argument is not used in the panel function, it is needed as a formal argument in S-Plus to absorb it out of ... and thereby prevent it from being forwarded to points.

Details

lm.influence is part of S-Plus and R case.lm and plot.case are based on: Section 4.3.3 "Influence of Individual Obervations in Chambers and Hastie", *Statistical Models in S*.

Value

`case.lm` returns a matrix, with one row for each observation in the original dataset. The columns contain the diagnostic statistics: `e` (residuals), `h*` (hat diagonals), `si*` (deleted standard deviation), `sta.res` (standardized residuals), `stu.res*` (Studentized deleted resididuals), `dffit` (difference in fits, change in predicted y when observation i is deleted), `dffits*` (standardized difference in fits, standardized change in predicted y when observation i is deleted), `cook*` (Cook's distance), and DFBETAs* (standardized difference in regression coefficients when observation i is deleted, one for each column of the x-matrix, including the intercept).

`plot.case` returns a `"trellis"` object containing the plot (including the starred columns by default) and also places the row.names of the flagged observations in the variable `.lm.case.large`. The variable `.lm.case.large` is placed by default into frame 0 in S-Plus and into `globalenv()` in R.

`panel.case` is a panel function for `plot.case`. The variable `.lm.case.large` is created one column at a time inside the panel function.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

`lm.influence`.

Examples

```
data(kidney)

kidney2.lm <- lm(clearance ~ concent + age + weight + concent*age,
                 data=kidney,
                 na.action=na.exclude,  ## recommended
                 x=TRUE)  ## the lm object must be computed with x=TRUE

kidney2.case <- case(kidney2.lm)

## this picture looks much better in portrait, specification is device dependent
## trellis.device(postscript, horizontal=TRUE)  ## postscript
## trellis.device(orientation="portrait")       ## S-Plus graphsheet

plot(kidney2.case, kidney2.lm, par.strip.text=list(cex=.9),
     layout=c(2,3))

.lm.case.large ## object placed by the panel.case function
               ## into frame 0 in S-Plus by default
               ## and into globalenv() in R by default
```

lm.regsubsets *Evaluate lm model with highest adjusted R^2.*

Description

The regsubsets function in the leaps package finds the model with the highest adjusted R^2. This function evaluates the full lm object for that model.

Usage

```
lm.regsubsets(object, model.number, ...)
```

Arguments

object	An object of class "regsubsets".
model.number	Index number generated by Rcmdr.
...	Other arguments.

Value

"lm" object for the selected model.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

lm, regsubsets

lmatRows *Find the row numbers in the lmat corresponding to the focus factor.*

Description

lmatRows finds the row numbers in the lmat (column numbers in the linfct in R) corresponding to the focus factor. See mmc for more information. These are internal functions that the user doesn't see. They are necessary when the design has more than one factor. lmatContrast converts user-specified contrasts of levels of a factor to the full lmat or linfct matrix that carries the information about other factors and their interactions and covariates.

Usage

```
lmatRows(x, focus)
## S3 method for class 'mmc.multicomp'
lmatRows(x, focus)
## S3 method for class 'multicomp'
lmatRows(x, focus)
## S3 method for class 'glht'
lmatRows(x, focus) ## R only
## S3 method for class 'lm'
lmatRows(x, focus)
lmatContrast(lmat.none, contrast.matrix)
```

Arguments

x	"lm" or "mmc.multicomp" or "multicomp" or "glht" object.
focus	The name of the term in the ANOVA table for which multiple comparisons are to be constructed.
lmat.none	lmat matrix with the S-Plus multicomp package or t(linfct) matrix with the R multcomp package. In both packages the matrix is the one used for estimating the group means.
contrast.matrix	Matrix of column contrasts for a factor. The columns are the contrasts, the rows are the levels of the factor.

Details

The MMC function are based on glht in R and on multicomp in S-Plus. The two packages have different conventions for specifying the linear contrasts. The lmatRows function gives appropriate values in each system.

Value

For lmatRows, vector of row numbers of the lmat, the matrix of linear contrasts defining the comparisons of interest. For lmatContrast, a linear contrast matrix that follows the conventions of the multiple comparisons package. It has columns for each contrast specified by the input contrast.matrix and rows as needed for the other terms in the model.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

mmc,

glht.

Examples

```
## catalystm example
## See ?MMC for more on this example
data(catalystm)
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)

catalystm.mmc <-
  if.R(r=mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey")),
       s=multicomp.mmc(catalystm1.aov, plot=FALSE))
dimnames(catalystm.mmc$mca$lmat)[[1]]
lmatRows(catalystm1.aov, focus="catalyst")

## user-specified contrasts
catalystm.lmat <- cbind("AB-D" =c( 1, 1, 0,-2),
                        "A-B"  =c( 1,-1, 0, 0),
                        "ABD-C"=c( 1, 1,-3, 1))
dimnames(catalystm.lmat)[[1]] <- levels(catalystm$catalyst)
zapsmall(lmatContrast(catalystm.mmc$none$lmat, catalystm.lmat))
```

`logit` *Logistic function and its inverse.*

Description

Logistic function and its inverse.

Usage

```
logit(p)
antilogit(x)
```

Arguments

`p`	Probability value, a vector of numbers between 0 and 1, inclusive.
`x`	Real number, a vector of numbers between `-Inf` and `Inf`.

Value

Vector of real values `log(p/(1-p))` for `logit`. Vector of probabilities `exp(x)/(1+exp(x))` for `antilogit` with boundary values of `-Inf` and `Inf` for `x` correctly handled.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

```
logit(seq(0,1,.1))
antilogit(logit(seq(0,1,.1)))
```

mcalinfct *MCA multiple comparisons analysis (pairwise)*

Description

MCA multiple comparisons analysis (pairwise). We calculate the contrast matrix for all pairwise comparisons, taking account of covariates and interactions.

Usage

```
mcalinfct(model, focus,
          mmm.data=model$model,
          formula.in=terms(model),
          linfct.Means=


          multcomp.meanslinfct(model, focus, mmm.data, formula.in,
                               contrasts.arg=model$contrasts),
          type="Tukey"
          )
```

Arguments

model	aov object
focus	name of one of the factors in the model, as a character object.
mmm.data	data.frame from which the model was estimated. Normally, the default is the correct value.
formula.in	formula of the model which was estimated. Normally, the default is the correct value. The use of the terms function honors the keep.order=TRUE if it was specified.
linfct.Means	Contrast matrix for the adjusted means of each level of the focus factor. Normally, the default is the correct value.
type	Name of the multiple comparison procedure to be used. See contrMat.

Value

Matrix to be used as a value for the linfct argument to glht.

Note

This function provides results similar to the mcp(focusname="Tukey") argument to glht. I think it provides better values for covariate and interaction terms.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

MMC

Examples

```
## See the examples in HH/scripts/MMC.cc176.R
```

---

mmc *MMC (mean–mean multiple comparisons) plots.*

Description

Constructs a "mmc.multicomp" object from the formula and other arguments. The object must be explicitly plotted.

Usage

```
mmc(model, ...)  ## R

## S3 method for class 'glht'
mmc(model, ...)


## Default S3 method:
mmc(model,       ## lm object
    linfct=NULL,
    focus=
    if (is.null(linfct))
    {
      if (length(model$contrasts)==1) names(model$contrasts)
      else stop("focus or linfct must be specified.")
    }
    else
    {
      if (is.null(names(linfct)))
        stop("focus must be specified.")
      else names(linfct)
    },
    focus.lmat,
    ylabel=deparse(terms(model)[[2]]),
    lmat=if (missing(focus.lmat)) {
      t(linfct)
    } else {
```

```
      lmatContrast(t(none.glht$linfct), focus.lmat)
      },
    lmat.rows=lmatRows(model, focus),
    lmat.scale.abs2=TRUE,
    estimate.sign=1,
    order.contrasts=TRUE,
    level=.95,
    calpha=NULL,
    alternative = c("two.sided", "less", "greater"),
    ...
    )

multicomp.mmc(x,  ## S-Plus
              focus=dimnames(attr(x$terms,"factors"))[[2]][1],
              comparisons="mca",
              lmat,
              lmat.rows=lmatRows(x, focus),
              lmat.scale.abs2=TRUE,
              ry,
              plot=TRUE,
              crit.point,
              iso.name=TRUE,
              estimate.sign=1,
              x.offset=0,
              order.contrasts=TRUE,
              main,
              main2,
              focus.lmat,
              ...)

## S3 method for class 'mmc.multicomp'
x[..., drop = TRUE]
```

Arguments

model	"aov" object in "lm" method.
ylabel	name of the response variable.
lmat	contrast matrix as in the S-Plus multicomp. The convention for lmat in R is to use the transpose of the linfct component produced by glht. Required for user-specified contrasts.
lmat.rows	rows in lmat for the focus factor.
focus	define the factor to compute contrasts of. See glht.
focus.lmat	R only. Contrast matrix used in the user-specified comparisons of the focus factor. This is the matrix the user constructs. This matrix is multiplied by the lmat from the none component to create the lmat for the user-specified contrasts. Display the hibrido.lmat and maiz2.lmat in the maiz example below to see what is happening.

`linfct` In R, see `glht`.

`...` other arguments. `alternative` and `base` are frequently used with `glht`.

`comparisons` argument to `multicomp`

`lmat.scale.abs2` logical, scale the contrasts in the columns of `lmat` to make the sum of the absolute values of each column equal 2.

`estimate.sign` numeric. If `0`, leave contrasts in the default lexicographic direction. If positive, force all contrasts to positive, reversing their names if needed (if contrast A-B is negative, reverse it to B-A). If negative, the force all contrasts to positive.

`order.contrasts` sort the contrasts in the (`mca`, `none`, `lmat`) components by height on the MMC plot. This will place the contrasts in the multicomp plots in the same order as in the MMC plot.

`alternative` Direction of alternative hypothesis. See `glht` in R. S-Plus `multicomp` uses the argument `bounds` for this concept.

`level` Confidence level. Defaults to 0.95.

`crit.point, calpha` critical value for the tests. The value from the specified `multicomp` method is used for the user-specified contrasts when `lmat` is specified. This argument is called `crit.point` with `multicomp` in S-Plus and `calpha` when used with `glht` and `confint` in R. In R, with a large number of levels for the focus factor, `calpha` should be specified. See notes below for discussion of the timing issues and the examples for an illustration how to use `calpha`.

`plot` logical, display the plot if `TRUE`.

`ry, iso.name, x.offset, main, main2` arguments to `plot.mmc.multicomp`.

`x, drop` See `"["`.

Details

By default, if `lmat` is not specified, we plot the isomeans grid and the pairwise comparisons for the `focus` factor. By default, we plot the specified contrasts if the `lmat` is specified. Each contrast is plotted at a height which is the weighted average of the means being compared. The weights are scaled to the sum of their absolute values equals 2.

We get the right contrasts automatically if the aov is oneway. If we specify an lmat for oneway it must have a leading row of 0.

For any more complex design, we must study the `lmat` from the `mca` component of the result to see how to construct the `lmat` (with the extra rows as needed) and how to specify the `lmat.rows` corresponding to the rows for the focus factor.

`mmc` in R works from either an `"glht"` object or an `"aov"` object. `multicomp.mmc` in S-Plus works from an `"aov"` object.

Value

An "mmc.multicomp" object contains either the first two or all three of the "multicomp" components mca, none, lmat described here. Each "multicomp" component in R also contains a "glht" object.

mca	Object containing the pairwise comparisons.
none	Object comparing each mean to 0.
lmat	Object for the contrasts specified in the lmat argument.

"[.mmc.multicomp" is a subscript method.

Note

The multiple comparisons calculations in R and S-Plus use completely different functions. MMC plots in R are constructed by mmc based on

glht. MMC plots in S-Plus are constructed by multicomp.mmc based on the S-Plus

multicomp. The MMC plot is the same in both systems. The details of getting the plot differ.

Function mmc calls

glht and confint.glht. With a large number of levels for the focus factor, the confint function is exceedingly slow (80 minutes for 30 levels on 1.5GHz Windows XP). Therefore, always specify calpha to reduce the time to under a second for the same example.

plot.mmc.multicomp chooses sensible defaults for its many arguments. They will often need manual adjustment. The examples show several types of adjustments. We have changed the centering and scaling to avoid overprinting of label information. By default the significant contrasts are shown in a more intense color than the nonsignificant contrasts. We have an option to reduce the color intensity of the isomeans grid.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Heiberger, R.~M. and Holland, B. (2006). "Mean–mean multiple comparison displays for families of linear contrasts." *Journal of Computational and Graphical Statistics*, 15:937–955.

Hsu, J. and Peruggia, M. (1994). "Graphical representations of Tukey's multiple comparison method." *Journal of Computational and Graphical Statistics*, 3:143–161.

See Also

as.multicomp, plot.mmc.multicomp

Examples

```
## Use mmc with R.
## Use multicomp.mmc with S-Plus.

## data and ANOVA
## catalystm example
data(catalystm)

if.R(r=
  bwplot(concent ~ catalyst, data=catalystm,
         scales=list(cex=1.5),
         ylab=list("concentration", cex=1.5),
         xlab=list("catalyst",cex=1.5))
,s=
t(bwplot(catalyst ~ concent, data=catalystm,
         scales=list(cex=1.5),
         xlab=list("concentration", cex=1.5),
         ylab=list("catalyst",cex=1.5)))
)


catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
summary(catalystm1.aov)

catalystm.mca <-
if.R(r=glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey")),
     s=multicomp(catalystm1.aov, plot=FALSE))
## plot(catalystm.mca)
if.R(s=catalystm.mca,
     r=confint(catalystm.mca))

## pairwise comparisons
old.omd <- par(omd=c(0,.95,0,1))
catalystm.mmc <-
if.R(r=mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey")),
     s=multicomp.mmc(catalystm1.aov, plot=FALSE))
catalystm.mmc
if.R(s=plot(catalystm.mmc, x.offset=1),
     r=plot(catalystm.mmc, ry=c(50,58), x.offset=1.8))
## tiebreaker
plotMatchMMC(catalystm.mmc$mca, xlabel.print=FALSE)

## user-specified contrasts
catalystm.lmat <- cbind("AB-D" =c( 1, 1, 0,-2),
                        "A-B"  =c( 1,-1, 0, 0),
                        "ABD-C"=c( 1, 1,-3, 1))
dimnames(catalystm.lmat)[[1]] <- levels(catalystm$catalyst)

catalystm.mmc <-
if.R(r=mmc(catalystm1.aov,
       linfct = mcp(catalyst = "Tukey"),
       focus.lmat=catalystm.lmat),
```

```
     s=multicomp.mmc(catalystm1.aov,
       focus.lmat=catalystm.lmat, plot=FALSE))
catalystm.mmc

if.R(s=plot(catalystm.mmc, x.offset=1),
     r=plot(catalystm.mmc, ry=c(50,58), x.offset=1.8))
## tiebreaker
plotMatchMMC(catalystm.mmc$lmat, xlabel.print=FALSE, col.signif='blue')
par(old.omd)


## Dunnett's test
## weightloss example
data(weightloss)
if.R(r=
bwplot(loss ~ group, data=weightloss,
       scales=list(cex=1.5),
       ylab=list("Weight Loss", cex=1.5),
       xlab=list("group",cex=1.5))
,s=
t(bwplot(group ~ loss, data=weightloss,
       scales=list(cex=1.5),
       xlab=list("Weight Loss", cex=1.5),
       ylab=list("group",cex=1.5)))
)

weightloss.aov <- aov(loss ~ group, data=weightloss)
summary(weightloss.aov)

if.R(r={
group.count <- table(weightloss$group)
},s={})

tmp.dunnett <-
if.R(r=
glht(weightloss.aov,
     linfct=mcp(group=contrMat(group.count, base=4)),
     alternative="greater")
,s=
multicomp(weightloss.aov,
          method="dunnett", comparisons="mcc",
          bounds="lower", control=4,
          valid.check=FALSE)
)
plot(tmp.dunnett)

tmp.dunnett.mmc <-
if.R(r=
   mmc(weightloss.aov,
       linfct=mcp(group=contrMat(group.count, base=4)),
       alternative="greater")
,s=
   multicomp.mmc(weightloss.aov,
```

```
                  method="dunnett", comparisons="mcc",
                  bounds="lower", control=4,
                  valid.check=FALSE, plot=FALSE)
)

tmp.dunnett.mmc
plot(tmp.dunnett.mmc)



## two-way ANOVA
## display example

data(display)

if.R(r=
interaction2wt(time ~ emergenc * panel.ordered, data=display)
     ,s=
interaction2wt(time ~ emergenc * panel.ordered, data=display,
               xlim=c(.5,4.5), key.in=list(x=-1.8))
)

displayf.aov <- aov(time ~ emergenc * panel, data=display)
anova(displayf.aov)

## multiple comparisons
## MMC plot
displayf.mmc <-
if.R(r={
        mmc(displayf.aov,
            linfct=mcp(panel="Tukey", interaction_average=TRUE, covariate_average=TRUE))
},
     s=multicomp.mmc(displayf.aov, "panel", plot=FALSE))

if.R(s=
       plot(displayf.mmc)
    ,r=
       plot(displayf.mmc, x.offset=1, ry=c(17,26))
)


panel.lmat <- cbind("3-12"=c(-1,-1,2),
                    "1-2"=c( 1,-1,0))
dimnames(panel.lmat)[[1]] <- levels(display$panel)

displayf.mmc <-
if.R(r={
        mmc(displayf.aov, linfct=mcp(panel="Tukey"),
            `interaction_average`=TRUE, focus.lmat=panel.lmat)
},
     s=multicomp.mmc(displayf.aov, "panel",
       focus.lmat=panel.lmat, plot=FALSE))
```

```
if.R(s=
       plot(displayf.mmc)
    ,r=
       plot(displayf.mmc, x.offset=1, ry=c(17,26))
)



## split plot design with tiebreaker plot
##
## This example is based on the query by Tomas Goicoa to R-news
## http://article.gmane.org/gmane.comp.lang.r.general/76275/match=goicoa
## It is a split plot similar to the one in HH Section 14.2 based on
## Yates 1937 example.  I am using the Goicoa example here because its
## MMC plot requires a tiebreaker plot.


data(maiz)

if.R(s={old.omd <- par(omd=c(.1,1,.05,1))},
     r={})
interaction2wt(yield ~ hibrido+nitrogeno+bloque, data=maiz,
               key.in=list(x=-5), ## ignored by R
               par.strip.text=list(cex=.7))
interaction2wt(yield ~ hibrido+nitrogeno, data=maiz)
 if.R(s={par(old.omd)},
      r={})

maiz.aov <- aov(yield ~ nitrogeno*hibrido + Error(bloque/nitrogeno), data=maiz)

summary(maiz.aov)
summary(maiz.aov,
        split=list(hibrido=list(P3732=1, Mol17=2, A632=3, LH74=4)))

## multicomp(maiz.aov, focus="hibrido")         ## can't use 'aovlist' objects
## glht(maiz.aov, linfct=mcp(hibrido="Tukey"))  ## can't use 'aovlist' objects

sapply(maiz[-1], contrasts)
if.R(r={
  ## R mmc requires treatment contrasts
  contrasts(maiz$nitrogeno) <- "contr.treatment"
  contrasts(maiz$bloque) <- "contr.treatment"
  sapply(maiz[-1], contrasts)
},
     s={})

## Both R glht() and S-Plus multicomp() require aov, not aovlist
maiz2.aov <- aov(terms(yield ~ bloque*nitrogeno + hibrido/nitrogeno,
                       keep.order=TRUE), data=maiz)
summary(maiz2.aov)

## There are many ties in the group means.
## These are easily seen in the MMC plot, where the two clusters
```

```
## c("P3747", "P3732", "LH74") and c("Mol17", "A632")
## are evident from the top three contrasts including zero and the
## bottom contrast including zero.  The significant contrasts are the
## ones comparing hybrids in the top group of three to ones in the
## bottom group of two.

## We have two graphical responses to the ties.
## 1. We constructed the tiebreaker plot.
## 2. We construct a set of orthogonal contrasts to illustrate
##    the clusters.

## pairwise contrasts with tiebreakers.
if.R(s={
  maiz2.mmc <- multicomp.mmc(maiz2.aov, focus="hibrido", plot=FALSE)
  old.omd <- par(omd=c(.05,.85,0,1))
  plot(maiz2.mmc, ry=c(145,170), x.offset=4)
  par(omd=c(.05,.85,0,1))
  plotMatchMMC(maiz2.mmc$mca)
  par(old.omd)
},r={
  maiz2.mmc <- mmc(maiz2.aov,
                   linfct=mcp(hibrido="Tukey", interaction_average=TRUE))
  old.omd <- par(omd=c(.05,.85,.35,1)) ## x1 x2 y1 y2
  plot(maiz2.mmc)
  par(omd=c(.05,.85,0,.5), new=TRUE)
  plotMatchMMC(maiz2.mmc$mca, cex.axis=.7)
  par(old.omd)
})


## orthogonal contrasts
## user-specified contrasts
hibrido.lmat <- cbind("PPL-MA" =c(2, 2,-3,-3, 2),
                      "PP-L"   =c(1, 1, 0, 0,-2),
                      "P47-P32"=c(1,-1, 0, 0, 0),
                      "M-A"    =c(0, 0, 1,-1, 0))
dimnames(hibrido.lmat)[[1]] <- levels(maiz$hibrido)
hibrido.lmat
maiz2.mmc <-
  if.R(s=multicomp.mmc(maiz2.aov, focus="hibrido",
         focus.lmat=hibrido.lmat,
         plot=FALSE),
       r=mmc(maiz2.aov, linfct=mcp(hibrido="Tukey"),
            `interaction_average`=TRUE, focus.lmat=hibrido.lmat)
       )

if.R(s={
  old.omd <- par(omd=c(.05,.85,0,1))
  plot(maiz2.mmc, ry=c(145,170), x.offset=4)
  par(omd=c(.05,.85,0,1))
  plotMatchMMC(maiz2.mmc$lmat, col.signif='blue')
  par(old.omd)
},r={
```

```
  old.omd <- par(omd=c(.05,.85,.35,1)) ## x1 x2 y1 y2
  plot(maiz2.mmc)
  par(omd=c(.05,.85,0,.45), new=TRUE)
  plotMatchMMC(maiz2.mmc$lmat, cex.axis=.7, col.signif='blue')
  par(old.omd)
})
```

`mmc.mean` *MMC (mean–mean multiple comparisons) plots from the sufficient statistics for a one-way design.*

Description

Constructs a "`mmc.multicomp`" object from the sufficient statistics for a one-way design. The object must be explicitly plotted.

Usage

```
multicomp.mean(group, n, ybar, s, alpha=.05,  ## S-Plus
               ylabel="ylabel", focus.name="focus.factor", plot=FALSE,
               lmat, labels=NULL, ...,
               df=sum(n) - length(n),
               sigmahat=(sum((n-1)*s^2) / df)^.5)

multicomp.mmc.mean(group, n, ybar, s, ylabel, focus.name,  ## S-Plus
                   lmat,
                   ...,
                   comparisons="mca",
                   lmat.rows=seq(length=length(ybar)),
                   ry,
                   plot=TRUE,
                   crit.point,
                   iso.name=TRUE,
                   estimate.sign=1,
                   x.offset=0,
                   order.contrasts=TRUE,
                   method="tukey",
                   df=sum(n)-length(n),
                   sigmahat=(sum((n-1)*s^2)/df)^.5)
```

Arguments

`group`	character vector of levels
`n`	numeric vector of sample sizes
`ybar`	vector of group means

s	vector of group standard deviations
alpha	Significance levels of test
ylabel	name of response variable
focus.name	name of factor
plot	logical. Should the "mmc.multicomp" object be automatically plotted? ignored in R.
lmat	lmat from multicomp in S-Plus or t(linfct) from glht in R.
labels	labels argument for multicomp in S-Plus. Not used in R.
method	method for critical point calculation. This corresponds to method in S-Plus multicomp and to type in R glht
df	scalar, residual degrees of freedom
sigmahat	sqrt(MSE) from the ANOVA table
...	other arguments
comparisons	argument to S-Plus multicomp only.
estimate.sign, order.contrasts, lmat.rows	See lmat.rows in mmc.
ry	See argument ry.mmc in plot.mmc.multicomp.
crit.point	See argument crit.point in S-Plus multicomp. The equivalent is not in glht.
iso.name, x.offset	See plot.mmc.multicomp.

Value

multicomp.mmc.mean returns a "mmc.multicomp" object.

multicomp.mean returns a "multicomp" object.

Note

The multiple comparisons calculations in R and S-Plus use completely different functions. MMC plots in R are constructed by mmc based on

glht. MMC plots in S-Plus are constructed by multicomp.mmc based on the S-Plus

multicomp. The MMC plot is the same in both systems. The details of getting the plot differ.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Heiberger, R.~M. and Holland, B. (2006). "Mean–mean multiple comparison displays for families of linear contrasts." *Journal of Computational and Graphical Statistics*, 15:937–955.

Hsu, J. and Peruggia, M. (1994). "Graphical representations of Tukey's multiple comparison method." *Journal of Computational and Graphical Statistics*, 3:143–161.

See Also

mmc

Examples

```
## This example is from Hsu and Peruggia

## This is the S-Plus version
## See ?aovSufficient for R

if.R(r={},
s={

data(pulmonary)
pulmonary.aov <- aovSufficient(FVC ~ smoker,
                               data=pulmonary)
summary(pulmonary.aov)


## multicomp object
pulmonary.mca <-
multicomp.mean(pulmonary$smoker,
               pulmonary$n,
               pulmonary$FVC,
               pulmonary$s,
               ylabel="pulmonary",
               focus="smoker")


pulmonary.mca
## lexicographic ordering of contrasts, some positive and some negative
plot(pulmonary.mca)



pulm.lmat <- cbind("npnl-mh"=c( 1, 1, 1, 1,-2,-2), ## not.much vs lots
                   "n-pnl"  =c( 3,-1,-1,-1, 0, 0), ## none vs light
                   "p-nl"   =c( 0, 2,-1,-1, 0, 0), ## {} arbitrary 2 df
                   "n-l"    =c( 0, 0, 1,-1, 0, 0), ## {} for 3 types of light
                   "m-h"    =c( 0, 0, 0, 0, 1,-1)) ## moderate vs heavy
dimnames(pulm.lmat)[[1]] <- row.names(pulmonary)
```

```
pulm.lmat


## mmc.multicomp object
pulmonary.mmc <-
multicomp.mmc.mean(pulmonary$smoker,
                   pulmonary$n,
                   pulmonary$FVC,
                   pulmonary$s,
                   ylabel="pulmonary",
                   focus="smoker",
                   lmat=pulm.lmat,
                   plot=FALSE)


old.omd <- par(omd=c(0,.95, 0,1))

## pairwise comparisons
plot(pulmonary.mmc, print.mca=TRUE, print.lmat=FALSE)

## tiebreaker plot, with contrasts ordered to match MMC plot,
## with all contrasts forced positive and with names also reversed,
## and with matched x-scale.
plotMatchMMC(pulmonary.mmc$mca)

## orthogonal contrasts
plot(pulmonary.mmc)

## pairwise and orthogonal contrasts on the same plot
plot(pulmonary.mmc, print.mca=TRUE, print.lmat=TRUE)

par(old.omd)
})
```

`multicomp.order` *Update a multicomp object by ordering its contrasts.*

Description

Update a multicomp object by ordering its contrasts. The default `sort.by = "height"` matches the order in the MMC plot. An alternate `sort.by = "estimate"` matches the order of the half-normal plot. Or the argument `sort.order` can be used to specify any other order.

Usage

```
multicomp.order(mca, sort.by = "height", sort.order = NULL)

multicomp.label.change(x, old="adj", new="new", how.many=2)

## S3 method for class 'multicomp'
```

```
multicomp.label.change(x, old="adj", new="new", how.many=2)

## S3 method for class 'mmc.multicomp'
multicomp.label.change(x, old="adj", new="new", how.many=2)
```

Arguments

mca	"multicomp" object. This is the result of multicomp in S-Plus or the result from applying as.multicomp to a "glht" object in R.
sort.by	Either "height" or "estimate".
sort.order	Vector of indices by which the contrasts are to be sorted. When sort.order in non-NULL, it is used.
x	"multicomp" object.
old	character string to be removed from contrast names.
new	replacement character string to be inserted in contrast names.
how.many	number of times to make the replacement.

Value

The result is a "multicomp" object containing the same contrasts as the argument.
multicomp.order sorts the contrasts (and renames them consistently) according to the specifications.
multicomp.label.change changes the contrast names according to the specifications.

When sort.by=="height", sort the contrasts by the reverse order of the heights. This provides a "multicomp" object that will be plotted by plot.multicomp in the same order used by plot.mmc.multicomp. If there is not "height" component, the original "multicomp" object is returned.

When sort.by=="estimate", sort the contrasts by the reverse order of the contrast estimates. This provides the same order as the half-normal plot.

When sort.order in non-NULL, sort the contrasts in that order.

Note

S-Plus use the multicomp functions and R uses the multcomp package.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Heiberger, R.~M. and Holland, B. (2006). "Mean–mean multiple comparison displays for families of linear contrasts." *Journal of Computational and Graphical Statistics*, 15:937–955.

See Also

MMC, as.glht in R, multicomp.reverse

Examples

```
## continue with the example in mmc in R, or multicomp.mmc in S-Plus
data(catalystm)

catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)

if.R(r={
catalystm.mca <-
   glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey"))
print(confint(catalystm.mca))

catalystm.mmc <-
   mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey"))
## the contrasts have been ordered by height (see ?MMC),
## which in this example corresponds to sort.order=c(1,2,4,3,5,6),
## and reversed, to make the contrast Estimates positive.
print(as.glht(catalystm.mmc$mca))

## ## For consistency with the S-Plus example,
## ## we change all factor level "A" to "control".
## as.glht(multicomp.label.change(catalystm.mmc$mca, "A", "control"))
},s={
catalystm.mca <-
   multicomp(catalystm1.aov, method="Tukey")
print(catalystm.mca)

catalystm.mmc <-
   multicomp.mmc(catalystm1.aov, method="Tukey", plot=FALSE)
## the contrasts have been ordered by height (see ?MMC),
## which in this example corresponds to sort.order=c(1,2,4,3,5,6),
## and reversed, to make the contrast Estimates positive.
print(catalystm.mmc$mca)

## S-Plus multicomp already uses simple names.  This function is
## therefore used in more complex two-way ANOVA examples.  We illustrate
## here by changing all factor level "A" to "control".
print(multicomp.label.change(catalystm.mmc$mca, "A", "control"))
})
```

`multicomp.reverse` *Force all comparisons in a "multicomp" object to have the same sign.*

Description

Force all comparisons in a `"multicomp"` object to have the same sign. If the contrast `"A-B"` has a negative estimate, reverse it show the contrast `"B-A"` with a positive estimate.

Usage

```
multicomp.reverse(y, estimate.sign = 1, ...)
```

Arguments

y	"multicomp" object
estimate.sign	If estimate.sign==1, reverse the negatives. If estimate.sign==-1, reverse the positives. Both the names of the comparisons and the numerical values are reversed. If estimate.sign==0, return the argument.
...	other arguments not used.

Value

The result is a "multicomp" object containing the same contrasts as the argument but with the sign of the contrasts changed as needed.

Note

S-Plus use the multicomp functions and R uses the multcomp package.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Heiberger, R.~M. and Holland, B. (2006). "Mean–mean multiple comparison displays for families of linear contrasts." *Journal of Computational and Graphical Statistics*, 15:937–955.

See Also

MMC, multicomp.order

Examples

```
## see example in multicomp.order
```

norm.curve *plot a normal or a t-curve with both x and z axes.*

Description

Plot a normal curve or a t-curve with both x (with mean and se as specified) and z or t (mean=0, se=1) axes. Shade a region for rejection region, acceptance region, confidence interval. The density axis is marked in units appropriate for the z or t axis. The existence of any of the arguments se, sd, n forces dual x and (z or t) scales. When none of these arguments are used, the main title defaults to "Standard Normal Density N(0,1)" and only the z scale is printed. A second density curve, appropriate for an alternative hypothesis is displayed when the argument axis.name="z1" is specified. The shaded area is printed on the plot.

When the optional argument df.t is specified, then a t-distribution with df.t degrees of freedom is plotted.

norm.observed plots a vertical line with arrowhead markers at the location of the observed xbar.

normal.and.t.dist is a driver function that uses all the others. It's primary function is drawing a plot. It returns an invisible list containing the values it calculated and displayed on the graph.

norm.curve draws the curves and filled areas as requested by the normal.and.t.dist function. Any out of bounds errors (for example, with normal.and.t.dist(deg.free=1)) are suppressed with par(err=-1) by this function and restored to the previous value when the norm.curve function completes.

Usage

```
normal.and.t.dist(mu.H0          = 0,
                  mu.H1          = NA,
                  obs.mean       = 0,
                  std.dev        = 1,
                  n              = NA,
                  deg.freedom    = NA,
                  alpha.left     = alpha.right,
                  alpha.right    = .05,
                  Use.mu.H1      = FALSE,
                  Use.obs.mean   = FALSE,
                  Use.alpha.left = FALSE,
                  Use.alpha.right= TRUE,
                  hypoth.or.conf = 'Hypoth',
                  xmin           = NA,
                  xmax           = NA,
                  gxbar.min      = NA,
                  gxbar.max      = NA,
                  cex.crit       = 1.2,
                  polygon.density= -1,
                  polygon.lwd    = 4,
                  col.mean       = 'limegreen',
```

```
                    col.mean.label = 'limegreen',
                    col.alpha      = 'blue',
                    col.alpha.label= 'blue',
                    col.beta       = 'red',
                    col.beta.label = 'red',
                    col.conf       = 'palegreen',
                    col.conf.arrow = 'darkgreen',
                    col.conf.label = 'darkgreen'
                    )

  norm.setup(xlim=c(-2.5,2.5),
             ylim = c(0, 0.4)/se,
             mean=0,
             main=main.calc,
             se=sd/sqrt(n), sd=1, n=1,
             df.t=NULL,
             Use.obs.mean=TRUE,
              ...)

  norm.curve(mean=0, se=sd/sqrt(n),
            critical.values=mean + se*c(-1, 1)*z.975,
            z=if(se==0) 0 else
                do.call("seq", as.list(c((par()$usr[1:2]-mean)/se, length=109))),
            shade, col="blue",
            axis.name=ifelse(is.null(df.t) || df.t==Inf, "z", "t"),
            second.axis.label.line=3,
            sd=1, n=1,
            df.t=NULL,
            axis.name.expr=axis.name,
            Use.obs.mean=TRUE,
            col.label=col,
            hypoth.or.conf="Hypoth",
            col.conf.arrow=par("col"),
            col.conf.label=par("col"),
            col.crit=ifelse(hypoth.or.conf=="Hypoth", 'blue', col.conf.arrow),
            cex.crit=1.2,
            polygon.density=-1,
            polygon.lwd=4,
            col.border=ifelse(is.na(polygon.density), FALSE, col),
            ...)

  norm.observed(xbar, t.xbar, t.xbar.H1=NULL,
                col="green",
                p.val=NULL, p.val.x=par()$usr[2]+ left.margin,
                t.or.z=ifelse(is.null(deg.free) || deg.free==Inf, "z", "t"),
                t.or.z.position=par()$usr[1]-left.margin,
                cex.small=par()$cex*.7, col.label=col,
                xbar.negt=NULL, cex.large=par()$cex,
```

```
                left.margin=.15*diff(par()$usr[1:2]),
                sided="", deg.free=NULL)

norm.outline(dfunction, left, right, mu.H0, se, deg.free=NULL,
             col.mean="green")
```

Arguments

`xlim, ylim, xmin, xmax, gxbar.min, gxbar.max`
: `xlim, ylim`. Defaults to correct values for standard Normal(0,1). User must set values for other mean and standard error.

`mean`
: Mean of the normal distribution in xbar-scale, used in calls to `dnorm`.

`se`
: standard error of the normal distribution in xbar-scale, used in calls to `dnorm`.

`sd, std.dev, n`
: standard deviation and sample size of the normal distribution in x-scale. These may be used as an alternate way of specifying the standard error `se`.

`df.t, deg.freedom`
: Degrees of freedom for the t distribution. When `df.t` is `NULL`, the normal distribution is used.

`critical.values`
: Critical values in xbar-scale. A scalar value implies a one-sided test. A vector of two values implies a two-sided test.

`main`
: Main title.

`z`
: z-values (standardized to N(0,1)) used as base of plot.

`shade`
: Valid values for shade are "right", "left", "inside", "outside", "none". Default is "right" for one-sided critical.values and "outside" for two-sided critical values.

`col`
: color of the shaded region.

`col.label, col.alpha, col.alpha.label`
: color of the area of the shaded rejection region and its label.

`col.beta, col.beta.label`
: color of the area of the shaded region For Type II error and its label.

`hypoth.or.conf`
: `"Hypoth"` or `"Conf"`

`col.conf`
: Color of plot within confidence limits.

`col.conf.arrow`
: Color of arrow denoting confidence limits.

`col.conf.label`
: Color of label giving confidence level.

`col.mean.label`
: Color of label for observed mean.

`col.crit, cex.crit`
: Color and cex of critical values.

`axis.name, axis.name.expr`
: defaults to `"z"` for the standard normal scale centered on the null hypothesis value of the mean or to `"t"` for the t distribution with `df.t` degrees of freedom. For alternative hypotheses, the user must specify either `"z1"` or `"t1"` for the standard normal scale, or t distibution with `df.t` degrees of freedom, centered on the alternate hypothesis value of the mean. The `axis.name.expr` allows R users to say `expression(z[1])` to get real subscripts.

second.axis.label.line
: Defaults to 3. Normally not needed. When two curves are drawn, one normal and one t, then the second curve needs a different label for the y-axis. Set this value to 4 to avoid overprinting.

xbar, obs.mean
: xbar-value of the observed data.

t.xbar
: t-value of the observed data under the null hypothesis.

...
: Other arguments which are ignored.

Use.obs.mean
: Logical. If TRUE, then include "mean" on the plot.

alpha.right, alpha.left
: Area in tail of curve.

Use.alpha.right, Use.alpha.left
: Logical. If TRUE, then include the specified α on the plot.

t.xbar.H1
: t-value under alternate hypothesis.

p.val
: under specified hypothesis

p.val.x,t.or.z.position
: location on x-axis to put label

t.or.z
: label for axis.

cex.small
: cex for left margin labels of axis.

xbar.negt
: location in data scale of negative t- or z-value corresponding to observed x-value. Used for two-sided p-values.

cex.large
: cex for labels in top margin.

left.margin
: distance to the left of par()$usr[1].

sided
: type of test.

deg.free
: degrees of freedom or NULL.

dfunction
: "dnorm" or "dt"

left
: left end of interval

right
: right end of interval

mu.H0, mu.H1
: mean under the null hypothesis and alternative hypothesis.

Use.mu.H1
: Logical. If TRUE, then include mu.H1 on the plot.

col.mean
: Color of outline.

polygon.density, polygon.lwd, col.border
: density, lwd, border arguments to polygon. polygon.density is -1 by default to give a solid color filled region. Setting polygon.density to a positive value (we recommend 10) gives a diagonally-hatched area appropriate for printing the graph on a black and white printer.

Value

An invisible list containing the calculated values of probabilities and critical values in the data scale, the null hypothesis z- or t-scale, and the alternative hypothesis z- or t-scale, as specified. The components are: beta.left, beta.middle, beta.right, crit.val, crit.val.H1, crit.val.H1.left, crit.val.left, crit.val.left.z, crit.val.z, obs.mean.H0.p.val, obs.mean.H0.side, obs.mean.H0.z, obs.mean.H1.z, obs.mean.x.neg, obs.mean.x.pos, obs.mean.z.pos, standard, standard.error, standard.normal

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

```
normal.and.t.dist()
normal.and.t.dist(xmin=-4)
normal.and.t.dist(std.dev=2)
normal.and.t.dist(std.dev=2, Use.alpha.left=TRUE, deg.free=6)
normal.and.t.dist(std.dev=2, Use.alpha.left=TRUE, deg.free=6, gxbar.max=.20)
normal.and.t.dist(std.dev=2, Use.alpha.left=TRUE, deg.free=6,
                  gxbar.max=.20, polygon.density=10)
normal.and.t.dist(std.dev=2, Use.alpha.left=FALSE, deg.free=6,
                  gxbar.max=.20, polygon.density=10,
                  mu.H1=2, Use.mu.H1=TRUE,
                  obs.mean=2.5, Use.obs.mean=TRUE, xmin=-7)
normal.and.t.dist(std.dev=2, hypoth.or.conf="Conf")
normal.and.t.dist(std.dev=2, hypoth.or.conf="Conf", deg.free=8)

old.par <- par(oma=c(4,0,2,5), mar=c(7,7,4,2)+.1)

norm.setup()
norm.curve()

norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(100, 5, 100+5*(1.645))
norm.observed(112, (112-100)/5)
norm.outline("dnorm", 112, par()$usr[2], 100, 5)

norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(100, 5, 100+5*(-1.645), shade="left")

norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(mean=100, se=5, col='red')

norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(100, 5, 100+5*c(-1.96, 1.96))

norm.setup(xlim=c(-3, 6))
norm.curve(critical.values=1.645, mean=1.645+1.281552, col='green',
           shade="left", axis.name="z1")
norm.curve(critical.values=1.645, col='red')

norm.setup(xlim=c(-6, 12), se=2)
norm.curve(critical.values=2*1.645, se=2, mean=2*(1.645+1.281552),
           col='green', shade="left", axis.name="z1")
norm.curve(critical.values=2*1.645, se=2, mean=0,
           col='red', shade="right")


par(mfrow=c(2,1))
norm.setup()
```

```
norm.curve()
mtext("norm.setup(); norm.curve()", side=1,  line=5)
norm.setup(n=1)
norm.curve(n=1)
mtext("norm.setup(n=1); norm.curve(n=1)", side=1,  line=5)
par(mfrow=c(1,1))


par(mfrow=c(2,2))

## naively scaled,
## areas under the curve are numerically the same but visually different
norm.setup(n=1)
norm.curve(n=1)
norm.observed(1.2, 1.2/(1/sqrt(1)))
norm.setup(n=2)
norm.curve(n=2)
norm.observed(1.2, 1.2/(1/sqrt(2)))
norm.setup(n=4)
norm.curve(n=4)
norm.observed(1.2, 1.2/(1/sqrt(4)))
norm.setup(n=10)
norm.curve(n=10)
norm.observed(1.2, 1.2/(1/sqrt(10)))
mtext("areas under the curve are numerically the same but visually different",
      side=3, outer=TRUE)

## scaled so all areas under the curve are numerically and visually the same
norm.setup(n=1, ylim=c(0,1.3))
norm.curve(n=1)
norm.observed(1.2, 1.2/(1/sqrt(1)))
norm.setup(n=2, ylim=c(0,1.3))
norm.curve(n=2)
norm.observed(1.2, 1.2/(1/sqrt(2)))
norm.setup(n=4, ylim=c(0,1.3))
norm.curve(n=4)
norm.observed(1.2, 1.2/(1/sqrt(4)))
norm.setup(n=10, ylim=c(0,1.3))
norm.curve(n=10)
norm.observed(1.2, 1.2/(1/sqrt(10)))
mtext("all areas under the curve are numerically and visually the same",
      side=3, outer=TRUE)

par(mfrow=c(1,1))


## t distribution
mu.H0 <- 16
se.val <- .4
df.val <- 10
crit.val <- mu.H0 - qt(.95, df.val) * se.val
mu.alt <- 15
obs.mean <- 14.8
```

```
alt.t <- (mu.alt - crit.val) / se.val
norm.setup(xlim=c(12, 19), se=se.val, df.t=df.val)
norm.curve(critical.values=crit.val, se=se.val, df.t=df.val, mean=mu.alt,
           col='green', shade="left", axis.name="t1")
norm.curve(critical.values=crit.val, se=se.val, df.t=df.val, mean=mu.H0,
           col='gray', shade="right")
norm.observed(obs.mean, (obs.mean-mu.H0)/se.val)

## normal
norm.setup(xlim=c(12, 19), se=se.val)
norm.curve(critical.values=crit.val, se=se.val, mean=mu.alt,
           col='green', shade="left", axis.name="z1")
norm.curve(critical.values=crit.val, se=se.val, mean=mu.H0,
           col='gray', shade="right")
norm.observed(obs.mean, (obs.mean-mu.H0)/se.val)



## normal and t
norm.setup(xlim=c(12, 19), se=se.val, main="t(6) and normal")
norm.curve(critical.values=15.5, se=se.val, mean=16.3,
           col='gray', shade="right")
norm.curve(critical.values=15.5, se.val, df.t=6, mean=14.7,
           col='green', shade="left", axis.name="t1", second.axis.label.line=4)
norm.curve(critical.values=15.5, se=se.val, mean=16.3,
           col='gray', shade="none")

norm.setup(xlim=c(12, 19), se=se.val, main="t(6) and normal")
norm.curve(critical.values=15.5, se=se.val, mean=15.5,
           col='gray', shade="right")
norm.curve(critical.values=15.5, se=se.val, df.t=6, mean=15.5,
           col='green', shade="left", axis.name="t1", second.axis.label.line=4)
norm.curve(critical.values=15.5, se=se.val, mean=15.5,
           col='gray', shade="none")



par(old.par)
```

`npar.arma` *Count the number of parameters in an ARIMA model specification.*

Description

Count the number of parameters in an ARIMA model specification. When `arima==FALSE`, just the AR and MA parameters are counted. When `arima==TRUE`, then the number of difference parameters are also included.

Usage

```
npar.arma(x, arima=FALSE)
npar.sarma(model, arima=FALSE)
npar.rarma(arma, arima=FALSE)
```

Arguments

x	An "arima" object in S-Plus or a "Arima" object in R.
model	A model specification in the S-Plus style.
arma	A arma specification in the R style
arima	Logical. TRUE is number of differencings is to be counted.

Value

A scalar number giving the count.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

Examples

```
co2.arima <-
  if.R(s=
       arima.mle(co2, list(list(order=c(0,1,1)),
                           list(order=c(0,1,1), period=12)))
       ,r=
       arima(co2,
             order=c(0,1,1),
             seasonal=list(order=c(0,1,1), period=12))
       )

npar.arma(co2.arima)

npar.arma(co2.arima, arima=TRUE)

npar.sarma(list(list(order=c(0,1,1)),
                list(order=c(0,1,1), period=12)))

npar.sarma(list(list(order=c(0,1,1)),
                list(order=c(0,1,1), period=12)),
           arima=TRUE)

if.R(s={},
     r=npar.rarma(co2.arima$arma)
)
if.R(s={},
     r=npar.rarma(co2.arima$arma,
           arima=TRUE)
)
```

objip *loop through all attached directories looking for pattern*

Description

Loop objects() through all attached directories (items in the search() list) looking for a regular expression pattern.

Usage

```
objip(pattern, where = search(), frame=NULL, all.names=FALSE)
```

Arguments

pattern	Character string containing a regular expression that is used to list only a subset of the objects. Only names matching 'pattern' are returned.
where	an object defining a database in the search list.
frame	In S-Plus, an integer giving the frame number. In R, frame is ignored.
all.names	In R, a logical that is passed to the ls function. If 'TRUE', all object names are returned. If 'FALSE', names which begin with a '.' are omitted.

Value

A list of 0 or more character vectors. Each character vector has the name of one of the items in the search() list. Each character vector contains the names of the objects in the specified environment which match the pattern. If there are no matching names in an environment, then the corresponding character vector is removed from the result.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

ls,

Examples

```
objip("qq")
objip("^qq")
objip("qq$")
## Not run:
## R only
objip("rowSums", all.names=TRUE)

## End(Not run)
```

OddsRatio *Calculate or plot the odds ratio for a 2x2 table of counts.*

Description

Calculate or plot the odds ratio for a 2x2 table of counts. The plot shows the confidence intervals on the probability of row2 for fixed odds ratio and specified probability for row1.

Usage

```
OddsRatio(x, alpha = 0.05)

plotOddsRatio(x,
 ylab = "prob(col1 | row1)", xlab = "prob(col1 | row2)",
 alpha = c(0.05, 0.5),
 legend.x=1.05,
 oma=c(0,0,0,5), ...)
```

Arguments

x	2 x 2 table of counts
alpha	Significance levels of test. OddsRatio requires a single number in the range [0,1]. plotOddsRatio accepts more than one number on the range [0,1] and draws confidence lines at each value.
xlab, ylab	x- and y-labels for the plot Sensible defaults are generated.
...	other arguments, currently ignored.
legend.x	x position of left-hand side of legend.
oma	outer margin par()$oma, needed to make room for legend.

Value

plotOddsRatio draws a plot and invisibly returns the same list as OddsRatio for the first value of alpha. OddsRatio returns the list:

p1, p2	proportion of each row total observed in the first column.
omega1, omega2	odds for each row, p/(1-p)
psihat	odds ratio, omega2/omega1
s.ln.psihat	standard deviation of ln(psihat)
ci.ln.psihat	confidence interval for ln(psihat) using normal approximation
ci.psihat	confidence interval for psihat calculated ase p(ci.ln.psihat)
prob1	seq(0,1,.05), set of p1 values for plotting.
odds1	p1/(1-p1)
odds2	odds for the second row needed to retain psihat with the specified odds1, calculated as odds1*psihat.

ci.odds2	confidence interval for odds2
prob2	odds2 / (odds2+1)
ci.prob2	ci.odds2 / (ci.odds2+1)

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Examples

```
data(glasses)

## draw the iso-odds ratio plot with 50% CI and 95% CI,
## invisibly return the 95% CI
plotOddsRatio(glasses)
```

---

orthog.complete — *Construct an orthogonal matrix which is an arbitrary completion of the column space of the input set of columns.*

Description

Construct an orthogonal matrix which is an arbitrary completion of the column space of the input set of columns.

Usage

```
orthog.complete(x, normalize=TRUE, abs2.rows=1:nrow(x),
                Int=TRUE, drop.Int=Int)

orthog.construct(y, x, x.rows, normalize=FALSE)
```

Arguments

x	For orthog.complete, an n-row matrix of one or more columns. For orthog.construct, a set of contrasts for a factor.
y	matrix of coefficients specifying the linear combinations estimated. This will usually be the lmat from an S-Plus "multicomp" object or the linfct component from an R "glht" object.

`normalize, abs2.rows, x.rows`
: The default normalizes the sum of squares of the rows in `abs2.rows` or `x.rows` to 1. The optional value `normalize="abs2"` scales the rows in `abs2.rows` or `x.rows` to make the sum of all positive value equal 1 and the sum of all negative values equal -1. Together, the sum of the absolute values of the specified rows in each column is 2.

`Int`
: logical. Default `TRUE` means make all columns orthogonal to the constant column (Intercept in regression terminology). The alternative is to use only the columns in the input matrix `x`.

`drop.Int`
: logical. The default is to drop the constant column and to keep all columns when the constant is not automatically generated.

Details

This function is based on `qr.Q`. The input matrix `x` has `n` rows and an arbitrary non-zero number of columns. The result is an `n` by `n` orthogonal matrix. By default the first column of the result is constant and is not returned. The second column of the result is orthogonal to the first result column. Together the first two result columns span the space of the constant column and the first input column. The third result column is orthogonal to the first two result columns and the the three result columns together span the space of the constant column and the first two inout columns. Similarly for the remaining result columns. Result columns beyond the number of input columns are constructed as an arbitrary orthogonal completion.

If the input columns are orthogonal to each other and to the constant column, then the result columns are rescaled versions of the input columns.

Optionally (`drop.Int=FALSE`), the constant column can be returned.

By default the columns are scaled to have sum of squares equal 1. If `normalize="abs2"`, they are scaled to make the sum of all positive value equal 1 and the sum of all negative values equal -1. Together, the sum of the absolute values of each column is 2.

If the input is a set of columns from a contrast matrix for a design that has multiple terms, the `abs2.rows` argument is used to specify which rows are to be included in the normalization. These will normally be rows associated with one of the main effects.

Value

Matrix of orthogonal columns.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

MMC

Examples

```
zapsmall(
orthog.complete(cbind("4-12"=c(-1,-1, 0, 2),
                      "1-2" =c( 1,-1, 0, 0)))
)

zapsmall(
orthog.complete(cbind("4-12"=c(-1,-1, 0, 2),
                      "1-2" =c( 1,-1, 0, 0)),
                drop.Int=FALSE)
)

zapsmall(
orthog.complete(cbind("4-12"=c(-1,-1, 0, 2),
                      "1-2" =c( 1,-1, 0, 0)),
                normalize="abs2")
)


## used in MMC plots
tmp <- data.frame(y=rnorm(12),
                  a=factor(c("u","u","u","u",
                             "v","v","v","v",
                             "w","w","w","w")))
tmp.aov <- aov(y ~ a, data=tmp)
lmat <- if.R(
  s=multicomp(tmp.aov, focus="a")$lmat,
  r={lmat.reduced <- t(glht(tmp.aov, linfct=mcp(a="Tukey"))$linfct)
     rbind(lmat.reduced, AU=-apply(lmat.reduced[-1,], 2, sum))
    })
zapsmall(lmat)

lmat.complete <- orthog.complete(lmat, abs2.rows=-1,
                                 normalize="abs2",
                                 drop.Int=FALSE)[,1:3]
zapsmall(lmat.complete)
if.R(r=zapsmall(lmat.complete[-4,]),
     s={})
```

`panel.acf` *Panel functions for tsdiagplot.*

Description

Panel functions for `tsdiagplot`.

Usage

```
panel.acf(..., n.used)
panel.std.resid(...)
panel.gof(...)
```

Arguments

`...`	standard arguments to panel functions.
`n.used`	number of lags. THis number is needed to calculate the confidence interval which is `2//sqrt(n.used)`.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

`tsdiagplot`

---

panel.axis.right *Right-justify right-axis tick labels.*

Description

`panel.axis.right` is almost identical to `panel.axis`. `axis.RightAdjustRight` is almost identical to `axis.default`. The only difference is that these functions right-justify right-axis tick labels.

Usage

```
panel.axis.right(side = c("bottom", "left", "top", "right"),
                 at = pretty(scale.range),
                 labels = TRUE, draw.labels = TRUE,
                 check.overlap = FALSE, outside = FALSE, ticks = TRUE,
                 half = !outside,
                 which.half = switch(side, bottom = "lower",
                                     left = "upper", top = "upper",
                                     right = "lower"),
                 tck = as.numeric(ticks),
                 rot = if (is.logical(labels)) 0 else c(90, 0),
                 text.col = axis.text$col, text.alpha = axis.text$alpha,
                 text.cex = axis.text$cex, text.font = axis.text$font,
                 text.fontfamily = axis.text$fontfamily,
                 text.fontface = axis.text$fontface,
                 text.lineheight = axis.text$lineheight,
                 line.col = axis.line$col, line.lty = axis.line$lty,
                 line.lwd = axis.line$lwd, line.alpha = axis.line$alpha)
```

```
axis.RightAdjustRight(side = c("top", "bottom", "left", "right"),
                      scales, components, as.table,
                      labels = c("default", "yes", "no"),
                      ticks = c("default", "yes", "no"),
                      ...,

                      prefix = lattice.lattice.getStatus("current.prefix"))
```

Arguments

`side, at, labels, draw.labels, check.overlap, outside, ticks, half, which.half`
See `panel.axis` and `axis.default`

`tck, rot, text.col, text.alpha, text.cex, text.font, text.fontfamily`
See `panel.axis` and `axis.default`

`text.fontface, text.lineheight, line.col, line.lty, line.lwd, line.alpha`
See `panel.axis` and `axis.default`

`scales, components, as.table, ..., prefix`
See `axis.default`

Author(s)

Deepayan Sarkar `Deepayan.Sarkar@R-project.org` wrote `panel.axis` and `axis.default`. David Winsemius wrote the modification `panel.axis.right`. Richard Heiberger `rmh@temple.edu` wrote the modification `axis.RightAdjustRight`. Richard Heiberger is maintaining this distribution of both functions.

See Also

`panel.axis`

---

panel.bwplot.intermediate.hh

Panel functions for bwplot.

Description

Panel function for `bwplot` that give the user control over the placement of the boxes. When used with a positioned factor, the boxes are placed according to the position associated with the factor.

Usage

```
panel.bwplot.intermediate.hh(x, y, horizontal = TRUE,
          transpose=!horizontal,
          pch, col,
          at,  ## formerly S-Plus only, now totally ignored
          ...)
```

Arguments

`x, y, pch, col, horizontal`	see `xyplot` and `panel.bwplot`.
`transpose`	S-Plus only. The HH library transposes `"trellis"` `bwplot` objects to put the response variable on the vertical axis. R does the equivalent by placing the response variable on the left side of the `"~"` in the formula and with the `horizontal` argument.
`at`	formerly S-Plus only, now totally ignored.
`...`	Extra arguments, if any, for 'panel.bwplot'.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

`panel.xyplot`, `xyplot`, `interaction2wt`, `position`

`panel.bwplot.superpose`

Panel function for bwplot that displays an entire box in the colors coded by groups.

Description

Panel function for bwplot that displays an entire box (central dot, box, umbrella, outliers) in the same color, coded by the groups argument. The function is based on `panel.superpose`.

Usage

```
panel.bwplot.superpose(x, y, ...,
                       groups=groups,
                       col=rep(trellis.par.get("superpose.symbol")$col,
                               length=length(groups)),
                       pch=trellis.par.get("box.dot")$pch,
                       panel.groups=panel.bwplot.groups)

panel.bwplot.groups(..., col, pch, fill, fill.alpha=NULL, group.number)
```

Arguments

x, y	Standard arguments to a **lattice** panel function. When x has class positioned (see position), the position will be forwarded by panel.bwplot.superpose to panel.bwplot.groups.
...	Additional **lattice** arguments.
groups	Factor to be used for color coding entire boxes: central dot, rectangle, umbrella, and outlier symbol.
col	Colors to be assigned to the levels of the group. The default colors are taken from trellis.par.get("superpose.symbol")$col.
pch	Standard **lattice** arguments. The pch describes the central dot. The outlier dots are specified in the plot.symbol component of trellis.par.get.
fill, fill.alpha	These are related to the similarly named arguments in panel.bwplot. The fill argument is ignored. It is there to capture the automatically generated fill argument. The defaultNULL value of fill.alpha implies that there is no background color for the boxes. The user can set fill.alpha to a number between 0 and 1. The boxes will be shaded in a lighter version of their color as implied by the groups argument. The value 0 gives a transparent fill, and the value one makes the box the full opaque color.
panel.groups, group.number	See panel.superpose for details.

Details

panel.bwplot.superpose is the user-level function. panel.bwplot.groups is the panel.groups function called by panel.superpose.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

position, panel.bwplot.intermediate.hh, panel.superpose

Examples

```
tmp <- data.frame(Response=rnorm(20), Group=factor(rep(LETTERS[1:3], c(5,7,8))))

bwplot(Group ~ Response, data=tmp,
       main="Default panel.bwplot, groups ignored", groups=Group)

bwplot(Group ~ Response, data=tmp,
       main="panel.bwplot.superpose",
       groups=Group, panel=panel.bwplot.superpose)

bwplot(Group ~ Response, data=tmp,
       main="panel.bwplot.superpose with fill specified",
```

```
       groups=Group, panel=panel.bwplot.superpose,
       fill.alpha=.33)


bwplot(Group ~ Response, data=tmp,
       main="panel.bwplot.superpose, with color specified",
       groups=Group, panel=panel.bwplot.superpose,
       col=c("forestgreen","blue", "brown"))


Test <- data.frame(id=rep(letters, each=4),
                   Week=rep(c(0,1,3,6), 26),
                   Treatment=rep(c("A","B"), each=52),
                   y=rep(1:4, 52) + rep(4:5, each=52) + rnorm(104))
Test$WeekTrt <- with(Test, interaction(Week, Treatment))
position(Test$Week) <- unique(Test$Week)
position(Test$WeekTrt) <- as.vector(outer(position(Test$Week), c(-.2, .2), `+`))

tapply(Test$y, Test[c("Week", "Treatment")], median)

bwplot( y ~ WeekTrt, groups = Treatment, data = Test,
      main="default panel.bwplot, groups ignored")

bwplot( y ~ WeekTrt, groups = Treatment, data = Test,
      panel=panel.bwplot.superpose,
      scales=list(x=list(limits=c(-1, 7))),
      main="Minimal use of panel.bwplot.superpose")

bwplot( y ~ WeekTrt, groups = Treatment, data = Test,
       panel=panel.bwplot.superpose,
       scales=list(x=list(limits=c(-1, 7), at=position(Test$Week))),
       box.width=.3,
       xlab="Week",
       pch=c(17, 16),
       key=list(col=trellis.par.get()$superpose.symbol$col[1:2],
           border=TRUE, title="Treatment", cex.title=1, columns=2,
           text=list(levels(Test$Treatment)),
           points=list(pch=c(17, 16))),
       par.settings=list(plot.symbol=list(pch=c(17, 16), cex=.5)),
       main="panel.bwplot.superpose with additional annotations")


bwplot( y ~ WeekTrt, groups = Treatment, data = Test,
       panel=panel.bwplot.superpose,
       scales=list(x=list(limits=c(-1, 7), at=position(Test$Week))),
       box.width=.3,
       xlab="Week",
       pch=c(17, 16),
       key=list(col=trellis.par.get()$superpose.symbol$col[1:2],
           border=TRUE, title="Treatment", cex.title=1, columns=2,
           text=list(levels(Test$Treatment)),
           points=list(pch=c(17, 16))),
       par.settings=list(plot.symbol=list(pch=c(17, 16), cex=.5)),
```

```
main="panel.bwplot.superpose with fill and more complex panel.groups",
panel.groups = function(...) {
    panel.stripplot(...)
    panel.bwplot.groups(...)
},
fill.alpha=.33,
jitter.data = TRUE)
```

`panel.bwplott` *Extension to S-Plus trellis to allow transposed plots.*

Description

Extension to S-Plus trellis to allow transposed plots. All x - and y-components of the trellis object are interchanged. This function is not needed in R as lattice has a horizontal argument in its definitions.

Usage

```
panel.bwplott(x, y, box.ratio = 1,
              font = box.dot$font, pch = box.dot$pch, cex = box.dot$cex,
              col = box.dot$col, ..., transpose=FALSE)
```

Arguments

`x, y, box.ratio, font, pch, cex, col, ...`	See `panel.bwplot`.
`transpose`	logical. If `FALSE`, the plot is printed in the default orientation. If `TRUE`, the x- and y-components of the trellis object are interchanged. This has the effect, for example, of displaying vertical boxplots instead of the default horizontal boxplots.

Value

The function is used for its side effect of drawing boxplots in a trellis panel.

Note

This function is not needed in R. If it is used and

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`xyplot`.

`panel.cartesian` *trellis panel function, with labeled rows and columns and without strip labels.*

Description

trellis panel function, with labeled rows and columns and without strip labels. Designed for use with the ladder of powers plot.

Usage

```
panel.cartesian(x, y,
                x.label=unique(panel.labels[,"x"]),
                y.label=unique(panel.labels[,"y"]),
                group.label.side="",
                axis3.line=1,
                xg.label, yg.label, g.cex=.7,
                rescale=list(x=TRUE,y=TRUE), ...,
                browser.on=FALSE)
```

Arguments

`x, y`	x and y as for any other panel function
`x.label`	labels for the columns of the scatterplot matrix
`y.label`	labels for the rows of the scatterplot matrix
`axis3.line`	The `x.label` doesn't always show up in the right place. This allows the user to adjust it's position.
`group.label.side`	`c("","left","top")`, when the plotting formula is conditioned on a group factor, the levels of the group are displayed in the margins of the plot. The appearance depends on the setting of the trellis `between` argument. Getting it to look good for any given plot requires experimentation. Since it is redundant with the information in the strip labels, leaving it at the default `""` is often the best thing to do.
`xg.label`	group labels for rows of the scatterplot matrix
`yg.label`	group labels for rows of the scatterplot matrix
`g.cex`	`cex` for the group labels
`rescale`	alternate way to get something similar to `relation="free"`
`...`	other arguments
`browser.on`	logical, normally `FALSE`. This is a debugging tool. When `TRUE`, the `browser()` is turned on at various critical points.

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

ladder, xysplom

Examples

```
data(rent)  ## Weisberg's file alr162

rent.lm <- lm(rnt.alf ~ rnt.till + cow.dens + lime, data=rent)
rent$resid.rent <- resid(rent.lm)

xysplom(resid.rent ~ rnt.till + cow.dens | lime, data=rent,
        layout=c(2,2))

xysplom(resid.rent ~ rnt.till + cow.dens | lime, data=rent,
        layout=c(2,2),
        xlab="", ylab="",
        x.label="", y.label="",
        group.label.side="",
        par.strip.text=list(cex=1.2),
        panel=panel.cartesian,
        axis3.line=2.4,
        scales=list(
          relation="same",
          alternating=FALSE, labels=FALSE, ticks=FALSE),
        between=list(x=1, y=3))

xysplom(resid.rent ~ rnt.till + cow.dens | lime, data=rent,
        layout=c(2,2),
        xlab="", ylab="",
        x.label="", y.label="",
        group.label.side="",
        par.strip.text=list(cex=1.2),
        panel=panel.cartesian,
        axis3.line=3.6,
        scales=list(
          relation="same",
          alternating=FALSE, labels=FALSE, ticks=FALSE),
        rescale=list(x=FALSE, y=FALSE),
        between=list(x=1, y=3))

xysplom(resid.rent ~ rnt.till + cow.dens | lime, data=rent,
        layout=c(2,2),
        xlab="", ylab="",
        x.label="", y.label="",
        group.label.side="",
        par.strip.text=list(cex=1.2),
```

```
        panel=panel.cartesian,
        axis3.line=3.6,
        scales=list(
          relation="free",
          alternating=FALSE, labels=FALSE, ticks=FALSE),
        between=list(x=1, y=3))

tmp <-
xysplom(resid.rent ~ rnt.till + cow.dens | lime, data=rent,
        layout=c(2,2),
        xlab="", ylab="",
        y.label="resid",
        group.label.side="top",
        par.strip.text=list(cex=1.2),
        panel=panel.cartesian,
        axis3.line=3.6,
        scales=list(alternating=FALSE, labels=FALSE, ticks=FALSE),
        rescale=list(x=FALSE, y=FALSE),
        between=list(x=4, y=5))
if.R(r=tmp$par.settings <- list(layout.widths=list(right.padding=4)),
     s={})
tmp
```

`panel.ci.plot` *Default Panel Function for ci.plot*

Description

This is the default panel function for `ci.plot`.

Usage

```
panel.ci.plot(x, y, newdata, newfit = newfit, ...)
```

Arguments

`x`	Observed values of predictor variable.
`y`	Observed values of response variable.
`newdata`	`x` values for which predictions are calculated.
`newfit`	`data.frame` containing six components: `"fit"`, `"se.fit"`, `"residual.scale"`, `"df"`, `"ci.fit"`, `"pi.fit"`. In S-Plus these are the output from the `predict.lm` function. In R they are a rearrangement of the output of the `predict.lm` function.
`...`	other arguments passed to `panel.xyplot`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

ci.plot, xyplot, lm

`panel.dotplot.tb` *Dotplot with evenly spaced tiebreakers.*

Description

Dotplot with evenly spaced tiebreakers. Multiple hits on a specific x value are stacked.

Usage

```
panel.dotplot.tb(x, y, factor=.1,
                 jitter.data=TRUE, horizontal=TRUE,
                 max.freq=max(sapply(subsets, length)),
                 ...)
```

Arguments

`x, y`	See `xyplot`.
`factor`	jitter factor, see `xyplot`. Increment is `factor/max.freq` where `max.freq` is the maximum number of duplicates of any x value in any y group.
`jitter.data, horizontal`	Always `TRUE`.
`max.freq`	maximum number of observation at any combination of response values, factor levels, and group levels. If the formula includes one or more conditioning factors, then the user is responsible for providing a value for `max.freq`.
`...`	Other arguments for `xyplot`.

Details

Creates (possibly grouped) Dotplot of x against y. y is the 'factor'.

Warning

If the formula includes one or more conditioning factors, then the user is responsible for providing a value for `max.freq`. The default behavior is a different `max.freq` for each panel in a multi-panel display.

Author(s)

Richard M. Heiberger

Maintainer: Richard M. Heiberger <rmh@temple.edu>

Examples

```
x <- c(1,1,2,2,2,5,4,2,1,5)
y <- factor(letters[rep(1:2, 5)])

dotplot(x, panel=panel.dotplot.tb)
dotplot(x, panel=panel.dotplot.tb, factor=.2)
dotplot(y ~ x, panel=panel.dotplot.tb)
dotplot(y ~ x, panel=panel.dotplot.tb, cex=1.5, factor=.15)



quiz <- data.frame(scores=sample(10, 360, replace=TRUE),
                   date=rep(rep(c("0902", "0916", "0930"), c(40,40,40)), 3),
                   section=rep(
                     c("Stat1-3", "Stat1-5", "Stat1-8"),
                     c(120,120,120)))

dotplot(date ~ scores | section, data=quiz,
        panel=panel.dotplot.tb, factor=.5)

dotplot(date ~ scores | section, data=quiz,
        panel=panel.dotplot.tb, factor=.5,
        layout=c(1,3), between=list(y=1),
        main='Three quizzes for three sections of Stat 1')


## If the formula includes one or more conditioning factors, then the
## user is responsible for providing a value for the argument max.freq
##
a <- rep(1, 10)
z <- c(1,1,2,2,2,3,2,3,1,1)
g <- LETTERS[c(1,1,1,1,1,2,2,2,2,2)]

print(split=c(1,1,2,1), more=TRUE,
dotplot( a ~ z | g, panel=panel.dotplot.tb,
        factor=.6, cex=1.5, layout=c(2,1),
        main="different scaling in each panel")
)

print(split=c(2,1,2,1), more=FALSE,
dotplot( a ~ z | g, panel=panel.dotplot.tb, max.freq=3,
        factor=.6, cex=1.5, layout=c(2,1),
        main="same scaling in each panel")
)
```

`panel.interaction2wt` *Plot all main effects and twoway interactions in a multifactor design*

Description

This is the panel function for `interaction2wt`. The main diagonal displays boxplots for the main effects of each factor. The off-diagonals show the interaction plots for each pair of factors. The `i,j` panel shows the same factors as the `j,i` but with the trace- and x-factor roles interchanged.

Usage

```
panel.interaction2wt(x, y, subscripts,
                     responselab, trace.values,
                     factor.levels, factor.position,
                     fun = mean,
                     se,
                     ...,
                     box.ratio,
                     simple=FALSE,
                     simple.offset,
                     simple.scale,
                     simple.pch,
                     data.x,
                     col.by.row=TRUE,
                     key.in=NULL)

strip.interaction2wt(which.given, which.panel, var.name,
                     factor.levels, shingle.intervals,
                     strip.names = c(TRUE, TRUE), style = 1, ...)
```

Arguments

	`panel.interaction2wt` arguments:
	levels of x-factor
~~x~~ y	Summary value of response variable at each level of x- and trace-factors.
`subscripts`	used to get the right set of response values for the summary statistics on the off-diagonals
`responselab`	Character name of response variable, defaults to the name of the response variable.
`trace.values`	levels of trace-factor
`fun`	Summary function, defaults to `mean`
`se`	standard errors to be passed to `panel.intxplot`. Missing, logical, or a numeric vector. If `se` is missing or `FALSE`, or if `simple` is `FALSE`, then standard errors are not plotted. If `TRUE`, the standard errors are calculated from the sufficient statistics for each group as the group's standard deviation divided by the square root of the group's observation count. If a numeric vector, it is evaluated in the environment of the sufficient statistics. ,
`box.ratio`	passed to `panel.bwplot.intermediate.hh`,
`...`	extra arguments, primarily color, to be passed to `panel.bwplot.intermediate.hh`

`key.in` S-Plus only. Arguments to be passed through to the `key` for the trace-factor in each row of the display. The most likely argument is `x`, which is needed if the key is not correctly placed. Use, for example, `key.in=list(x=-3.5)` where the units are the units of the left column of panels and the value is the location where the left border of the key should be placed.

`factor.position` `"position"` attribute of factor.

`simple` logical. If `TRUE`, then simple effects are to be displayed.

`simple.offset, simple.scale` named list of offset and scale for the response and trace factors. See `interaction.positioned` for their use.

`simple.pch` Named list containing plotting characters for each level of one or more of the factors. It is used only when `simple==TRUE`. If the list contains plotting characters for both factors in the formula, the item associated with the first factor in the formula is used. If the `simple.pch` is missing, then the integers for the levels of the first factor in the model formula are used (For model formula `y ~ A*B`, the values are `seq(along=levels(A))`).

`data.x` `data.frame` containing factors from the input data.frame

`col.by.row` logical. If `TRUE` (the default), simple effects plots color the simple effects on the main diagonals in the same color as the trace levels in their row. If `FALSE`, then simple effects are colored to match the x levels in their column.

`strip.interaction2wt` arguments

`which.given, which.panel, var.name, factor.levels, shingle.intervals` see documentation for `strip.default`.

`strip.names` Force `strip.names=TRUE`

`style` force `style=1`

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

`interaction2wt`, `panel.bwplot.intermediate.hh`

Examples

```
## Not run:
tmp <- data.frame(y=rnorm(48),
                  A=factor(rep(1:2, 24)),
                  B=factor(rep(rep(1:3, each=2), 8)),
```

```
                  C=factor(rep(rep(1:4, each=6), 2)))
interaction2wt(y ~ A+B+C, data=tmp,
               key.in=list(x=-3), ## key.in is ignored by R
               xlim=c(.4, 4.5))
interaction2wt(y ~ B+C, data=tmp, key.in=list(x=-2), xlim=c(.4, 4.5))
position(tmp$B) <- c(1, 2.4, 3.8)
interaction2wt(y ~ B+C, data=tmp, key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ B+C, data=tmp, simple=TRUE,
               simple.scale=list(B=.18, C=.27),  box.ratio=.2,
               key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ C+B, data=tmp, simple=TRUE,
               simple.scale=list(B=.18, C=.27),  box.ratio=.2,
               key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ B+C, data=tmp, simple=TRUE,
               simple.scale=list(B=.18, C=.27),  box.ratio=.2,
               simple.pch=list(C=c(16,17,18,19)),
               key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ C+B, data=tmp, simple=TRUE,
               simple.scale=list(B=.18, C=.27),  box.ratio=.2,
               simple.pch=list(C=c(16,17,18,19)),
               key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ C+B, data=tmp, simple=TRUE,
               simple.scale=list(B=.18, C=.27),  box.ratio=.2,
               simple.pch=list(A=c(1:2), B=c(3:5), C=c(16,17,18,19)),
               key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ C+B, data=tmp, simple=TRUE,
               simple.scale=list(B=.18, C=.27),  box.ratio=.2,
               simple.pch=list(A=c(1:2)),
               key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ B+C, data=tmp, simple=TRUE,
               simple.scale=list(B=.18, C=.27),  box.ratio=.2,
               simple.pch=list(B=c(16,17,18)),
               key.in=list(x=-2), xlim=c(.4, 4.5),
               se=TRUE)

## End(Not run)
```

`panel.likert` *Panel functions for likert that include a stackWidth argument*

Description

`panel.barchart2` is based on `panel.barchart`
The changes are
* the heights in each horizontal stacked bar are constant.
* the widths in each vertical stacked bar are constant.
* the panel.barchart heights and widths are based on the box.width argument.
* the panel.barchart2 heights and widths when `stack==TRUE` are also based on the new `stackWidth` argument.

`panel.likert` calls `panel.barchart2`

scaling of stackWidth:
stackWidth <- stackWidth/mean(stackWidth) ## and maybe smaller with another /2

Usage

```
panel.barchart2(x, y, box.ratio = 1, box.width = box.ratio/(1 + box.ratio),
    horizontal = TRUE, origin = NULL, reference = TRUE, stack = FALSE,
    groups = NULL,
    col = if (is.null(groups)) plot.polygon$col else superpose.polygon$col,
   border = if (is.null(groups)) plot.polygon$border else superpose.polygon$border,
    lty = if (is.null(groups)) plot.polygon$lty else superpose.polygon$lty,
    lwd = if (is.null(groups)) plot.polygon$lwd else superpose.polygon$lwd,
    ..., identifier = "barchart",
          stackWidth=NULL)

panel.likert(..., horizontal=TRUE, reference.line.col="gray65")
```

Arguments

x, y, box.ratio, box.width, horizontal, origin, reference, stack, groups, col	See panel.barchart.
border, lty, lwd, identifier	See panel.barchart.
...	Extra arguments, if any, for panel.barchart.
stackWidth	Heights in each horizontal stacked bar, when stack=TRUE, are constant and specified by this argument. We recommend starting with stackWidth <- stackWidth/mean(stackWidth) and adjusting as seems appropriate.
reference.line.col	See likert.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert

---

panel.pairs.hh *Function based on S-Plus panel.pairs to add the subpanel.scales and panel.cex arguments.*

Description

Function based on S-Plus panel.pairs to add the subpanel.scales and panel.cex arguments. In R, this is an alias for panel.pairs.

Usage

```
panel.pairs.hh(x, y, z, subscripts, pscales, subpanel = panel.splom,
           varnames = dimnames(x)[[2]], ...,
           subpanel.scales, panel.cex=par()$cex)
```

Arguments

`x, y, z, subscripts, pscales, subpanel, varnames, ...`	See `splom` in S-Plus.
`subpanel.scales`	Controls the size of the tick labels in the diagonal panel.
`panel.cex`	Controls the size of the variable names in the diagonal panel.

Value

`"trellis" object`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`splom` in S-Plus.

Examples

```
if.R(s={
     longley <- data.frame(longley.x, Employed = longley.y)
     },r={
     data(longley)
     })

if.R(s=
     splom( ~ longley, pch=16, cex=.55,
           superpanel=panel.pairs.hh, subpanel.scales=list(cex=.8),
           pscales=2,
           panel.cex=.8)
     ,r=
     splom( ~ longley, pch=16,
           pscales=2,
           varname.cex=.8,
           axis.text.cex=.5)
   )
```

panel.xysplom *panel method for xysplom.*

Description

panel method for xysplom. It has a `corr` argument that is removed before sending the information on to panel.xyplot.

Usage

```
panel.xysplom(corr, ...)
```

Arguments

`corr`	logical. If `TRUE`, display the correlation and/or the regression coefficient for `lm(y ~ x)` for each panel in an additional strip label.
`...`	Remaining arguments to `panel.xyplot`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`xysplom`

partial.corr *partial correlations*

Description

The partial correlation of x and y conditioning on z is the ordinary correlation of the residuals from the regression of x on z and the regression of y on z.

Usage

```
partial.corr(vars, cond)
```

Arguments

`vars`	matrix of data.frame of all the variables to be correlated.
`cond`	matrix of data.frame of all the variables to be conditioned on.

Value

matrix of partial correlations of the numeric variables in the argument `vars` conditioning on the numeric variables in `cond`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

```
if.R(r=
partial.corr(longley[,1:3], longley[,4:6])
,s=
partial.corr(longley.x[,1:3], longley.x[,4:6])
)
```

`perspPlane` *Helper functions for regr2.plot*

Description

Helper functions for `regr2.plot`.

Usage

```
perspPlane(x, y, z, persp.out, ...)
perspFloor(x, y, z, persp.out, ...)
perspBack.wall.x(x, y, z, persp.out, ...)
perspBack.wall.y(x, y, z, persp.out, ...)
```

Arguments

`x,y,z`	Arguments to `trans3d` in R, or `perspp` in S-Plus.
`persp.out`	Result from previous call to `persp`.
`...`	Additional arguments to `persp`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`regr2.plot`

plot.hov *Homogeneity of Variance Plot*

Description

Oneway analysis of variance makes the assumption that the variances of the groups are equal. Brown and Forsyth, 1974 present the recommended test of this assumption. The Brown and Forsyth test statistic is the F statistic resulting from an ordinary one-way analysis of variance on the absolute deviations from the median. The hovPlot function graphs the components of the Brown and Forsyth test statistic.

Usage

```
hovPlot(x, data = sys.parent(), method = "bf",  ## x is a formula
        transpose = TRUE, ...)

## users will normally use the formula above and will not call the
## method directly.
hovPlot.bf(x, group,  ## x is the response variable
           y.name = deparse(substitute(x)),
           group.name = deparse(substitute(group)),
           transpose = TRUE, ...)

## users will normally use the formula above and will not call the
## panel function directly.
panel.hov(..., transpose = TRUE)
```

Arguments

x	Formula appropriate for oneway anova in hovPlot. Response variable in hovPlot.bf.
data	data.frame
method	Character string defining method. At this time the only recognized method is "bf" for the Brown-Forsyth method.
transpose	Always TRUE in R. Normally TRUE in S-Plus to force vertical boxplots.
group	factor.
y.name	name of response variable, defaults to variable name in formula.
group.name	name of factor, defaults to variable name in formula.
...	additional arguments.

Value

"trellis" object with three panels containing boxplots for each group: The observed data "y", the data with the median subtracted "y-med(y)", and the absolute deviations from the median "abs(y-med(y))" The Brown and Forsyth test statistic is the F statistic resulting from an ordinary one-way analysis of variance on the data points in the third panel.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Brown, M.~B. and Forsyth, A.~B. (1974). *Robust tests for equality of variances*. *Journal of the American Statistical Association*, 69:364–367.

See Also

aov, hov

Examples

```
data(turkey)

hov(wt.gain ~ diet, data=turkey)
hovPlot(wt.gain ~ diet, data=turkey)
```

---

plot.mmc.multicomp *MMC (Mean–mean Multiple Comparisons) plot.*

Description

MMC (Mean–mean Multiple Comparisons) plot.

Usage

```
## S3 method for class 'mmc.multicomp'
plot(x,
     xlab="contrast value",
     ylab=none$ylabel,
     focus=none$focus,
     main= main.method.phrase,
     main2=main2.method.phrase,
     main.method.phrase=
       paste("multiple comparisons of means of", ylab),
     main2.method.phrase=paste("simultaneous ",
       100*(1-none$alpha),"% confidence limits, ",
       method, " method", sep="" ),
     ry.mmc=TRUE,
     key.x=par()$usr[1]+ diff(par()$usr[1:2])/20,
     key.y=par()$usr[3]+ diff(par()$usr[3:4])/3,
     method=if (is.null(mca)) lmat$method else mca$method,
```

```
print.lmat=(!is.null(lmat)),
print.mca=(!is.null(mca) && (!print.lmat)),
iso.name=TRUE,
x.offset=0,
col.mca.signif="red",  col.mca.not.signif="black",
lty.mca.signif=1,  lty.mca.not.signif=6,
lwd.mca.signif=1,  lwd.mca.not.signif=1,
col.lmat.signif="blue", col.lmat.not.signif="black",
lty.lmat.signif=1, lty.lmat.not.signif=6,
lwd.lmat.signif=1, lwd.lmat.not.signif=1,
lty.iso=7, col.iso="darkgray", lwd.iso=1,
lty.contr0=2, col.contr0="darkgray", lwd.contr0=1,
decdigits.ybar=2,
...
)
```

Arguments

x	mmc.multicomp object
xlab	"contrast value". An alternate "" can help unclutter a figure when several MMC plots are displayed together.
ylab	name of response variable
focus	define the factor to compute contrasts of.
main, main2	main and second line of title of plot
main.method.phrase, main2.method.phrase	default expressions for title of plot
ry.mmc	range of values on the y-axis. It is similar to par("ylim"), but not the same as additional calculations are needed to maintain the isomeans grid as a square.
key.x, key.y	location of the key displayed when iso.name=FALSE.
method	method used to construct contrasts and confidence intervals. See the type argument to glht for the list.
print.lmat	logical. If TRUE, then display the user-specified contrasts.
print.mca	logical. If TRUE, then display the pair-wise contrasts.
iso.name	logical. If TRUE, label the isomeans grid with the factor levels. If FALSE, label the isomeans grid with sequential numbers and display a key relating the numbers to the factor levels.
x.offset	amount to move the vertical 0 line to the left or right to reduce overprinting of labels and plotted lines.
col.mca.signif, lty.mca.signif, lwd.mca.signif	color, line type, line width for significant pairwise contrasts.
col.mca.not.signif, lty.mca.not.signif, lwd.mca.not.signif	color, line type, line width for non-significant pairwise contrasts.
col.lmat.signif, lty.lmat.signif, lwd.lmat.signif	color, line type, line width for significant user-specified contrasts.

`col.lmat.not.signif, lty.lmat.not.signif, lwd.lmat.not.signif`
: color, line type, line width for non-significant user-specified contrasts.

`lty.iso, col.iso, lwd.iso`
: color, line type, line width for the isomeans grid.

`lty.contr0, col.contr0, lwd.contr0`
: color, line type, line width for the vertical contrast=0 line.

`decdigits.ybar`
: number of decimal digits in the left-axis labels.

`...`
: other arguments, currently ignored.

Note

`plot.mmc.multicomp` chooses sensible defaults for its many arguments. They will often need manual adjustment. The examples show several types of adjustments. We have changed the centering and scaling to avoid overprinting of label information. By default the significant contrasts are shown in a more intense color than the nonsignificant contrasts. We have an option to reduce the color intensity of the isomeans grid.

When there is overprinting of labels (a consequence of level means being close together), a tiebreaker plot may be needed. See `?MMC` for an example.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Heiberger, R.~M. and Holland, B. (2006). "Mean–mean multiple comparison displays for families of linear contrasts." *Journal of Computational and Graphical Statistics*, 15:937–955.

Hsu, J. and Peruggia, M. (1994). "Graphical representations of Tukey's multiple comparison method." *Journal of Computational and Graphical Statistics*, 3:143–161.

See Also

`mmc`, `plotMatchMMC`

Examples

```
data(catalystm)
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
summary(catalystm1.aov)

## See ?MMC to see why these contrasts are chosen
catalystm.lmat <- cbind("AB-D" =c( 1, 1, 0,-2),
                        "A-B"  =c( 1,-1, 0, 0),
                        "ABD-C"=c( 1, 1,-3, 1))
dimnames(catalystm.lmat)[[1]] <- levels(catalystm$catalyst)
```

```
catalystm.mmc <-
if.R(r={mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey"),
            focus.lmat=catalystm.lmat)}
    ,s={multicomp.mmc(catalystm1.aov, focus.lmat=catalystm.lmat,
                      plot=FALSE)}
)

## Not run:
## pairwise contrasts, default settings
plot(catalystm.mmc, print.lmat=FALSE)

## End(Not run)

## Centering, scaling, emphasize significant contrasts.
## Needed in R with 7in x 7in default plot window.
## Not needed in S-Plus with 4x3 aspect ratio of plot window.
plot(catalystm.mmc, x.offset=2.1, ry.mmc=c(50,58), print.lmat=FALSE)

## user-specified contrasts
plot(catalystm.mmc, x.offset=2.1, ry.mmc=c(50,58))

## reduce intensity of isomeans grid, number isomeans grid lines
plot(catalystm.mmc, x.offset=2.1, ry.mmc=c(50,58),
     lty.iso=2, col.iso='darkgray', iso.name=FALSE)

## both pairwise contrasts and user-specified contrasts
plot(catalystm.mmc, x.offset=2.1, ry.mmc=c(50,58), lty.iso=2,
     col.iso='darkgray', print.mca=TRUE)
```

`plot.multicomp` *Multiple comparisons plot that gives independent user control over the appearance of the significant and not significant comparisons.*

Description

Multiple comparisons plot that gives independent user control over the appearance of the significant and not significant comparisons. In R, both `plot.multicomp` `plot.multicomp.hh` coerce their argument to an `"glht"` object and plots that with the appropriate `plot` method. In R, `plot.multicomp.adjusted` replaces the bounds calculated by `multcomp:::confint.glht` with bounds based on a common standard error for a set of anova tables that are partitioned for the simple effects on an analysis conditioned on the levels of one of the factors. In S-Plus, `plot.multicomp.hh` augments the standard `plot.multicomp` to give additional user arguments to control the appearance of the plot.

`plotMatchMMC` uses the `plot.multicomp.hh` code. `plotMatchMMC` must immediately follow a plot of an `mmc.multicomp` object and is applied to either the `$mca` or `$lmat` component of the `mmc.multicomp` object. `plotMatchMMC` is used as a tiebreaker plot for the `MMC` plot. `plotMatchMMC` matches the horizontal scaling of the `MMC` plot and displays the individual contrasts in the same order as the `MMC` plot. See `mmc` for examples.

Usage

```
## S3 method for class 'multicomp'
plot(x, ...) ## R only

## S3 method for class 'multicomp.hh'
plot(x, ylabel = x$ylabel, href = 0, uniform = TRUE,
           plt.in = c(0.2, 0.9, 0.1, 0.9),
           x.label.adj=1,
           xrange.include=href,
           xlim,
           comparisons.per.page=21,
           col.signif=1, col.not.signif=1,
           lty.signif=4, lty.not.signif=4,
           lwd.signif=1, lwd.not.signif=1,
           ...,
           xlabel.print=TRUE, y.axis.side=2, ylabel.inside=FALSE)

plotMatchMMC(x, ...,
             xlabel.print=FALSE,
             cex.axis=par()$cex.axis,
             col.signif='red', main="",
             ylabel.inside=FALSE,
             y.axis.side=4,
             adjusted=FALSE)
```

Arguments

x	A "multicomp" object. plotMatchMMC will also accept a mmc.multicomp object. It will use the lmat component if there is one, otherwise it will use the mca component.
ylabel	Y label on graph.
y.axis.side	Y labels are on the left by default when plotting a "multicomp" object. We move them to the right when matching the x-axis of an MMC plot.
...	other arguments to plot.multicomp.
ylabel.inside	Logical value, if FALSE (the default), the plotMatchMMC right-axis labels are in the margin. If TRUE, the right-axis labels are in the figure area. Setting the argument to TRUE makes sense when plotting the lmat component of an mmc.multicomp object.
href	reference line for the intervals. The default is 0. S-Plus only.
xrange.include	xlim will be extended to include these values. S-Plus only.
uniform	S-Plus only. Logical value, if TRUE and the plots fill more than one page, the scale will be uniform across pages.
plt.in	S-Plus only. Value for par("plt") to make better use of the space on the plotting page.

`x.label.adj` S-Plus only. This is the `par("adj")` applied to the x-location of the y.labels on the `multicomp` plot.

`xlim` x-range of the plot.

`comparisons.per.page`
The default S-Plus `plot.multicomp` hardwires this to 21, which allows for all pairwise comparisons of 7 levels taken 2 at a time. The HH `plot.multicomp` makes it a variable. Use it together with `plt.in` to make better use of the space on the plot. S-Plus only.

`lty.signif, lwd.signif`
Line type, and line width for significant comparisons. S-Plus only.

`col.signif` Color for significant comparisons. S-Plus only for `plot.multicomp`. Both R and S-Plus for `plotMatchMMC`.

`col.not.signif, lty.not.signif, lwd.not.signif`
Color, line type, and line width for non-significant comparisons. S-Plus only.

`xlabel.print` logical. When `TRUE`, the caption under the plot is printed. When `FALSE`, the caption under the plot is not printed. It is helpful to set this to `FALSE` when the `multicomp` plot is used as a tiebreaker plot for the MMC plot. S-Plus only.

`cex.axis` `cex` for axis ticklabels.

`main` Main title for plot.

`adjusted` Logical. When `TRUE`, `HH:::plot.multicomp.adjusted` is used to replace the standard confidence bounds calculated by `multcomp:::confint.glht`, with bounds calculated by `as.multicomp.glht` with a rescaled critical value based on rescaling the standard error. This rescaling is used to construct a common standard error for a set of anova tables that are partitioned for the simple effects on an analysis conditioned on the levels of one of the factors. See the `clover.commonstrMS.clov.mmc` example in file `hh("scripts/Ch12-tway.r")`.

Value

`plot.multicomp` plots a `"multicomp"` object. In S-Plus, this masks the standard `plot.multicomp` in order to provide additional arguments for controlling the appearance. It defaults to the standard appearance. In R, it coerces its argument to a `"glht"` object and plots that with the appropriate `plot` method.

Note

The multiple comparisons calculations in R and S-Plus use completely different libraries.

Multiple comparisons in R are based on `glht`. Multiple comparisons in S-Plus are based on `multicomp`. The MMC plot in the HH library is the same in both systems. The details of getting the plot differ.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Heiberger, R. M. and Holland, B. (2006). "Mean–mean multiple comparison displays for families of linear contrasts." *Journal of Computational and Graphical Statistics*, 15:937–955.

See Also

mmc in both languages,

glht.

Examples

```
## data and ANOVA
data(catalystm)

catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
summary(catalystm1.aov)

catalystm.mca <-
if.R(r=glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey")),
     s=multicomp(catalystm1.aov, plot=FALSE))
if.R(s=plot(catalystm.mca),
     r=plot(confint(catalystm.mca, calpha=qtukey(.95, 4, 12)/sqrt(2))))
       ## calpha is strongly recommended in R with a large number of levels
       ## See ?MMC for details.
```

`position` *Find or assign the implied position for graphing the levels of a factor. A new class "positioned", which inherits from "ordered" and "factor", is defined.*

Description

The default values for plotting a factor `x` are the integers `1:length(levels(x))`. These functions provide a way of specifying alternate plotting locations for the levels.

Usage

```
position(x)

position(x) <- value

## S3 method for class 'positioned'
is.numeric(x, ...)
## S3 method for class 'positioned'
```

```
as.numeric(x, ...)
## S3 method for class 'positioned'
x[..., drop=FALSE]
## S3 method for class 'positioned'
is.na(x)
as.positioned(x)
as.position(x)
is.positioned(x)
positioned(x, ..., value)
## S3 method for class 'positioned'
print(x, ...)
## S3 method for class 'positioned'
unique(x, incomparables = FALSE, ...)
unpositioned(x, ...)
```

Arguments

x	numeric vector or factor
value	numerical values to be associated with levels(x). The length(value) must equal length(levels(as.factor(x))).
...	other arguments.
drop	See Extract.
incomparables	See unique.

Value

position(x) <- value first forces its argument to be an ordered factor and then assigns the value to the "position" attribute of the ordered factor. The result is assigned class "positioned" and returned.

position(x) returns the position values associated with levels(x). If x is a positioned factor, then the "position" attribute is returned. If x is a factor, then the integers 1:length(levels(x)) are returned. For anything else, as.numeric(x) is returned.

as.position(x) returns a numeric vector the length of the original vector. If x inherits from "factor", then the values in the vector are the values in position(x) subscripted by the levels of the factor. If x is numeric, then x itself is returned.

unpositioned(x) removes the "position" attribute and removes the "positioned" value from the the oldClass of the object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

panel.interaction2wt,

factor.

Examples

```
## ordered with character levels defaults to
## integer position of specified levels
tmp <- ordered(c("mm","cm","m","m","mm","cm"),
               levels=c("mm","cm","m")) ## size order
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## position is assigned to ordered in specified order
tmp <- ordered(c("cm","mm","m","m","mm","cm"),
               levels=c("mm","cm","m")) ## size order
levels(tmp)
position(tmp) <- c(-3, -2, 0) ## log10 assigned in size order
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## numeric stays numeric
tmp <- c(0.010, 0.001, 1.000, 1.000, 0.001, 0.010)
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## factor with numeric levels, position is integer position in size order
tmp <- factor(c(0.010, 0.001, 1.000, 1.000, 0.001, 0.010))
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
```

```
unique(tmp)

## ordered with numeric levels, position is numeric value in size order
tmp <- ordered(c(0.010, 0.001, 1.000, 1.000, 0.001, 0.010))
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)


## factor with numeric levels
## position is assigned in size order
tmp <- factor(c(0.010, 0.001, 1.000, 1.000, 0.001, 0.010))
levels(tmp)
position(tmp) <- c(-3, -2, 0) ## log10 assigned in size order
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)



## boxplots coded by week
tmp <- data.frame(Y=rnorm(40, rep(c(20,25,15,22), 10), 5),
                  week=ordered(rep(1:4, 10)))
position(tmp$week) <- c(1, 2, 4, 8)

if.R(r=
       bwplot(Y ~ week, horizontal=FALSE,
              scales=list(x=list(limits=c(0,9),
                          at=position(tmp$week),
                          labels=position(tmp$week))),
              data=tmp, panel=panel.bwplot.intermediate.hh)
    ,s=
       t(bwplot(week ~ Y, at=position(tmp$week),
                scales=list(y=list(limits=c(0,9),
                            at=position(tmp$week), labels=position(tmp$week))),
                data=tmp, panel=panel.bwplot.intermediate.hh))
    )


#### You probably don't want to use the next two examples.
#### You need to be aware of their behavior.
```

```
##
## factor with character levels defaults to
## integer position of sorted levels.
## you probably DON'T want to do this!
tmp <- factor(c("cm","mm","m","m","mm","cm")) ## default alphabetic order
tmp
as.numeric(tmp)
levels(tmp)   ## you probably DON'T want to do this!
position(tmp) ## you probably DON'T want to do this!
as.numeric(tmp)
##
## position is assigned to factor in default alphabetic order.
## you probably DON'T want to do this!
tmp <- factor(c("cm","mm","m","m","mm","cm"))
levels(tmp)
position(tmp) <- c(-3, -2, 0) ## assigned in default alphabetic order
tmp
as.numeric(tmp)
levels(tmp)   ## you probably DON'T want to do this!
position(tmp) ## you probably DON'T want to do this!
as.numeric(tmp)
```

`positioned-class` *Class "positioned", extends "ordered" to specify the position for graphing the levels of a factor.*

Description

The default values for plotting a factor `x` are the integers `1:length(levels(x))`. This class and its functions provide a way of specifying alternate plotting locations for the levels.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class `"ordered"`, directly. Class `"factor"`, by class "ordered", distance 2. Class `"oldClass"`, by class "ordered", distance 3.

Methods

No methods defined with class "positioned" in the signature. S3-type methods are `"[.positioned"`, `as.double.positioned`, `as.numeric.positioned`, `as.positioned`, `is.numeric.positioned`, `is.positioned`, `positioned`, `print.positioned`, `unique.positioned`.
Although `interaction.positioned` should be a method, it isn't because `interaction` is not a generic and can't easily be made one since the name `interaction.plot` conflicts.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

position.

---

print.tsdiagplot *Print a "tsdiagplot" object.*

Description

Print a "tsdiagplot" object.

Usage

```
## S3 method for class 'tsdiagplot'
print(x, ...)
print1.tsdiagplot(x)
print2.tsdiagplot(x)
```

Arguments

x	a "tsdiagplot" object
...	Optional arguments to print. The only argument that is used is pages. If pages is not used or pages==1, then use print1.tsdiagplot. If pages!=1, then use print2.tsdiagplot.

Details

A "tsdiagplot" object is a collection of several "trellis" objects. We provide two options for printing them.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

tsdiagplot

print.TwoTrellisColumns

Print two conformable trellis plots in adjacent columns with user control of widths.

Description

Print two conformable trellis plots in adjacent columns with user control of widths. Left y ticklabels and left.strip are removed from the right-hand plot.

Usage

```
as.TwoTrellisColumns5(left,  ## left  is the left trellis object
                      right, ## right is the right trellis object
                      ## Both left and right must have identical
                      ## settings for number and size of vertical panels,
                      ## left-axis labels, number of lines in main, sub, legend.
                      ...,
                      pw=c(.3, .30, .01, .30, .09),
                      px=list(
                        LL=c(0, pwc[1]),
                        LP=pwc[1:2],
                        ML=pwc[2:3],
                        RP=pwc[3:4],
                        RL=pwc[4:5]),
                      pwc=cumsum(pw),
                      strip.left=TRUE,
                      y.tck=c(0,0)
                      )

## S3 method for class 'TwoTrellisColumns5'
print(x, px=attr(x, "px"), ...)

leftLabels.trellis(x)
rightLabels.trellis(x)
panelOnly.trellis(x, strip.left=FALSE, y.tck=0)
mainSubLegend.trellis(x)
emptyRightAxis(x)
```

Arguments

left, right	Conformable "trellis" objects. Both must have the identical settings for number and size of vertical panels, left-axis labels, number of lines in main, sub, legend.
x	"trellis" object.

px	These are used x-values used in the position argument of the print.trellis function. The default (constructed from the pw argument) makes the Left and Right panels the same width and the Middle containing the y-axis is given the remainder. Overlapping is permitted. The appearance depends on the width of the graphics device.
pw, pwc	pw vector of five positive numbers that sum to 1. These are the relative widths of the five sections of the result: LeftLabels, LeftPanel, MainSubLegend, RightPanel, RightLabels. pwc is the cumulative sum of pw. pwc is expanded in the px argument to the x values used in the position argument of the print.trellis function.
strip.left	See barchart.
y.tck	A vector of one or two numeric values. This will be used as the y tck value for the right column of panels. See 'tck' in barchart for details.
...	Other arguments are ignored.

Details

as.TwoTrellisColumns5 constructs a "TwoTrellisColumns5" object, which is a list of five trellis objects named "LL", "LP","ML", "RP", "RL". LL is the left labels from the left input object. LP is the panels from the left input object. ML is the middle labels from the left object; these are the main title, sub title, and legend. RP is the panels from the right input object. RL is the right labels from the right input object.

print.TwoTrellisColumns5 is a print method for a "TwoTrellisColumns5" object. It takes left-to-right positioning information from the "px" attribute of its argument x or from an input argument. The numbers are used as the "x" information for the position argument to the print.trellis method.

emptyLeftAxis,leftLabels.trellis,rightLabels.trellis,panelOnly.trellis, mainSubLegend.trellis,emptyLeftStrip,emptyRightAxis are functions which blank out the various components of the trellis argument and retains their vertical spacing.

Value

A "TwoTrellisColumns5" object, consisting of a list containing the constructed left, middle, and right trellis objects, and an attribute containing the px value.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert for the details on the motivating example.

Examples

```
library(HH)
```

```
## These are based on the Professional Challenges example in ?likert
data(ProfChal)
levels(ProfChal$Subtable)[6] <- "Prof Recog" ## reduce length of label

## initial ordering of Question factor
PCC <- likert(Question ~ . | Subtable, ProfChal, ylab=NULL,
              rightAxis=TRUE,
              layout=c(1,6),
              strip=FALSE,
              strip.left=strip.custom(bg="gray97"),
              par.strip.text=list(cex=.7),
              scales=list(y=list(relation="free")),
              main="Is your job professionally challenging?")

## initial ordering of Question factor
PCP <- likert(Question ~ . | Subtable, ProfChal, ylab=NULL,
              as.percent=TRUE,
              layout=c(1,6),
              strip=FALSE,
              strip.left=strip.custom(bg="gray97"),
              par.strip.text=list(cex=.7),
              scales=list(y=list(relation="free")),
              main="Is your job professionally challenging?")

## Not run:
## default equal widths of the two panels
as.TwoTrellisColumns5(PCP, PCC)  ## 11in x 7in

## make left panel twice as wide as right panel
as.TwoTrellisColumns5(PCP, PCC, pw=c(.3, .4, .01, .2, .09))  ## 11in x 7in
##                                  --------------------    ## sum to 1.00

## make left panel twice as wide as right panel, and control position of main and legend
as.TwoTrellisColumns5(PCP, PCC,  ## 11in x 7in
                      px=list(
                        LL=c(.00,  .30),
                        LP=c(.30,  .70),
                        ML=c(.60,  .61),  ## arbitrary,
                                          ## visually center the labels and legend
                        RP=c(.71,  .91),
                        RL=c(.91, 1.00)))

## End(Not run)

## Size that works in default 7x7 window.  7x7 is not recommended for
## this example because most of the space is used for labeling and not
## much for the panels containing the data.  Use the px values for the
## 11x7 illustrated above in the dontrun section.

as.TwoTrellisColumns5(PCP, PCC,  ## 7in x 7in
                      px=list(
                        LL=c(.00,  .50),
                        LP=c(.50,  .70),
```

```
                        ML=c(.50,  .51),  ## arbitrary,
                                          ## visually center the labels and legend
                        RP=c(.71,  .87),
                        RL=c(.87, 1.00)))

## Ordering the rows by the lengths of the positive bars and also
## put percents and counts on the same plot.
## The easiest way is to use the LikertPercentCountColumns function:

LikertPercentCountColumns(Question ~ . | Subtable, ProfChal,
                          layout=c(1,6), scales=list(y=list(relation="free")),
                          ylab=NULL, between=list(y=0),
                          strip.left=strip.custom(bg="gray97"), strip=FALSE,
                          par.strip.text=list(cex=.7),
                          positive.order=TRUE,
                          main="Is your job professionally challenging?")

## Not run:

## Ordering the rows by the lengths of the positive bars and also
## putting percents and counts on the same plot requires coordination.
## The easiest way is to order the original tables of counts by the
## order of the percent plot.

percentPlot <- likert(Question ~ . | Subtable, ProfChal,
                      as.percent=TRUE,
                      layout=c(1,6), scales=list(y=list(relation="free")),
                      ylab=NULL, between=list(y=0),
                      strip.left=strip.custom(bg="gray97"), strip=FALSE,
                      par.strip.text=list(cex=.7),
                      positive.order=TRUE,
                      main="Is your job professionally challenging?")
## percentPlot
pct.order <- percentPlot$y.limits[[1]]

ProfChal2 <- ProfChal
ProfChal2$Question <- factor(ProfChal2$Question, levels=rev(pct.order))

countPlot   <- likert(Question ~ . | Subtable, ProfChal2,
                      layout=c(1,6),
                      rightAxis=TRUE,
                      scales=list(y=list(relation="free"),
                                  x=list(at=c(0, 250, 500))),
                      ylab=NULL, between=list(y=0),
                      strip.left=strip.custom(bg="gray97"), strip=FALSE,
                      par.strip.text=list(cex=.7),
                      main="Is your job professionally challenging?")
## countPlot
levels(ProfChal$Subtable)[6] <-
      "Attitude\ntoward\nProfessional\nRecognition" ## Restore original label

## Size that works in default 7x7 window.  7x7 is not recommended for
## this example because most of the space is used for labeling and not
```

```
## much for the panels containing the data.  Use the px values for the
## 11x7 illustrated above in the dontrun section.

as.TwoTrellisColumns5(percentPlot, countPlot,  ## 7in x 7in
                      px=list(
                        LL=c(.00,  .50),
                        LP=c(.50,  .70),
                        ML=c(.50,  .51),  ## arbitrary,
                                          ## visually center the labels and legend
                        RP=c(.71,  .87),
                        RL=c(.87, 1.00)))

## End(Not run)
```

---

push.vp.hh *push and pop a grid viewport, turn clipping off, change scale.*

Description

push and pop a grid viewport, turn clipping off, change scale.

Usage

```
push.vp.hh(scale = 100)
pop.vp.hh()
```

Arguments

scale	argument to the unit function.

Details

Used in panel.cartesian to ease labeling the rows and columns of a scatterplot matrix.

Value

An object of class "unit".

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

viewport, unit, panel.cartesian

`pyramidLikert` *Print a Likert plot as a Population Triangle*

Description

Prints a likert plot in the traditional format for a population pyramid, with the Left and Right sides in separate panels, with the x-tick marks on the left side made positive, and with the y-axis in the Middle.

Usage

```
## S3 method for class 'pyramidLikert'
print(x, ...,
                      panel.width=.48,
                      px=list(
                        L=c(0, panel.width),
                        R=c(1-panel.width, 1),
                        M=c(panel.width, 1-panel.width)),
                     keepLegend=(length(x$legend$bottom$args$text) > 2),
                     xlab.top=list(
                       L=list(x$legend$bottom$args$text[1]),
                       R=list(x$legend$bottom$args$text[2]),
                       M=list(x$ylab, just=1)))

as.pyramidLikert(x, ...,
                   panel.width=.48,
                   px=list(
                     L=c(0, panel.width),
                     R=c(1-panel.width, 1),
                     M=c(panel.width, 1-panel.width)),
                   keepLegend=(length(x$legend$bottom$args$text) > 2),
                   xlab.top=list(
                     L=list(x$legend$bottom$args$text[1]),
                     R=list(x$legend$bottom$args$text[2]),
                     M=list(x$ylab, just=1)))
```

Arguments

`x`	a single-panel `'trellis'` object.
`...`	Other arguments are ignored.
`panel.width`	Numeric scalar between 0 and 0.5. Common width of left and right panels. The default value .48 value works well for the `USAge.table` example. This number is expanded in the `px` argument to the `x` values used in the `position` argument of the `print.trellis` function.
`px`	`x` values used in the position argument of the `print.trellis` function. The default makes the Left and Right panels the same width and the Middle containing the y-axis is given the remainder.

keepLegend — If TRUE and x contains a bottom legend, then it is printed along with the Middle section containing the y-axis. If FALSE or there is no bottom legend, then the bottom legend is not printed.

xlab.top — A vector of three labels. The default is designed for a population triangle with two levels (usually, Male on one side and Female on the other). The Left and Right labels are taken from the first two labels in the legend. The Middle value is the variable name for the y-axis.

Details

This is a print method for population triangles. It is designed for a likert plot with one left-side level and one right-side level. It works for any single-panel "trellis" object, in the sense that it produces a plot.

Value

The input argument x.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert

Examples

```
library(HH)
data(USAge.table)  ## from latticeExtra
USA79 <- USAge.table[75:1, 2:1, "1979"]/1000000
PL <- plot(as.likert(USA79),
                     main="Population of United States 1979 (ages 0-74)",
                     xlab="Count in Millions",
                     ylab="Age",
                     scales=list(
                       y=list(
                         limits=c(0,77),
                         at=seq(1,76,5),
                         labels=seq(0,75,5),
                         tck=.5))
                      )
PL
as.pyramidLikert(PL)

likert(USAge.table[75:1, 2:1, c("1939","1959","1979")]/1000000,
       main="Population of United States 1939,1959,1979 (ages 0-74)",
       sub="Look for the Baby Boom",
       xlab="Count in Millions",
       ylab="Age",
       scales=list(
```

```
          y=list(
            limits=c(0,77),
            at=seq(1,76,5),
            labels=seq(0,75,5),
            tck=.5)),
        strip.left=FALSE, strip=TRUE,
        layout=c(3,1), between=list(x=.5))


## For additional examples, see demo(PoorChildren, package="HH")
```

`regr1.plot` *plot x and y, with optional straight line fit and display of squared residuals*

Description

Plot `x` and `y`, with optional fitted line and display of squared residuals. By default the least squares line is calculated and used. Any other straight line can be specified by placing its coefficients in `coef.model`. Any other fitted model can be calculated by specifying the `model` argument. Any other function of one variable can be specified in the `alt.function` argument. At most one of the arguments `model`, `coef.model`, `alt.function` can be specified.

Usage

```
regr1.plot(x, y,
           model=lm(y~x),
           coef.model,
           alt.function,
           main="put a useful title here",
           xlab=deparse(substitute(x)),
           ylab=deparse(substitute(y)),
           jitter.x=FALSE,
           resid.plot=FALSE,
           points.yhat=TRUE,
           pch=16,
           ..., length.x.set=51,
           x.name,
           pch.yhat=16,
           cex.yhat=par()$cex*.7,
           err=-1)
```

Arguments

x	x variable
y	y variable

model	Defaults to the simple linear model lm(y ~ x). Any model object with one x variable, such as the quadratic lm(y ~ x + I(x^2)) can be used.
coef.model	Defaults to the coefficients of the model argument. Other intercept and slope coefficients for a straight line (for example, c(3,5)) can be entered to illustrate the sense in which they are not "least squares".
alt.function	Any function of a single argument can be placed here. For example, alt.function=function(x) {3 + 2*x + 3*x^2}. All coefficients must be specified.
main, xlab, ylab	arguments to plot.
jitter.x	logical. If TRUE, the x is jittered before plotting. Jittering is often helpful when there are multiple y-values at the same level of x.
resid.plot	If FALSE, then do not plot the residuals. If "square", then call resid.squares to plot the squared residuals. If TRUE (or anything else), then call resid.squares to plot straight lines for the residuals.
points.yhat	logical. If TRUE, the predicted values are plotted.
...	other arguments.
length.x.set	number of points used to plot the predicted values.
x.name	If the model argument used a different name for the independent variable, you might need to specify it.
pch	Plotting character for the observed points.
pch.yhat	Plotting character for the fitted points.
cex.yhat	cex for the fitted points.
err	The default -1 suppresses warnings about out of bound points.

Note

This plot is designed as a pedagogical example for introductory courses. When resid.plot=="square", then we actually see the set of squares for which the sum of their areas is minimized by the method of "least squares".

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Smith, W. and Gonick, L. (1993). *The Cartoon Guide to Statistics*. HarperCollins.

See Also

resid.squares

Examples

```
data(hardness)

## linear and quadratic regressions
hardness.lin.lm  <- lm(hardness ~ density,                data=hardness)
hardness.quad.lm <- lm(hardness ~ density + I(density^2), data=hardness)

anova(hardness.quad.lm)  ## quadratic term has very low p-value

par(mfrow=c(1,2))

regr1.plot(hardness$density, hardness$hardness,
           resid.plot="square",
           main="squared residuals for linear fit",
           xlab="density", ylab="hardness",
           points.yhat=FALSE,
           xlim=c(20,95), ylim=c(0,3400))

regr1.plot(hardness$density, hardness$hardness,
           model=hardness.quad.lm,
           resid.plot="square",
           main="squared residuals for quadratic fit",
           xlab="density", ylab="hardness",
           points.yhat=FALSE,
           xlim=c(20,95), ylim=c(0,3400))

par(mfrow=c(1,1))
```

`regr2.plot` *3D plot of z against x and y, with regression plane fit and display of squared residuals.*

Description

3D plot of z against x and y, with regression plane fit and display of squared residuals.

Usage

```
regr2.plot(x, y, z,
           main.in="put a useful title here",
           resid.plot=FALSE,
           plot.base.plane=TRUE,
           plot.back.planes=TRUE,
           plot.base.points=FALSE,
           eye=NULL,                    ## S-Plus
           theta=0, phi=15, r=sqrt(3), ticktype="detailed", ## R
           ...)
```

Arguments

x,y,z	See persp.
main.in	main title for plot.
resid.plot	Argument to resid.squares.
plot.base.plane, plot.back.planes, plot.base.points	Should these items be plotted?
eye	S-Plus only. See persp.
theta, phi, r, ticktype	R only. See persp.
...	Other arguments to persp.

Note

This plot is designed as a pedagogical example for introductory courses. When resid.plot=="square", then we actually see the set of squares for which the sum of their areas is minimized by the method of "least squares".

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Smith, W. and Gonick, L. (1993). *The Cartoon Guide to Statistics*. HarperCollins.

See Also

resid.squares, regr1.plot

Examples

```
data(fat)
regr2.plot(fat[,"abdomin"], xlab="abdomin",
           fat[,"biceps"],  ylab="biceps",
           fat[,"bodyfat"], zlab="bodyfat",
           resid.plot="square",
           eye=c(335.5, 115.65, 171.9),   ## used only in S-Plus
           theta=140, phi=35, r=sqrt(15), ## used only in R
           box=is.R(),
           plot.back.planes=FALSE,
           main="Least-squares with two X-variables")
```

`resid.squares` *plot squared residuals in inches to match the y-dimension*

Description

plot squared residuals in inches to match the y-dimension

Usage

```
resid.squares(x, y, y.hat, resid.plot = "square", ...)
```

Arguments

`x`	x values
`y`	observed y values
`y.hat`	predicted y values
`resid.plot`	If `"square"`, then plot the squared residuals. If `TRUE` (or anything else), then plot straight lines for the residuals.
`...`	Other graphics arguments.

Details

The goal is to get real squares on the screen or paper. The trick is to play games with the aspect ratio. We find the number of inches that each vertical residual occupies. We then find the number of x-units that corresponds to, and plot a rectangle with height=height in the y-data units and with width=the number of x-units that we just calculated.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

`regr1.plot`

Examples

```
data(hardness)

hardness.lin.lm  <- lm(hardness ~ density, data=hardness)

plot(hardness ~ density, data=hardness, xlim=c(22,73), ylim=c(0,3400))
abline(hardness.lin.lm)
resid.squares(hardness$density, hardness$hardness,
              predict(hardness.lin.lm))

plot(hardness ~ density, data=hardness, xlim=c(22,73), ylim=c(0,3400))
abline(hardness.lin.lm)
resid.squares(hardness$density, hardness$hardness,
              predict(hardness.lin.lm), resid.plot = "line")
```

---

residual.plots *Residual plots for a linear model.*

Description

Residual plots for a linear model. Four sets of plots are produced: (1) response against each of the predictor variables, (2) residuals against each of the predictor variables, (3) partial residuals for each predictor against that predictor ("partial residuals plots", and (4) partial residuals against the residuals of each predictor regressed on the other predictors ("added variable plots").

Usage

```
residual.plots(lm.object, X=dft$x,
               layout=c(dim(X)[2],1),
               par.strip.text=list(cex=.8),
               scales.cex=.6,
               na.action=na.pass,
               y.relation="free",
               ...)
```

Arguments

lm.object	An object inheriting from "lm". It may be necessary for the lm.object to be constructed with arguments x=TRUE, y=TRUE.
X	The x matrix of predictor variables used in the linear model lm.object.
layout, par.strip.text	trellis or lattice arguments.
scales.cex	cex argument forwarded to the scales argument of xyplot.
na.action	A function to filter missing data. See lm.
y.relation	See relation in the discussion of the scales argument in xyplot.
...	Other arguments for xysplom or xyplot.

Value

A list of four trellis objects, one for each of the four sets of plots. The objects are named `"y.X"`, `"res.X"` `"pres.X"`, `"pres.Xj"`. The default "printing" of the result will produce four pages of plots, one set per page. They are often easier to read when all four sets appear as separate rows on one page (this usually requires an oversize device), or two rows are printed on each of two pages.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

Examples

```
if.R(s={
     longley <- data.frame(longley.x, Employed = longley.y)
     },r={
     data(longley)
     })

longley.lm <- lm( Employed ~ . , data=longley, x=TRUE, y=TRUE)
## 'x=TRUE, y=TRUE' are needed to pass the S-Plus CMD check.
## They may be needed if residual.plots() is inside a nested set of
## function calls.

tmp <- residual.plots(longley.lm)

## print two rows per page
print(tmp[[1]], position=c(0, 0.5, 1, 1.0), more=TRUE)
print(tmp[[2]], position=c(0, 0.0, 1, 0.5), more=FALSE)
print(tmp[[3]], position=c(0, 0.5, 1, 1.0), more=TRUE)
print(tmp[[4]], position=c(0, 0.0, 1, 0.5), more=FALSE)
```

`ResizeEtc` *Display multiple independent trellis objects on the same coordinated scale.*

Description

This function is a wrapper for several of the functions in the `latticeExtra` package.

Usage

```
ResizeEtc(c.list,
          condlevelsName,
          x.same, y.same,
          layout,
          strip=TRUE,
          strip.left=TRUE,
          strip.values, strip.left.values,
          strip.par, strip.left.par,  ## only the second is effective
                                      ## when both are specified
          resize.height, resize.width,
          main,
          ...)
```

Arguments

`c.list`	combination of two or more trellis objects from `c.trellis`. If `c.list` has names, the names will appear in the strips.
`condlevelsName`	Name of the dimname of the items in the `c.list`.
`x.same, y.same`	If `TRUE`, force all panels to have the same `x.limits` or `y.limits`.
`layout`	Standard lattice `layout` argument.
`strip, strip.left`	standard lattice arguments described in `barchart`.
`strip.values, strip.left.values`	strip names for the panels. Only the second is effective when both are specified.
`strip.par, strip.left.par`	`par.strip.text`. Only the second is effective when both are specified.
`resize.height, resize.width`	`h` and `w` arguments to `resizePanels`.
`main`	Main title for resulting combined plot.
`...`	Other arguments to `barchart`.

Value

`"trellis"` object combining each of the individual plots in the `c.list` argument according to the specifications in the other arguments.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`c.trellis`, `plot.likert`

Examples

```
## see the examples in ?HH:::plot.likert

require(grid)
require(lattice)
require(latticeExtra)
require(HH)

## This is the same example as in ?HH:::plot.likert
## Here, it is done with explicit use of ResizeEtc.

data(ProfChal)
tmp <- data.matrix(ProfChal[,1:5])
rownames(tmp) <- ProfChal$Question

AA <- likert(tmp[1,], box.width=unit(.4,"cm"), positive.order=TRUE)
BB <- likert(tmp[2:6,], box.width=unit(.4,"cm"), positive.order=TRUE)
CC <- likert(tmp[7:10,], box.width=unit(.4,"cm"), positive.order=TRUE)
DD <- likert(tmp[11:12,], box.width=unit(.4,"cm"), positive.order=TRUE)
EE <- likert(tmp[13:14,], box.width=unit(.4,"cm"), positive.order=TRUE)
FF <- likert(tmp[15:16,], box.width=unit(.4,"cm"), positive.order=TRUE)

BB

## print(AA, more=TRUE,  split=c(1,1,3,2))
## print(BB, more=TRUE,  split=c(2,1,3,2))
## print(CC, more=TRUE,  split=c(3,1,3,2))
## print(DD, more=TRUE,  split=c(1,2,3,2))
## print(EE, more=TRUE,  split=c(2,2,3,2))
## print(FF, more=FALSE, split=c(3,2,3,2))

ResizeEtc(c.list=c(AA,BB,CC,DD,EE,FF),
          layout=c(1,6), main="Not yet good enough")

Group <- levels(ProfChal$Subtable)

ResizeEtc(c.list=c(AA,BB,CC,DD,EE,FF),
          condlevelsName='Group',
          x.same=TRUE,
          layout=c(1,6),
          strip.left.values=Group,
          strip.left.par=list(cex=.7, lines=5),
          resize.height=c(1,5,4,2,2,2)+.5,
          main=list("Is your job professionally challenging?", x=unit(.65, "npc")))
```

`ResizeEtc.likertPlot` *Display multiple independent trellis objects, representing likert plots, on the same coordinated scale.*

Description

This is a method for ResizeEtc intended for use with "likert" plots that allows positive values on the negative side of the axis.

Usage

```
## S3 method for class 'likertPlot'
ResizeEtc(c.list,
                    x,
                    x.pl.nonames,
                    horizontal,
                    ...)
```

Arguments

c.list	combination of two or more trellis objects from c.trellis. If c.list has names, the names will appear in the strips.
x	List of two-dimensional objects with the same columns. See plot.likert.list for details.
x.pl.nonames	List of "likert" objects corresponding to the items in argument x. The items in x.pl.nonames are unnamed.
horizontal	Standard argument for barchart.
...	Other arguments to ResizeEtc.

Value

The result is a "trellis" object. It is essentially the same object returned by ResizeEtc with possibly adjusted x tick-labels to put positive labels on the negative axis. If horizontal==FALSE, then the possible adjusted labels are the y tick-labels.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

ResizeEtc, likert.

---

seqplot *Time series plot.*

Description

Time series plot.

Usage

```
seqplot(xts, ...)

## Default S3 method:
seqplot(xts,
        pch.seq=letters,
        groups=as.numeric(cycle(xts)),
        a=NULL, b=NULL, h=NULL, v=NULL,
        ylab=deparse(substitute(xts)),
        xlab=ifelse(is.null(units(xts)),"Time",units(xts)),
        lwd=0, lty=c(1,3),
        type="b",
        ...)

## S3 method for class 'rts'
seqplot(xts, pch.seq=letters, groups=as.numeric(cycle(xts)),
        x.at=pretty(time(xts)[groups==min(groups)]),
        x.labels,
        ylab=deparse(substitute(xts)),
        ...)

## S3 method for class 'ts'
seqplot(xts, pch.seq=letters, groups=as.numeric(cycle(xts)),
        x.at=pretty(time(xts)[groups==min(groups)]),
        x.labels,
        ylab=deparse(substitute(xts)),
        ...)
```

Arguments

xts	Time series

pch.seq	sequence of pch characters for use with the time series. The characters repeat over the cycle of the series.
groups	Numeric vector used to choose the plotting characters over cycles.
a, b, h, v	Arguments to panel.abline.
ylab, xlab, lwd, lty, type	stndard trellis arguments.
x.at, x.labels	shortcut for scales=list(x=list(at=x.at, labels=x.labels))
...	Additional arguments to xyplot.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

tsacfplots

Examples

```
seqplot(co2)
```

---

seqplot.forecast *seqplot with confidence bands for the forecast region.*

Description

seqplot with confidence bands for the forecast region.

Usage

```
seqplotForecast(xts, forecast, multiplier = 1.96,
                series = deparse(substitute(observed)), ylim,
                CI.percent=round((1-2*(1-pnorm(multiplier)))*100,2),
                main = paste(
                series, " with forecast + ",
                CI.percent, "% CI", sep=""),
                xlab=NULL, ylab=NULL,
                ...) ## x.at, xlim
```

Arguments

xts	This is the observed series
forecast	forecast values based on the model
multiplier	Half-width of confidence interval in standard normal units. Defaults to 1.96.
CI.percent	Width of confidence band. Defaults to the standard normal, two-sided value associated with the multiplier (95 percent for the default multiplier=1.96).

series	Name of time series will be used to construct the main title for the plot.
ylim, xlab, ylab, main	standard trellis parameters
...	additional arguments to xyplot.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

seqplot

---

strip.background0 *Turn off the coloring in the trellis strip labels. Color 0 is the background color.*

Description

Turn off the coloring in the trellis strip labels. Color 0 is the background color.

Usage

```
strip.background0()
```

Author(s)

Richard M. Heiberger <rmh@temple.edu>

---

strip.xysplom *strip function that is able to place the correlation or regression coefficient into the strip label.*

Description

strip function that is able to place the correlation and/or regression coefficient into the strip label.

Usage

```
strip.xysplom(which.given, which.panel, var.name, factor.levels,
 shingle.intervals, par.strip.text = trellis.par.get("add.text"),
 strip.names = c(TRUE, TRUE), style = 1, ...)
```

Arguments

`which.given, which.panel, var.name, factor.levels, shingle.intervals`
: arguments to `strip.default`.

`par.strip.text, strip.names, style, ...`
: more arguments to `strip.default`.

Details

The function looks for the specific factor names `c("corr","beta","corr.beta")`. If it finds them, it goes up the calling sequence to locate the data for the panel. Then it calculates the correlation and/of regression coefficient and inserts the calculated value(s) as the value for the strip label.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`xysplom`

---

sufficient *Calculates the mean, standard deviation, and number of observations in each group of a data.frame that has one continuous variable and two factors.*

Description

Calculates the mean, standard deviation, and number of observations in each group of a data.frame that has one continuous variable and two factors.

Usage

```
sufficient(x,
           yname = dimnames(x)[[2]][[1]],
           factor.names.keep = dimnames(x)[[2]][-c(1, 2)])
```

Arguments

`x`
: `data.frame` containing a continuous variable and two factors.

`yname`
: Character name of response variable.

`factor.names.keep`
: Character vecotr containing the names of two factors in the `x` data.frame.

Value

Data.frame containing five columns and as many rows as are implied by the crossing of the two factors. Each row contains the mean in a column with the name `yname` and its factor values in columns named with the name in `factor.names.keep`. The standard deviation of the observations in the group are in the column `"sd"` and the number of observations in the group is in the column `"nobs"`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`intxplot`

`summary.arma.loop` *summary and print and subscript methods for tsdiagplot and related objects.*

Description

summary and print and subscript methods for tsdiagplot and related objects.

Usage

```
## S3 method for class 'arma.loop'
summary(object, ...)
## S3 method for class 'arma.loop.list'
summary(object, ...)
## S3 method for class 'arma.loop'
print(x, ...)
## S3 method for class 'arma.loop.list'
print(x, ...)
## S3 method for class 'tsacfplots'
print(x, ...)
## S3 method for class 'arma.loop'
x[..., drop = TRUE]
## S3 method for class 'diag.arma.loop'
x[..., drop = TRUE]
```

Arguments

`x, object`	object to be summarized or printed or subscripted.
`...`	additional arguments
`drop`	See `Extract`.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

arma.loop, tsacfplots, tsdiagplot

`t.trellis` *Interchange the x- and y-axes for an S-Plus trellis object. Interchange the conditioning variables for an R trellis object.*

Description

In S-Plus, change the `"trellis"` object to effectively, and after-the-fact convert formulas from `(y ~ x | g)` to `(x ~ y | g)`. This is needed in S-Plus because most S-Plus trellis functions do not permit factors on the right-hand side of the formula.

In R lattice functions, factors are permitted on the right-hand side of the formula. Therefore, to change the x and y axes within each panel, change the formula from `(y ~ x | g)` to `(x ~ y | g)`. The HH `t.trellis` function in R calls `lattice::t.trellis` to interchange the conditioning variables for an R trellis object. See `update.trellis`.

Usage

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

Arguments

`x`	any `"trellis"` object.

Details

In S-Plus, a warning is generated if the panel component of the `"trellis"` object is a function. No warning is generated when the panel component of the `"trellis"` object is a character string naming the function. We interchange the `x` and `y` values of the `c("x", "y", "xlab", "ylab", "xlim", "ylim", "adj.xlim", "adj.ylim")` components of the `"trellis"` object, and are aware of special features of the panel functions listed in the S-Plus version of `t.trellis`. The transpose will usually work correctly for other panel functions.

Value

In S-Plus, a `"trellis"` object with all x and y components interchanged. In R, the argument `"trellis"` object is sent to `lattice::t.trellis` to interchange the conditioning variables.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

update.trellis

Examples

```
tmp <- data.frame(y=rnorm(30), x=factor(rep(1:3,10)))
if.R(r=
     bwplot(y ~ x, data=tmp)
     ,s=
     t(bwplot(x ~ y, data=tmp))
     )
```

---

tsacfplots *Coordinated time series and ACF and PCF plots.*

Description

Coordinated time series and ACF and PCF plots.

Usage

```
tsacfplots(x,
           ylab=deparse(substitute(x)),
           x.name=ylab[[1]],
           main=paste("Series:", x.name),
           lag.at=NULL,
           lag.max=NULL,
           lag.units=NULL,
           lag.0=TRUE,
           ...)

acf.pacf.plot(x,
      ylab=NULL,
      series=deparse(substitute(x)),
      main=paste("ACF and PACF:", series),
      lag.max,
      lag.units=frequency(x),
      lag.at=pretty(apacf$lag),
      lag.labels=lag.at*lag.units,
              lag.0=TRUE,
      ...)
```

Arguments

x	time series
ylab, main	standard trellis arguments.
x.name, series	Character string, name for the time series.
lag.at	Location of ticks for the acf and pacf plots.
lag.labels	Labels for ticks for the acf and pacf plots.
lag.max	Maximum lag used in the acf and pacf plots.
lag.units	Units for time series, defaults to frequency(x)
lag.0	Logical. If TRUE, then plot the correlation (identically 1) at lag=0. If FALSE, do not plot the correlation at lag=0.
...	Additional arguments to seqplot for tsacfplots. Additional arguments to strip.default for acf.pacf.plot.

Details

The acf and pacf plots are scaled identically.

Value

"tsacfplots" object containing two "trellis" objects.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

seqplot

Examples

```
tsacfplots(co2)
acf.pacf.plot(co2)
```

---

tsdiagplot *Times series diagnostic plots for a structured set of ARIMA models.*

Description

Times series diagnostic plots for a structured set of ARIMA models.

Usage

```
tsdiagplot(x,
           p.max=2, q.max=p.max,
           model=c(p.max, 0, q.max), ## S-Plus
           order=c(p.max, 0, q.max), ## R
           lag.max=36, gof.lag=lag.max,
           armas=if.R(
             s=arma.loop(x, model=model,
               series=deparse(substitute(x)), ...),
             r=arma.loop(x, order=order,
               series=deparse(substitute(x)), ...)),
           diags=diag.arma.loop(armas, x,
                                lag.max=lag.max,
                                gof.lag=gof.lag),
           ts.diag=rearrange.diag.arma.loop(diags),
           lag.units=ts.diag$tspar["frequency"],
           lag.lim=range(pretty(ts.diag$acf$lag))*lag.units,
           lag.x.at=pretty(ts.diag$acf$lag)*lag.units,
           lag.x.labels={tmp <- lag.x.at
                         tmp[as.integer(tmp)!=tmp] <- ""
                         tmp},
           lag.0=TRUE,
           main, lwd=0,
           ...)

acfplot(rdal, type="acf",
        main=paste("ACF of std.resid:", rdal$series,
                   "   model:",         rdal$model),
        lag.units=rdal$tspar["frequency"],
        lag.lim=range(pretty(rdal[[type]]$lag)*lag.units),
        lag.x.at=pretty(rdal[[type]]$lag)*lag.units,
        lag.x.labels={tmp <- lag.x.at
                      tmp[as.integer(tmp)!=tmp] <- ""
                      tmp},
        lag.0=TRUE,
        xlim=xlim.function(lag.lim/lag.units),
        ...)

aicsigplot(z, z.name=deparse(substitute(z)), series.name="ts",
           model=NULL,
           xlab="", ylab=z.name,
           main=paste(z.name,  series.name, model),
           layout=c(1,2), between=list(x=1,y=1), ...)

residplot(rdal,
          main=paste("std.resid:", rdal$series,
                     "   model:",  rdal$model),
          ...)
```

```
gofplot(rdal,
        main=paste("P-value for gof:", rdal$series,
                   "   model:",         rdal$model),
       lag.units=rdal$tspar["frequency"],
        lag.lim=range(pretty(rdal$gof$lag)*lag.units),
        lag.x.at=pretty(rdal$gof$lag)*lag.units,
        lag.x.labels={tmp <- lag.x.at
                      tmp[as.integer(tmp)!=tmp] <- ""
                      tmp},
        xlim=xlim.function(lag.lim/lag.units),
        pch=16, ...)
```

Arguments

x	Time series vector.
p.max, q.max	Maximum number of AR and MA arguments to use in the series of ARIMA models.
model	A valid S-Plus model for arima.mle.
order	A valid R order for arima. The additional argument seasonal may also be used.
lag.max	Maximum lag for the acf and pacf plots.
gof.lag	Maximum lag for the gof plots.
armas	An arma.loop object.
diags	An diag.arma.loop object.
ts.diag, rdal	A list constructed as a rearranged diag.arma.loop object.
lag.units	Units for time series, defaults to frequency(x)
lag.lim	scaling for xlim in acf and pacf plots.
lag.x.at, lag.x.labels	Location of ticks and labels for the acf and pacf plots.
lag.0	Logical. If TRUE, then plot the correlation (identically 1) at lag=0. If FALSE, do not plot the correlation at lag=0.
type	"acf" or "pacf"
z	A matrix constructed as the aic or sigma2 component of the sumamry of a arma.loop object.
z.name	"aic" or "sigma2"
series.name	Character string describing the time series.
xlab, ylab, layout, between, pch, xlim, main, lwd	Standard trellis arguments.
...	Additional arguments. tsdiagplot sends them to arima or arima.mle. acfplot, aicsigplot residplot, and gofplot send them to xyplot.

Value

`tsdiagplot` returns a `"tsdiagplot"` object which is a list of `"trellis"` objects. It is printed with its own print method.

The other functions return `"trellis"` objects.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

References

"Displays for Direct Comparison of ARIMA Models" The American Statistician, May 2002, Vol. 56, No. 2, pp. 131-138. Richard M. Heiberger, Temple University, and Paulo Teles, Faculdade de Economia do Porto.

Richard M. Heiberger and Burt Holland (2004), Statistical Analysis and Data Display, Springer, ISBN 0-387-40270-5

See Also

`tsacfplots`, `arma.loop`

Examples

```
data(tser.mystery.X)
X <- tser.mystery.X

X.dataplot <- tsacfplots(X, lwd=1, pch.seq=16, cex=.7)
X.dataplot

X.loop <- if.R(
               s=
               arma.loop(X, model=list(order=c(2,0,2)))
               ,r=
               arma.loop(X, order=c(2,0,2))
               )
X.dal <- diag.arma.loop(X.loop, x=X)
X.diag <- rearrange.diag.arma.loop(X.dal)
X.diagplot <- tsdiagplot(armas=X.loop, ts.diag=X.diag, lwd=1)
X.diagplot

X.loop
X.loop[["1","1"]]
```

`vif` *Calculate the Variance Inflation Factor*

Description

The VIF for predictor i is $1/(1 - R_i^2)$, where R_i^2 is the R^2 from a regression of predictor i against the remaining predictors.

Usage

```
vif(xx, ...)

## Default S3 method:
vif(xx, y.name, na.action = na.exclude, ...) ## xx is a data.frame

## S3 method for class 'formula'
vif(xx, data, na.action = na.exclude, ...)   ## xx is a formula

## S3 method for class 'lm'
vif(xx, na.action = na.exclude, ...)  ## xx is a "lm" object computed with x=TRUE
```

Arguments

`xx`	`data.frame`, or `formula`, or `lm` object computed with `x=TRUE`.
`na.action`	See `na.action`.
`...`	additional arguments.
`y.name`	Name of Y-variable to be excluded from the computations.
`data`	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.

Details

A simple diagnostic of collinearity is the *variance inflation factor, VIF* one for each regression coefficient (other than the intercept). Since the condition of collinearity involves the predictors but not the response, this measure is a function of the X's but not of Y. The VIF for predictor i is $1/(1-R_i^2)$, where R_i^2 is the R^2 from a regression of predictor i against the remaining predictors. If R_i^2 is close to 1, this means that predictor i is well explained by a linear function of the remaining predictors, and, therefore, the presence of predictor i in the model is redundant. Values of VIF exceeding 5 are considered evidence of collinearity: The information carried by a predictor having such a VIF is contained in a subset of the remaining predictors. If, however, all of a model's regression coefficients differ significantly from 0 (p-value $< .05$), a somewhat larger VIF may be tolerable.

Value

Vector of VIF values, one for each X-variable.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

lm.

Examples

```
data(usair)
if.R(s={usair <- usair}, r={})

usair$lnSO2 <- log(usair$SO2)
usair$lnmfg <- log(usair$mfgfirms)
usair$lnpopn <- log(usair$popn)

usair.lm <- lm(lnSO2 ~ temp + lnmfg + wind + precip, data=usair, x=TRUE)

vif(usair.lm)  ## the lm object must be computed with x=TRUE

vif(lnSO2 ~ temp + lnmfg + wind + precip, data=usair)

vif(usair)

vif(usair, y.name="lnSO2")
```

X.residuals *Residuals from the regression of each column of a data.frame against all the other columns.*

Description

Calculate the residuals from the regression of each column of a data.frame against all the other columns.

Usage

```
X.residuals(x, ...)

## Default S3 method:
X.residuals(x, y.name, na.action = na.exclude, ...) ## x is a data.frame
```

```
## S3 method for class 'formula'
X.residuals(x, data, na.action = na.exclude, ...)   ## x is a formula

## S3 method for class 'lm'
X.residuals(x, na.action = na.exclude, ...)  ## x is a "lm" object computed with x=TRUE
```

Arguments

x	data.frame, or formula, or lm object computed with x=TRUE.
na.action	See na.action.
...	additional arguments.
y.name	Name of Y-variable to be excluded from the computations.
data	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.

Value

Data.frame of residuals, one column from each regression.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard~M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

lm, vif, case.lm.

Examples

```
data(usair)
usair$lnSO2 <- log(usair$SO2)
usair$lnmfg <- log(usair$mfgfirms)
usair$lnpopn <- log(usair$popn)

usair.lm <- lm(lnSO2 ~ temp + lnmfg + wind + precip, data=usair, x=TRUE)

X.residuals(usair.lm)  ## the lm object must be computed with x=TRUE

X.residuals(lnSO2 ~ temp + lnmfg + wind + precip, data=usair)

X.residuals(usair)

X.residuals(usair, y.name="lnSO2")
```

xysplom *scatterplot matrix with potentially different sets of variables on the rows and columns.*

Description

scatterplot matrix with potentially different sets of variables on the rows and columns. The slope or regression coefficient for simple least squares regression can be displayed in the strip label for each panel.

Usage

```
xysplom(x, ...)

## S3 method for class 'formula'
xysplom(x, data = sys.parent(), na.action = na.pass, ...)

## Default S3 method:
xysplom(x, y=x, group, relation="free",
        x.relation=relation, y.relation=relation,
        xlim.in=NULL, ylim.in=NULL,
        corr=FALSE, beta=FALSE, abline=corr||beta, digits=3,
        x.between=NULL, y.between=NULL,
        between.in=list(x=x.between, y=y.between),
        scales.in=list(
          x=list(relation=x.relation, alternating=FALSE),
          y=list(relation=y.relation, alternating=FALSE)),
        strip.in=strip.xysplom,
        pch=16, cex=.75,
        panel.input=panel.xysplom, ...,
        cartesian=TRUE,
        plot=TRUE)
```

Arguments

x	In the "formula" method, a formula. In the "default" method, a data.frame. Any variables that are used in a formula with + should be numeric. Factors are not rejected, but their levels will be combined strangely.
...	other arguments to xyplot. z
data	data.frame
na.action	See na.action. Defaults to na.pass because xyplot does sensible things with missing data.
y	In the "default" method, a data.frame with the same number of rows as the data.frame in x.

group — In the "default" method, a data.frame with the same number of rows as the data.frame in x.

relation, x.relation, y.relation,scales.in — Alternate ways to get to the scales(relation=) arguments to xyplot.

xlim.in, ylim.in — Alternate ways to get to the scales(limits=) arguments to xyplot.

corr, beta — Display the correlation and/or the regression coefficient for lm(y ~ x) for each panel in an additional strip label.

abline — logical. If TRUE, draw the least squares regression line within each panel. By default the abline is FALSE unless at least one of corr or beta is TRUE.

digits — number of significant digits for the correlation coefficient.

x.between, y.between, between.in — Alternate ways to get to the between= argument to xyplot.

strip.in — strip function that knows how to handle the corr and beta displays.

pch, cex — arguments to xyplot

panel.input — panel function used by xyplot within each panel. When abline==FALSE, the default panel function calls panel.xyplot. When abline==TRUE, the default panel function calls panel.xyplot and panel.abline(lm(y~x, na.action=na.exclude)).
Note that we use na.action=na.exclude inside lm.

cartesian — When cartesian==TRUE, the cartesian product of the left-hand side number of variables and the right-hand side number of variables defines the number of panels in the display. When cartesian==FALSE, each variable in the left-hand side is paired with the variable in the corresponding position in the right-hand side and only those pairs are plotted. Both sides must have the same number of variables.

plot — Defaults to TRUE. See details.

Details

The argument plot=TRUE is the normal setting and then the function returns a "trellis" object. When the argument plot=FALSE, the function returns the argument list that would otherwise be sent to xyplot. This list is interesting when the function xysplom was designed because the function works by restructuring the input data and running xyplot on the restructured data.

Value

When plot=TRUE (the normal setting), the "trellis" object containing the graph.
When plot=FALSE, the restructured data that must be sent to the xyplot function.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2004b). *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer. ISBN 0-387-40270-5.

See Also

xyplot in R.

Examples

```
## xysplom syntax options

tmp <- data.frame(y=rnorm(12), x=rnorm(12), z=rnorm(12), w=rnorm(12),
                  g=factor(rep(1:2,c(6,6))))
tmp2 <- tmp[,1:4]

xysplom(y + w ~ x , data=tmp, corr=TRUE, beta=TRUE, cartesian=FALSE, layout=c(1,2))

xysplom(y + x ~ z | g, data=tmp, layout=c(2,2))
xysplom(y + x ~ z | g, data=tmp, cartesian=FALSE)

xysplom(w + y ~ x + z, data=tmp)
xysplom(w + y ~ x + z | g, data=tmp, layout=c(2,4))
xysplom(w + y ~ x + z | g, data=tmp, cartesian=FALSE)

## Not run:
## xyplot in R has many similar capabilities with xysplom
if.R(r=
       xyplot(w + z ~ x + y, data=tmp, outer=TRUE)
     ,s=
       {}
    )

## End(Not run)
```

Index