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


See Also

ancova
ancova

Compute and plot oneway analysis of covariance

Description

Usage

ancova(formula, data.in = sys.parent(), ..., 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, ...)

## S3 method for class 'ancova':
coefficients(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` in R and both `xyplot` and `trellis.args` in S-Plus.
- **object**: An "aov"
- **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


See Also

`ancova-class` `aov` `xyplot`

Examples

```r
hotdog <- read.table(hh("datasets/hotdog.dat"), header=TRUE)

## y ~ x
ancova(Sodium ~ Calories, data=hotdog, groups=Type)

## y ~ a
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
apple <- read.table(hh("datasets/apple.dat"), header=TRUE)
apple$treat <- factor(apple$treat)
contrasts(apple$treat) <- contr.treatment(6)
apple$block <- factor(apple$block)
ancova(yield ~ treat + pre, data=apple, blocks=block)
```

4
**anova.mean**  
*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**

`anova.mean(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` in S-Plus and `anova.lm` in R if the original data, rather than the summary data, had been available.

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**See Also**

`anova.lm` in S-Plus and `anova.lm` in R. `plot.mmc.multicomp`

**Examples**

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

pulmonary <- read.table(hh("datasets/pulmonary.dat"), header=TRUE, row.names=NULL)
names(pulmonary)[3] <- "FVC"
names(pulmonary)[1] <- "smoker"
pulmonary$smoker <- factor(pulmonary$smoker, levels=pulmonary$smoker)
row.names(pulmonary) <- pulmonary$smoker
pulmonary
```
anova.mean(pulmonary$smoker, 
pulmonary$n, 
pulmonary$FVC, 
pulmonary$s, 
ylabel="pulmonary")

aov.sufficient  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 vcov.sufficient.

Usage
aov.sufficient(formula, data = NULL, 
projections = FALSE, qr = TRUE, contrasts = NULL, 
weights = data$n, sd = data$s, 
...)
vcov.sufficient(object, ...)

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

Value
For aov.sufficient, an object of class c("aov", "lm"). For vcov.sufficient, 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 aov.sufficient in glht requires the vcov. and df arguments.
Author(s)
Richard M. Heiberger <rmh@temple.edu>

See Also
MMC and aov in R, aov in S-Plus.

Examples
## This example is from Hsu and Peruggia
## This is the R version
## See ?aov.sufficient for S-Plus

if.R(s={},
  r=
  
pulmonary <- read.table(hh("datasets/pulmonary.dat"), header=TRUE,
                        row.names=NULL)
  names(pulmonary)[3] <- "FVC"
  names(pulmonary)[1] <- "smoker"
  pulmonary$smoker <- factor(pulmonary$smoker, levels=pulmonary$smoker)
  row.names(pulmonary) <- pulmonary$smoker
  pulmonary.aov <- aov.sufficient(FVC ~ smoker,
                             data=pulmonary)
  summary(pulmonary.aov)
  pulmonary.mca <- glht(pulmonary.aov,
                       linfct=mcp(smoker="Tukey"),
                       df=pulmonary.aov$df.residual,
                       vcov.=vcov.sufficient)
  plot(pulmonary.mca)

  pulm.lmat <- cbind("npnl-mh"=c( 1, 1, 1, 1,-2,-2), ## not.much vs lots
                      "n-nl"  =c( 3,-1,-1,-1, 0, 0), ## none vs light
                      "p-nl"  =c( 0, 2,-1,-1, 0, 0), ## {} arbitrary 2 df
                      "n-1"   =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)
  if.R(r=pulm.lmat <- rbind(Int=0, pulm.lmat[-1,]),
        s={})
  pulm.lmat

  pulmonary.mmc <- glht.mmc(pulmonary.aov,
                            linfct=mcp(smoker="Tukey"),
                            df=pulmonary.aov$df.residual,
                            vcov.=vcov.sufficient,
                            lmat=pulm.lmat,
                            calpha=attr(confint(pulmonary.mca)$confint,"calpha"))
  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.
plot(confint(as.glht(pulmonary.mmc$mca)),
     xlim=par()$usr[1:2], xaxs="i",
     main="", xlab=""
)

## orthogonal contrasts
plot(pulmonary.mmc, print.lmat=TRUE, col.lmat.signif='blue', col.iso='gray'
)

## pairwise and orthogonal contrasts on the same plot
plot(pulmonary.mmc, print.mca=TRUE, print.lmat=TRUE,
     col.mca.signif='red', col.lmat.signif='blue', col.iso='gray',
     lty.lmat.not.signif=2)
}

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, ...)

## 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

print.glht.mmc.multicomp(x, ...) ## R. yes, spell it out.

as.multicomp(x, ...)

## S3 method for class 'glht':
as.multicomp(x,      ## glht object
focus,      ## currently required
ylabel=as.character(terms(x$model)[[2]]),
means=model.tables(x$model, type="means",
as.glht(x, ...)

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

Arguments


... 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

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 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. glht.mmc.glht sets this argument to TRUE for the none component.

level Confidence level. Defaults to 0.95.

calpha User-specified critical point. See confint.glht.hh and confint.glht.

method See type in confint.glht.
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.

`print.glht.mmc.multicomp(x, ...)` uses `print.glht` on each component of a `mmc.multicomp` object and therefore prints only the estimates of the comparisons.

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.

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 getting the plot differ.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

`mmc, glht` in R, `multicomp` in S-Plus.

```
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 assistance from Deepayan Sarkar.

See Also
interaction2wt

bwplot.formula
Replacement for bwplot.formula and compute.packet in R to allow them to work with the new "positioned" class.

Description
Replacement for bwplot.formula and compute.packet in R trellis to allow them to work with the new "positioned" class.

Usage
bwplot.formula(x, data = NULL, allow.multiple = is.null(groups) || outer, outer = FALSE, auto.key = FALSE, aspect = "fill", panel = "panel.bwplot", prepanel = NULL, scales = list(), strip = TRUE, groups = NULL, xlab, xlim, ylab, ylim, box.ratio = 1, horizontal = NULL, drop.unused.levels = lattice.getOption("drop.unused.levels"), ..., default.scales = if (horizontal)
  list(y = list(tck = 0, alternating = FALSE, rot = 0))
else
  list(x = list(tck = 0, alternating = FALSE)),
subscripts = !is.null(groups),
subset = TRUE)

compute.packet.new(cond, levels, cond.levels)

Arguments
x, data, allow.multiple, outer, auto.key, aspect, panel, prepanel, scales, strip, groups, xlab,
See bwplot.formula.
cond, levels standard values used by lattice:::compute.packet.
cond.levels
Details

`bwplot` makes the assumption that the numeric equivalent of a factor `f` are always integers from the sequence `seq(1:length(levels(f)))`. When the "positioned" class is introduced, this is no longer the right assumption. The numeric values need to be chosen from the "position" attribute.

Value

See `bwplot.formula`.

Author(s)

original by Deepayan Sarkar (Deepayan.Sarkar@R-project.org). Modifications by Richard M. Heiberger <rmh@temple.edu>

See Also

`bwplot.formula`.

---

### ci.plot

#### 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

```r
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=range(newfit$pi.fit),  
pch=16,  
main.cex=1,  
main=list(paste(100*conf.level,  
"% confidence and prediction intervals for ",  
substitute(lm.object), sep=""), cex=main.cex), ...  
)```

12
Arguments

\texttt{lm.object} \hspace{1cm} \text{Linear model for one y and one x variable.}
\texttt{xlim} \hspace{1cm} \text{xlim for plot. Default is based on data from which \texttt{lm.object} was constructed.}
\texttt{newdata} \hspace{1cm} \text{\texttt{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 \texttt{data.frame} containing a vector spanning the range of observed data. User-specified values are appended to the default vector.}
\texttt{conf.level} \hspace{1cm} \text{Confidence level for intervals, defaults to .95}
\texttt{data} \hspace{1cm} \text{\texttt{data} extracted from the \texttt{lm.object}}
\texttt{newfit} \hspace{1cm} \text{Constructed \texttt{data.frame} containing the predictions, confidence interval, and prediction interval for the \texttt{newdata}.}
\texttt{ylim} \hspace{1cm} \text{ylim for plot. Default is based on the constructed prediction interval.}
\texttt{pch} \hspace{1cm} \text{Plotting character for observed points.}
\texttt{main.cex} \hspace{1cm} \text{Font size for main title.}
\texttt{main} \hspace{1cm} \text{Main title for plot}
\texttt{...} \hspace{1cm} \text{Additional arguments to be passed to panel function.}

Value

"\texttt{trellis}" object containing the plot.

Note

The \texttt{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 \texttt{newfit} within the function depends on the system.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

\texttt{lm, predict.lm}

Examples

\begin{verbatim}
  tmp <- data.frame(x=rnorm(20), y=rnorm(20))
  tmp.lm <- lm(y ~ x, data=tmp)
  ci.plot(tmp.lm)
\end{verbatim}
**col.hh**  
*Initializing Trellis Displays*

**Description**

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

**Usage**

```r
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**

```r
## function name or result of function call
if.R(r={
  trellis.device(theme="col.hh")
  trellis.device(theme=col.hh())
}

## result of function call
trellis.par.set(theme=col.hh())
}
```

---

**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**

```r
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

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

```r
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

dev.copy2eps in R, export.graph in S.

Examples

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

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

```r
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

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


See Also

ancova, ancova, ancova, ancova, ancova, ancova, ancova, ancova, ancova, ancova

Examples

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

| hh | Resolve filenames relative to the HH directory. |

### Description

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

### Usage

`hh(file)`

### Arguments

- `file` Character string giving the pathname of a file in the HH online files relative to the `options()$HH.ROOT.DIR` directory. `file` is the name exactly as specified in Heiberger and Holland (2004). The `hh` function resolves those names to the location where the files are stored on your computer.

### Details

The datasets and code listings in Heiberger and Holland (2004) 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 library from an R or S-Plus package, then option `options()$HH.ROOT.DIR` is set automatically.

If you installed the HH library by unzipping the downloaded online files (not a package), then you must manually define the `HH.ROOT.DIR` option in your working directory (for example, "c:/HOME/yourname/.Data" in S-Plus, or the `.GlobalEnv` in R). You may do this in your `.First` function in either system. Or you may enter at the command prompt a statement of the form `options(HH.ROOT.DIR="c:/HOME/hh")` in Windows or `options(HH.ROOT.DIR="/usr/users/hh")` in Unix. See Appendix B of Heiberger and Holland (2004) for further details. The `options` statement may need to be modified to match the location of the `hh` directory on your machine. If you use more than one computer, you may need a different value for the `HH.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` function itself is unchanged.

### Value

Fully expanded, absolute pathname for the input filename.
Example

```r
hotdog <- read.table(hh("datasets/hotdog.dat"), header=TRUE)
```

**hov**

*Homogeneity of Variance*

Description

One way 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

```r
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


See Also

`aov`, `plot.hov`

Examples

```r
turkey <- read.table(hh("datasets/turkey.dat"), header=FALSE)
names(turkey) <- c("diet","wt.gain")
turkey$diet <- factor(turkey$diet,
    labels=c("control","A1","A2","B1","B2"))

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

```r
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

```r
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

```r
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`.

22
Value

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

positioned.

Examples

```r
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

<table>
<thead>
<tr>
<th>Plot all main effects and twoway interactions in a multifactor design</th>
</tr>
</thead>
</table>

#### 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

```r
interaction2wt(x, ...)
```

- **S3 method for class 'formula':**
  ```r
  interaction2wt(x, data = sys.parent(), responselab, ...)
  ```

- **Default S3 method:**
  ```r
  interaction2wt(x, response.var, responselab = deparse(substitute(y)), 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 = list(cex=.7, responselab=responselab),
scales.additional,
main.in =
  paste(responselab,
    " : main effects and 2-way interactions",
    sep = ""),
xlab = list(labels = ""),
ylab = list(labels = ""),
simple=FALSE,
box.ratio=if (simple) .32 else 1,
label.as.interaction.formula=TRUE,
...
main.cex)

Arguments

Arguments when x is a formula.

x
  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.

... 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
  bwplot in R or bwplot in S-Plus.

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 \texttt{between} argument.

\texttt{between}  
trellis/lattice \texttt{between} argument. If used, \texttt{between} has precedence over both the \texttt{x.between} and \texttt{y.between} arguments.

cex  
S-Plus: changes the size of the median dot in the boxplots. R: doesn’t do anything.

\texttt{panel.input}  
panel function. Default is \texttt{panel.interaction2wt}.

\texttt{label.as.interaction.formula}  
logical. If \texttt{TRUE}, each panel has a single strip label of the form \( y \sim a \mid b \). If \texttt{FALSE}, each panel has a pair of strip labels, one for the \texttt{trace} factor and one for the \texttt{x} factor.

\texttt{strip.input}  
strip function. Default depends on the value of \texttt{label.as.interaction.formula}.

\texttt{par.strip.text.input}  
\texttt{par.strip.text} argument.

\texttt{scales.additional}  
additional arguments to \texttt{scales} argument.

\texttt{main.in}  
Text of main title.

\texttt{xlab}  
No effect.

\texttt{ylab}  
No effect.

\texttt{main.cex}  
cex for main title.

\texttt{rot}  
Rotation of x tick labels and y tick labels. Only 0 and 90 will look good.

Value

"\texttt{trellis}" object containing the plot.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

\texttt{panel.interaction2wt}

Examples

\begin{verbatim}
vulcan <- read.table(hh("datasets/vulcan.dat"), header=TRUE) vulcan$filler <- factor(vulcan$filler) position(vulcan$raw) <- (1:4)+.5 position(vulcan$pretreat) <- 2:4 interaction2wt(wear ~ filler + pretreat + raw, data=vulcan) interaction2wt(wear ~ filler + raw, data=vulcan,)
\end{verbatim}

25
interaction2wt(wear ~ filler + raw, data=vulcan,
    simple=TRUE, simple.scale=c(fill=.15, raw=.2))

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)
    )
    print(
        bwplot(len ~ supp | unpositioned(dose), data=ToothGrowth, layout=c(3,1))
    )
    print(
        bwplot(len ~ dose | supp, data=ToothGrowth, layout=c(2,1),
               box.ratio=.5,
               panel=panel.bwplot.intermediate.hh,
               scales=list(x=list(at=position(ToothGrowth$dose))))
    )
    print(
        bwplot(len ~ dose | supp, data=ToothGrowth, layout=c(2,1),
               box.ratio=.5,
               scales=list(x=list(at=position(ToothGrowth$dose))))
    )
}, 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",
    condition.length=1,
    condition.length=1,
    condition.length=1,
    condition.length=1,
    condition.length=1,
    condition.length=1,
    condition.length=1,
    condition.length=1,
    condition.length=1,
    condition.length=1,
    condition.length=1,
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(groups)),
offset=
  as.numeric(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

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

rhiz.clover <- read.table(hh("datasets/rhiz3-clover.dat"), header=TRUE)
rhiz.clover$comb <- factor(rhiz.clover$comb,  
    labels=c("clover","clover+alfalfa"))
position(rhiz.clover$comb) <- c(2,5)
rhiz.clover$strain <- factor(rhiz.clover$strain,  
    labels=c('3DOk1','3DOk5','3DOk4','3DOk7','3DOk13','k.comp'))
```
rhiz.clover$Npg <- rhiz.clover$nitro / rhiz.clover$weight

## 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

```r
result <- data.frame(-1/x, -1/sqrt(x), log(x), sqrt(x), x, x^2)
```

```r
names(result) <- c(-1, -.5, 0, .5, 1, 2)
```

#### Usage

```r
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,
strip.function = ladder.f,
strip.number = 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
...
)
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.

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 in R or strip.default in S-Plus.

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.
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


See Also

panel.cartesian

Examples

```r
## some country names have embedded blanks
tv <- if.R(r=
  read.fwf(hh("datasets/tv.dat"),
    widths=c(22,6,7,7,4,2),
    strip.white=TRUE,
    na.strings="*",
    row.names=1)
  ,s=
  read.table(hh("datasets/tv.dat"),
    sep=c(1,23,29,36,43,47),
    na.strings="*"
  )
)
names(tv) <- c("life.exp","ppl.per.tv","ppl.per.phys",
  "fem.life.exp","male.life.exp")
```
## Default: single strip label per panel
ladder(life.exp ~ ppl.per.phys, data=tv,
    main="Ladder of Powers for Life Expectancy and People per Physician",
    dsx="ppp", dsy="le")

## double strip label
if.R(r=
ladder(life.exp ~ ppl.per.phys, data=tv,
    main="Ladder of Powers for Life Expectancy and People per Physician",
    strip.number=2,
    dsx="ppp", dsy="le")
    ,'s=
ladder(life.exp ~ ppl.per.phys, data=tv,
    main="Ladder of Powers for Life Expectancy and People per Physician",
    strip.number=2,
    dsx="ppp", dsy="le",
    axis3.line=1.2)
)

## turn off strip labels
if.R(r=
ladder(life.exp ~ ppl.per.phys, data=tv,
    strip.number=0,
    main="Ladder of Powers for Life Expectancy and People per Physician")
    ,'s=
ladder(life.exp ~ ppl.per.phys, data=tv,
    strip.number=0,
    main="Ladder of Powers for Life Expectancy and People per Physician",
    axis3.line=0)
    )

---

**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').
Description

Case statistics for regression analysis. `lm.case` 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

```
# Example usage
fit <- lm(y ~ x, data=mtcars)
lm.case(fit)
plot.case(fit)
```

Arguments

- **fit**: "lm" object computed with `x=TRUE`
- **lms**: `summary.lm(fit)`
lm.influence(fit)

x

In plot.case, the matrix output from lm.case 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

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 lm.case and plot.case are based on: Section 4.3.3 "Influence of Individual Observations in Chambers and Hastie", Statistical Models in S.
**Value**

`lm.case` 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 residuals), `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**


**See Also**

`lm.influence` in R, `lm.influence` in S-Plus.

**Examples**

```r
kidney <- read.table(hh("datasets/kidney.dat"), header=TRUE)

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

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

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

36
Description

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

Usage

```r
calinfct(model, focus,
    mmm.data=model$model,
    formula.in=terms(model),
    linfct.Means=
        multcomp::meanslinfct(model, focus, mmm.data, formula.in),
    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.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.
isowna, 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 glht.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


## Examples

```r
## This example is from Hsu and Peruggia
## This is the S-Plus version
## See ?aov.sufficient for R

if.R(r={},

```}

```r
pulmonary <- read.table(hh("datasets/pulmonary.dat"), header=TRUE, row.names=NULL)
names(pulmonary)[3] <- "FVC"
names(pulmonary)[1] <- "smoker"
pulmonary$smoker <- factor(pulmonary$smoker, levels=pulmonary$smoker)
row.names(pulmonary) <- pulmonary$smoker
pulmonary.aov <- aov.sufficient(FVC ~ smoker, data=pulmonary)
summary(pulmonary.aov)
```

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

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

```r
pulm.lmat <- cbind("npnl-mh"=c( 1, 1, 1, 1,-2,-2), ## not.much vs lots
"n-npl" =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
```

```r
## mmc.multicomp object
pulmonary.mmc <-
multicomp.mmc.mean(pulmonary$smoker, pulmonary$n, pulmonary$FVC, pulmonary$s, 40
```

See Also

- mmc

Examples

```r
```
 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

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

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

## S3 method for class 'lm':
glht.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)
},
ylabel=as.character(terms(model)[[2]]),
lmat=t(linfct),
lmat.rows=-1,
lmat.scale.abs2=TRUE,
estimate.sign=1,
order.contrasts=TRUE,
level=.95,
calpha=NULL,
alternative = c("two.sided", "less", "greater"),
...
)

multicomp.mmc(..., comparisons="mca", ## S-Plus
    lmat, lmat.rows=-1,
lmat.scale.abs2=TRUE,
    ry, 
    plot=TRUE,
    crit.point, 
    iso.name=TRUE,
estimate.sign=1,
x.offset=0, 
    order.contrasts=TRUE, 
    main, 
    main2)

"[.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 mcp in R.
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 \texttt{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 (\texttt{mca}, \texttt{none}, \texttt{lmat}) components by height on the MMC plot. This will place the contrasts in the multicomparisons plots in the same order as in the MMC plot.

alternative

Direction of alternative hypothesis. See \texttt{confint} in R. S-Plus \texttt{multicomp} uses the argument \texttt{bounds} for this concept.

level

Confidence level. Defaults to 0.95.

crit.point, calpha

critical value for the tests. The value from the specified \texttt{multicomp} method is used for the user-specified contrasts when \texttt{lmat} is specified. This argument is called \texttt{crit.point} with \texttt{multicomp} in S-Plus and \texttt{calpha} when used with \texttt{glht} and \texttt{confint} in R.

plot

logical, display the plot if \texttt{TRUE}.

ry, iso.name, x.offset, main, main2

arguments to \texttt{plot.mmc.multicomp}.

x, drop

See "[".

Details

By default, if \texttt{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 \texttt{lmat} is specified.

We get the right contrasts automatically if the \texttt{aov} is one-way. If we specify an \texttt{lmat} for one-way it must have a leading row of 0.

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

\texttt{glht} in R works from either an "\texttt{glht}" object or an "\texttt{aov}" object. \texttt{multicomp.mmc} in S-Plus works from an "\texttt{aov}" object.

Value

An "\texttt{mmc.multicomp}" object contains either the first two or all three of the "\texttt{multicomp}" components

\texttt{mca}

\texttt{none}

\texttt{lmat}

\texttt{mca} Object containing the pairwise comparisons.

\texttt{none} Object comparing each mean to 0.
lmomat  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 glht.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.

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


See Also

as.multicomp, plot.mmc.multicomp

Examples

## Use glh.t.mmc with R.
## Use multicomp.mmc with S-Plus.

## data and ANOVA

catalystm <- read.table(hh("datasets/catalystm.dat"), header=FALSE,
col.names=c("catalyst","concent"))
catalystm$catalyst <- factor(catalystm$catalyst, labels=c("A","B","C","D"))

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 ~ concentrat, data=catalystm,
scales=list(cex=1.5),
xlab=list("concentration", cex=1.5),
ylab=list("catalyst", cex=1.5))

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

## pairwise comparisons

catalystm.mmc <-
if.R(r=glht.mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey")),
s=multicomp.mmc(catalystm1.aov, plot=FALSE))
catalystm.mmc
plot(catalystm.mmc)
plot(catalystm.mmc$none)
plot(catalystm.mmc$lmat)

## user-specified contrasts

catalystm.lmat <- cbind("AB-D" =c(0, 1, 1, 0,-2),
"A-B" =c(0, 1,-1, 0, 0),
"ABD-C"=c(0, 1, 1,-3, 1))
if.R(r=catalystm.lmat <- catalystm.lmat[-2,],
s={})
dimnames(catalystm.lmat)[[1]] <- dimnames(catalystm.mmc$lmat)[[1]]
zapsmall(catalystm.lmat)
if.R(s=dimnames(catalystm.mca$lmat)[[1]],
r=dimnames(catalystm.mca$linfct)[[2]])
catalystm.mmc <-
if.R(r=glht.mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey"),
  lmat=catalystm.lmat)
,s=multicomp.mmc(catalystm1.aov, lmat=catalystm.lmat,
  plot=FALSE))

plot(catalystm.mmc)
plot(catalystm.mmc)
plot(catalystm.mmc$none)
plot(catalystm.mmc$lmat)
```

## Dunnett's test

```r
weightloss <- read.table(hh("datasets/weightloss.dat"), header=TRUE)
weightloss <- data.frame(loss=unlist(weightloss),
```

45
group=rep(names(weightloss), rep(10,5))

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=
  glht.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 <- read.table(hh("datasets/display.dat"), header=TRUE)
display$panel <- factor(display$panel)  ## display$panel <- positioned(display$panel, value=(1:3)+.5)
display$emergenc <- factor(display$emergenc)
```r
displayf.aov <- aov(time ~ emergenc * panel, data=display)
anova(displayf.aov)

## multiple comparisons
tmp <- if.R(  
r=glht(displayf.aov, linfct=mcp(panel="Tukey")),
  s=multicomp(displayf.aov, "panel", plot=FALSE)
  zapsmall(    
    if.R(r=t(tmp$linfct),
      s=tmp$lmat)
  )
)

## MMC plot
displayf.mmc <-
if.R(r=glht.mmc(displayf.aov, linfct=mcp(panel="Tukey"), focus="panel", lmat.rows=5:6),
  s=multicomp.mmc(displayf.aov, "panel", lmat.rows=6:8, plot=FALSE))
plot(displayf.mmc)

## orthogonal contrasts
zapsmall(mca.lmat <- displayf.mmc$mca$lmat)
panel.lmat <- cbind("3-12"=mca.lmat[,1] + mca.lmat[,2],
                     "1-2"=mca.lmat[,3])
displayf.mmc <-
if.R(r=glht.mmc(displayf.aov, linfct=mcp(panel="Tukey"), focus="panel",
                lmat.rows=5:6, lmat=panel.lmat),
  s=multicomp.mmc(displayf.aov, "panel", lmat.rows=6:8,
                lmat=panel.lmat, plot=FALSE))
plot(displayf.mmc)

## 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.

maiz <- read.table(hh("datasets/maiz.dat"), header=TRUE)
maiz$hibrido <- factor(maiz$hibrido,
  levels=c("P3747","P3732","Mol17","A632","LH74"))
maiz$nitrogeno <- factor(maiz$nitrogeno)
position(maiz$nitrogeno) <- c(1, 2.5, 4, 5.5) ## forces class="ordered"
interaction2wt(yield ~ hibrido+nitrogeno+bloque, data= maiz)
interaction2wt(yield ~ hibrido+nitrogeno, data=maiz)
maiz.aov <- aov(yield ~ nitrogeno*hibrido + Error(bloque/nitrogeno), data=maiz)
summary(maiz.aov)
summary(maiz.aov,

47
```
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 glht.mmc requires treatment contrasts
  contrasts(maiz$nitrogeno) <- "contr.treatment"
  sapply(maiz[-1], contrasts)
},
  s={})

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

if.R(s={
  maiz2.mca <- multicomp(maiz2.aov, focus="hibrido")
  ## plot(maiz2.mca)
  dimnames(maiz2.mca$lmat)[[1]]
  maiz2.mmc <- multicomp.mmc(maiz2.aov, focus="hibrido",
    lmat.rows=16:20, plot=FALSE)
  old.mar <- par(mar=c(15,4,4,7)+.1)
  plot(maiz2.mmc)
  par(mar=c(2,4,28,7)+.1, new=TRUE)
  old.cex <- par(cex=.8)
  plot(maiz2.mmc$mca, col.signif=8, lty.signif=1, xlabel.print=FALSE,
    xaxs="d", plt=par()$plt+c(0,0,-.25,.05), xrange.include=c(-30,40))
  par(old.cex)
  par(old.mar)
},r={
  maiz2.mca <- glht(maiz2.aov, linfct=mcp(hibrido="Tukey"))
  dimnames(maiz2.mca$linfct)[[2]]
  maiz2.mmc <- glht.mmc(maiz2.aov, linfct=mcp(hibrido="Tukey"), lmat.rows=9:12)
  old oma <- par(oma=c(12,3,0,4))
  plot(maiz2.mmc)
  par(oma=c(0,3,22,4), new=TRUE)
  plot(maiz2.mmc$mca,
    xlim=par()$usr[1:2], xaxs="i",
    main="", xlab="", cex.axis=.7)
  par(old.oma)
})

---

**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


See Also

MMC, as.glht in R, multicomp.reverse

Examples

```r
## continue with the example in glht.mmc in R, or multicomp.mmc in S-Plus

catalystm <- read.table(hh("datasets/catalystm.dat"), header=FALSE,
                        col.names=c("catalyst","concent"))
catalystm$catalyst <- factor(catalystm$catalyst, labels=c("A","B","C","D"))
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)

if.R(r=
  glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey"))
  print(confint(catalystm.mca))

  glht.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=

  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

**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**


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.

Usage

```r
norm.setup(xlim.in=c(-2.5,2.5),
           ylim.in = c(0, 0.4)/se,
           mean=0,
           main.in=main.calc,
           se=sd/sqrt(n), sd=1, n=1,
           df.t=NULL,
           ...
)

norm.curve(mean=0, se=sd/sqrt(n),
           critical.values=mean + se*c(-1, 1)*z.975,
           z=do.call("seq",
                       as.list(c((par()$usr[1:2]-mean)/se, length=109))),
           shade, col=par("col"),
           axis.name=ifelse(is.null(df.t) || df.t == Inf, "z", "t"),
           second.axis.label.line=3,
           sd=1, n=1,
           df.t=NULL,
           ...
)

norm.observed(xbar, t.xbar, col="blue")
```
Arguments

xlim.in, ylim.in

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, 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
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.in
Main title. Default value is: if (is.null(df.t)) ## normal ifelse(!is.null(se) && is.null(sd) && is.null(n)), paste("normal density: se = ", round(se,3)), "Standard Normal Density N(0,1)" else { ## t distribution if (length(df.t) != 1) stop("df.t must have length 1") ifelse(!is.null(se) && is.null(sd) && is.null(n)), paste("t density: se = ", round(se,3), ", df = ", df.t, sep=""), paste("t density, df =", df.t)) }

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 and the area of the shaded region.

axis.name
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 distribution with df.t degrees of freedom, centered on the alternate hypothesis value of the mean.

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
xbar-value of the observed data.

t.xbar
t-value of the observed data under the null hypothesis.

...
Other arguments which are ignored.

Author(s)

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

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.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*(-1.96, 1.96))

norm.setup(xlim=c(-3, 6))
norm.curve(crit=1.645, mean=1.645+1.281552, col='green', shade="left", axis.name="z1")
norm.curve(crit=1.645, col='red')

norm.setup(xlim=c(-6, 12), se=2)
norm.curve(crit=2*1.645, se=2, mean=2*(1.645+1.281552), col='green', shade="left", axis.name="z1")
norm.curve(crit=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(crit=crit.val, se=se.val, df.t=df.val, mean=mu.alt,
   col="green", shade="left", axis.name="t1")
norm.curve(crit=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(crit=crit.val, se=se.val, mean=mu.alt,
   col="green", shade="left", axis.name="z1")
norm.curve(crit=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(crit=15.5, se=se.val, mean=16.3,
   col="gray", shade="right")
norm.curve(crit=15.5, se.val, df.t=6, mean=14.7,
  col='green', shade="left", axis.name="t1", second.axis.label.line=4)
norm.curve(crit=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(crit=15.5, se=se.val, mean=15.5,
  col='gray', shade="right")
norm.curve(crit=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(crit=15.5, se=se.val, mean=15.5,
  col='gray', shade="none")

par(old.par)

---

**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)`

**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.

**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**

- `objects` in R, `objects` in S-Plus.
Examples

```
objip("qq")
objip("qq")
objip("qq$")
```

---

**odds.ratio**

*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**

```
odds.ratio(x, alpha = 0.05)
```

```
plot.odds.ratio(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. **odds.ratio** requires a single number in the range [0,1]. **plot.odds.ratio** 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.
- **legend.x**: x position of left-hand side of legend.
- **oma**: outer margin `par()$oma`, needed to make room for legend.

**Value**

- **plot.odds.ratio** draws a plot and invisibly returns the same list as **odds.ratio** for the first value of **alpha**. **odds.ratio** 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 $\psi_{ihat}$ calculated as $p(ci.ln.psihat)$
prob1 $\text{seq}(0,1,.05)$, set of $p_1$ values for plotting.
odds1 $p_1/(1-p_1)$
odds2 odds for the second row needed to retain $\psi_{ihat}$ with the specified $odds1$, calculated as $odds1*psi_{ihat}$.
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

Examples
glasses <- read.table(hh("datasets/glasses.dat"), header=FALSE)
glasses <- matrix(unlist(glasses),
nrow=2,
dimnames=list(c("glasses","no.glasses"),
c("delinq","non.del")))

## draw the iso-odds ratio plot with 50% CI and 95% CI,
## invisibly return the 95% CI
plot.odds.ratio(glasses)

orthog.complete Construct an orthogonal matrix which is an arbitrary completion of the column space of the input set of columns.

Description

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 \texttt{orthog.complete}, an \(n\)-row matrix of one or more columns. For \texttt{orthog.construct}, a set of contrasts for a factor.

y  
matrix of coefficients specifying the linear combinations estimated. This will usually be the \texttt{lm} from an S-Plus "\texttt{multicomp}" object or the \texttt{linfct} component from an R "\texttt{glht}" object.

\texttt{normalize}, \texttt{abs2.rows}, \texttt{x.rows}
The default normalizes the sum of squares of the rows in \texttt{abs2.rows} or \texttt{x.rows} to 1. The optional value \texttt{normalize="abs2"} scales the rows in \texttt{abs2.rows} or \texttt{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.

\texttt{Int}
logical. Default \texttt{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 \(x\).

\texttt{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 \texttt{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 input 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 (\texttt{drop.Int=FALSE}), the constant column can be returned.

By default the columns are scaled to have sum of squares equal 1. If \texttt{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 \texttt{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


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={})

60
Panel functions for bwplot.

Description

Panel function for bwplot that give the user control over the placement of the boxes.

Usage

panel.bwplot.intermediate.hh(x, y, horizontal = TRUE, 
    transpose=!horizontal, 
    pch, col, 
    at=if (horizontal) levels(as.factor(y)) else levels(as.factor(x)), ## S-Plus only 
    ...) 

Arguments

x, y, pch, col, horizontal

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

S-Plus only. bwplot in S-Plus gets x and y as numeric. level and position information is not sent through, therefore we need to specify it here when non-default positions are used.

...

Extra arguments, if any, for ’panel.bwplot’.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

panel.xyplot, bwplot, interaction2wt
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 in S-Plus or panel.bwplot in R.

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

bwplot in S-Plus or bwplot in R.

Examples

62
**Description**

trellis panel function, with labeled rows and columns and without strip labels. Designed for use with the ladder of powers plot.

**Usage**

```r
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 its 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


See Also

ladder, xysplom

Examples

rent <- read.table(hh("datasets/rent.dat"), ## Weisberg's file alr162
col.names=c("rnt.alf","rnt.till",
"cow.dens","prop.past","lime"))
rent$lime <- factor(rent$lime, labels=c("no.lime","lime"))
rent$alf.till <- rent$rnt.alf / rent$rnt.till

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,
panel.ci.plot  

**Default Panel Function for ci.plot**

**Description**

This is the default panel function for ci.plot.

**Usage**

```r
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>
panel.interaction2wt  Plot all main effects and two-way interactions in a multi-factor 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, 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:

x levels of x-factor
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

See Also

ci.plot, xyplot, lm
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 `TRUE`, the standard errors are calculated from the sufficient statistics for each group as the groups standard deviation divided by the square root of the group observation count. If a numeric vector, it is evaluated in the environment of the sufficient statistics.

Passed to `panel.bwplot.intermediate.hh`

extra arguments, primarily color, to be passed to `panel.bwplot.intermediate.hh`

arguments to be passed through to the `key` for the trace-factor in each row of the display.

"position" attribute of factor.

logical. `TRUE` if simple effects are to be displayed.

named list of offset and scale for the response and trace factors. See `interaction.positioned` for their use.

data frame containing factors from the input data frame

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.

see documentation for `strip.default`

Force `strip.names=TRUE`

force `style=1`

Richard M. Heiberger <rmh@temple.edu>


`interaction2wt`, `panel.bwplot.intermediate.hh`
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

```r
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

```r
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=

68
splot( ~ logley, 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

```r
if.R(r=
  partial.corr(longley[,1:3], longley[,4:6])
,s=
  partial.corr(longley.x[,1:3], longley.x[,4:6])
)
```

---

**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 `plot.hov` function graphs the components of the Brown and Forsyth test statistic.

Usage

```r
plot.hov(x, data = sys.parent(), method = "bf", ## x is a formula
  transpose.in = TRUE, ...)
```

## users will normally use the formula above and will not call the
## method directly.

```r
plot.hov.bf(x, group, ## x is the response variable
  y.name = deparse(substitute(x)),
  group.name = deparse(substitute(group)),
  transpose.in = TRUE, ...)
```

## users will normally use the formula above and will not call the
## panel function directly.

```r
panel.hov(..., transpose = TRUE)
```
Arguments

- `x` Formula appropriate for oneway anova in `plot.hov`. Response variable in `plot.hov.bf`.
- `data` data.frame
- `method` Character string defining method. At this time the only recognized method is "bf" for the Brown-Forsyth method.
- `transpose.in, 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


See Also

- `aov`, `hov`

Examples

```r
turkey <- read.table(hh("datasets/turkey.dat"), header=FALSE)
names(turkey) <- c("diet","wt.gain")
turkey$diet <- factor(turkey$diet, 
  labels=c("control","A1","A2","B1","B2"))

hov(wt.gain ~ diet, data=turkey)
plot.hov(wt.gain ~ diet, data=turkey)
```
plot.mmc.multicomp  

MMC (Mean–mean Multiple Comparisons) plot.

Description

MMC (Mean–mean Multiple Comparisons) plot.

Usage

plot.mmc.multicomp(x,  
xlab="contrast value",  
ylab=none$ylab,  
focus=none$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

main, main2

main.method.phrase, main2.method.phrase

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
col.mca.not.signif, lty.mca.not.signif, lwd.mca.not.signif
col.lmat.signif, lty.lmat.signif, lwd.lmat.signif
col.lmat.not.signif, lty.lmat.not.signif, lwd.lmat.not.signif
col.iso, lty.iso, lwd.iso
col.contr0, lty.contr0, lwd.contr0
decdigits.ybar number of decimal digits in the left-axis labels.

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.

Author(s)

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


See Also

mmc

Examples

```r
## continue with the example in mmc
catalystm <- read.table(hh("datasets/catalystm.dat"), header=FALSE, 
  col.names=c("catalyst","concent"))
if.R(r=
  oldcon <- options(contrasts = c("contr.treatment", "contr.treatment"))
  ,s={})
catalystm$catalyst <- factor(catalystm$catalyst, labels=c("A","B","C","D"))
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
if.R(r=
  options(oldcon)
  ,s={})
catalystm.mca <-
if.R(r=glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey")), 
  s=multicomp(catalystm1.aov, plot=F))
catalystm.lmat <- cbind("AB-D" =c(0, 1, 1, 0,-2),
  "A-B" =c(0, 1,-1, 0, 0),
  "ABD-C"=c(0, 1, 1,-3, 1))
if.R(r=catalystm.lmat <- catalystm.lmat[-2,],
  s={})
dimnames(catalystm.lmat)[[1]] <-
if.R(s=dimnames(catalystm.mca$lmat)[[1]], 
  r=dimnames(catalystm.mca$linfct)[[2]])
catalystm.mmc <-
if.R(r=glht.mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey"), 
  lmat=catalystm.lmat) 
  ,s=multicomp.mmc(catalystm1.aov, lmat=catalystm.lmat, 
  plot=FALSE)
)
old.mar <- if.R(s=par(mar=c(5,12,4,6)+.1), 
  r=par(mar=c(5,6,4,4)+.1))
```

74
## pairwise contrasts, default settings
plot(catalystm.mmc, print.lmat=FALSE)

## centering, scaling, emphasize significant contrasts
plot(catalystm.mmc, x.offset=1.6, ry.mmc=c(50,58), print.lmat=FALSE, 
     col.mca.signif='red')

## user-specified contrasts
plot(catalystm.mmc, x.offset=1.6, ry.mmc=c(50,58), 
     col.lmat.signif=blue)

## reduce intensity of isomeans grid, number isomeans grid lines
plot(catalystm.mmc, x.offset=1.6, ry.mmc=c(50,58), 
     lty.iso=2, col.iso='darkgray', iso.name=FALSE, 
     col.lmat.signif='blue')

## both pairwise contrasts and user-specified contrasts
plot(catalystm.mmc, x.offset=1.6, ry.mmc=c(50,58), lty.iso=2, 
     col.iso='darkgray', print.mca=TRUE, 
     col.lmat.signif='blue', col.mca.signif='red')

par(old.mar)

---

**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` and `plot.multicomp.hh` coerce their argument to an "glht" object and plots that with the appropriate `plot` method. In S-Plus, `plot.multicomp.hh` augments the standard `plot.multicomp` to give additional user arguments to control the appearance of the plot.

**Usage**

```r
## S3 method for class 'multicomp':
plot(x, ...) ## R only

## S3 method for class 'multicomp.hh':
plot(x, ylabel = x$ylabel, href = 0, uniform = T, 
     plt.in = c(0.2, 0.9, 0.1, 0.9), 
     x.label.adj=1, 
     xrange.include=href, 
     xlim, 
     comparisons.per.page=21,
```
Arguments

x A "multicomp" object.
ylabel Y label on graph.
... other arguments to plot.multicomp.
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.

col.signif, lty.signif, lwd.signif Color, line type, and line width for significant comparisons. S-Plus only.
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.

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 an "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


See Also

mmc in both languages, glht in R, multicomp in S-Plus.

Examples

```r
## data and ANOVA

catalystm <- read.table(hh("datasets/catalystm.dat"), header=FALSE, col.names=c("catalyst","concent"))
catalystm$catalyst <- factor(catalystm$catalyst, labels=c("A","B","C","D"))
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
catalystm.mca <- if.R(r=glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey")), s=multicomp(catalystm1.aov, plot=F))
if.R(s=plot(catalystm.mca), r=plot(confint(catalystm.mca)))
```

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

is.numeric.positioned(x, ...)
as.numeric.positioned(x, ...)
as.double.positioned(x, ...)
x[...], drop=FALSE
"[.positioned"(x, ..., drop=FALSE)
as.positioned(x)
is.positioned(x)
positioned(x, ..., value)
print.positioned(x, ...)
unique.positioned(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 "[" in R or "]" in S-Plus.
incomparables See unique in R or unique in S-Plus.

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, or an ordered factor whose
levels cannot be coerced to numeric, then the integers 1:length(levels(x)) are returned.
If x is an an ordered factor whose levels can be coerced to numeric, then the numeric values
corresponding to the levels are returned. If x is numeric, then x itself is returned.

as.numeric.positioned(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 in R, factor in S-Plus.

Examples

```r
## 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.numeric(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.numeric(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.numeric(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.numeric(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.numeric(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.numeric(tmp)

## boxplots coded by week
tmp <- data.frame(Y=rnorm(40, rep(c(20,25,15,22), 10)),
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


See Also

position.
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`

---

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

regr.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 \texttt{lm(y ~ x)}. Any model object with one \texttt{x} variable, such as the quadratic \texttt{lm(y ~ x + I(x^2))} can be used.
coef.model   Defaults to the coefficients of the \texttt{model} argument. Other intercept and slope coefficients for a straight line (for example, \texttt{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, \texttt{alt.function=function(x) {3 + 2*x + 3*x^2}}. All coefficients must be specified.
main, xlab, ylab arguments to \texttt{plot}.
jitter.x   logical. If \texttt{TRUE}, the \texttt{x} is jittered before plotting. Jittering is often helpful when there are multiple \texttt{y}-values at the same level of \texttt{x}.
resid.plot   If \texttt{FALSE}, then do not plot the residuals. If "\texttt{square}" , then call \texttt{resid.squares} to plot the squared residuals. If \texttt{TRUE} (or anything else), then call \texttt{resid.squares} to plot straight lines for the residuals.
points.yhat   logical. If \texttt{TRUE}, the predicted values are plotted.
... other arguments.
length.x.set   number of points used to plot the predicted values.
x.name   If the \texttt{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 \texttt{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


See Also

\texttt{resid.squares}

Examples

```r
hardness <- read.table(hh("datasets/hardness.dat"), header=TRUE)

## 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))
```

84
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.

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


See Also

regr1.plot

Examples

hardness <- read.table(hh("datasets/hardness.dat"), header=TRUE)
hardness.lin.lm <- lm(hardness ~ density, data=hardness)

plot(hardness ~ density, data=hardness, xlab=c(22,73), ylab=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 trellis.args in S-Plus and in xyplot in R.
...

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.
Examples

```r
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)
```

---

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/or 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

\texttt{sufficient(x,}
\quad \texttt{yname = dimnames(x)[[2]][[1]],}
\quad \texttt{factor.names.keep = dimnames(x)[[2]][-c(1, 2)]})

Arguments

\begin{itemize}
\item \texttt{x} \quad \texttt{data.frame} containing a continuous variable and two factors.
\item \texttt{yname} \quad \texttt{Character name of response variable.}
\item \texttt{factor.names.keep} \quad \texttt{Character vector containing the names of two factors in the x data.frame.}
\end{itemize}

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 \texttt{yname} and its factor values in columns named with the name in \texttt{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 \texttt{<rmh@temple.edu>}

See Also

\texttt{intxplot}

\begin{itemize}
\item \texttt{t.trellis} \quad \textit{Interchange the x- and y-axes for an S-Plus trellis object. Identity function for an R lattice object.}
\end{itemize}

Description

In S-Plus, change the "trellis" object to effectively, and after-the-fact convert formulas from \texttt{(y ~ x | g)} to \texttt{(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 we make \texttt{t.trellis} an identity operator in R.

Usage

\texttt{t.trellis(x)}

Arguments

\begin{itemize}
\item \texttt{x} \quad \texttt{any "trellis" object.}
\end{itemize}
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 returned unchanged.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

In R: *xyplot*. In S-Plus: *xyplot*.

Examples

```r
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))
)
```

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(x, ...)

## Default S3 method:
vif(x, y.name, na.action = na.exclude, ...) ## x is a data.frame

## S3 method for class 'formula':
vif(x, data, na.action = na.exclude, ...) ## x is a formula

## S3 method for class 'lm':
vif(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.

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

See Also

`lm` in R or `lm` in S-Plus.

Examples

```r
usair <- read.table(hh("datasets/usair.dat"),
  col.names=c("SO2","temp","mfgfirms","popn",
    "wind","precip","raindays"))
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

```r
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`.

92
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**


**See Also**

`lm` in R or `lm` in S-Plus, vif, lm.case.

**Examples**

```r
usair <- read.table(hh("datasets/usair.dat"),
  col.names=c("SO2","temp","mfgfirms","popn",
           "wind","precip","raindays"))
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
...
other arguments to xyplot.
data  data.frame
na.action  See na.action in R, na.exclude in S-Plus. 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


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 ~ x , data=tmp)

xysplom(y ~ x + w , data=tmp)
xysplom(y + w ~ x , data=tmp)

xysplom(y + w ~ x , data=tmp, corr=TRUE)
xysplom(y + w ~ x , data=tmp, beta=TRUE)
xysplom(y + w ~ x , data=tmp, abline=TRUE)
xysplom(y + w ~ x , data=tmp, corr=TRUE, abline=FALSE)

xysplom(y ~ x | g, data=tmp)

xysplom(y ~ x | g, data=tmp, layout=c(2,1))

xysplom(y + x ~ z | g, data=tmp)
xysplom(y + x ~ z | g, data=tmp, layout=c(2,2))
xysplom(y ~ x + z | g, data=tmp)
xysplom(y ~ x + z | g, data=tmp, layout=c(2,2))

xysplom(w + y ~ x + z, data=tmp)
xysplom(w + y ~ x + z | g, data=tmp)
xysplom(w + y ~ x + z | g, data=tmp, layout=c(2,4))

xysplom(w + y ~ x + z, data=tmp, cartesian=FALSE)
xysplom(w + y ~ x + z, data=tmp, cartesian=FALSE)
xysplom(w + y ~ x + z | g, data=tmp, cartesian=FALSE)

xysplom(w + y ~ x + z, data=tmp, scales=list(relation="same"))
xysplom(w + y ~ x + z, data=tmp, x.relation="same")

xysplom(~ y + x + z , data=tmp)
xysplom(~ y + x + z | g, data=tmp)
xysplom(~ y + x + z , data=tmp, corr=TRUE)
xysplom(~ y + x + z | g, data=tmp, corr=TRUE)
xysplom(~ y + x + z | g, data=tmp, corr=TRUE, digits=2)
xysplom(~ y + x + z | g, data=tmp, corr=TRUE, layout=c(3,6))

## Not run:
## These three examples run from R and the command line in S-Plus.
## They don't run from inside the Splus CMD check.

xysplom(~ tmp)
xysplom(~ tmp | tmp$g)
xysplom(tmp$y ~ tmp2 | tmp$g)

## End(Not run)
xysplom(g ~ x, data=tmp)
xysplom(x ~ g, data=tmp)

## Subscripting requires the x=, y= notation.
## Subscripting doesn't work with the y ~ x notation.
## Not run:
  xysplom(~ tmp[, c("x","y")])  ## doesn't work
  xysplom(tmp2[, c("w","z")], tmp[, c("x","y")])  ## doesn't work
## End(Not run)

## use this instead
  xysplom(x = tmp[, c("x","y")])
  xysplom(y = tmp2[, c("w","z")], x = tmp[, c("x","y")])

## or, even better, use the y ~ x notation
  xysplom(~ x + y, data=tmp)
  xysplom(w + z ~ x + y, data=cbind(tmp, tmp2))

## xyplot in R has many similar capabilities with xysplom
if.R(r=
  xyplot(w + z ~ x + y, data=tmp, outer=TRUE)
  ,s=
    {}
  )

97