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.
anova.ancova(object, ...)

predict.ancova(object, ...)

print.ancova(x, ...)

model.frame.ancova(formula, ...)

summary.ancova(object, ...)

plot.ancova(x, ...)

coef.ancova(object, ...)

coefficients.ancova(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
aov xyplot

Examples

```r
hotdog <- read.table(hh("datasets/hotdog.dat"), header=TRUE)
## y ~ x  ## constant line across all groups
ancova(Sodium ~ Calories, data=hotdog, groups=Type)

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

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

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

## 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$block)
ancova(yield ~ treat + pre, data=apple, blocks=apple$block)
```

**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="group")
pulmonary

anova.mean(row.names(pulmonary),
           pulmonary$n,
           pulmonary$ybar,
           pulmonary$s,
           ylabel="pulmonary")
```
Support functions in R for MMC (mean–mean multiple comparisons) plots.

Description

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

Usage

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

Arguments

- `x` "hmtest" object for `as.multicomp`. An "mmc.multicomp" object for `print.mmc.multicomp`. A "multicomp" object for `as.hmtest` and `plot.multicomp`.
- `...` other arguments.
- `group` not currently used
- `focus` response variable name on the graph.
- `ylabel` response variable name on the graph.
- `lmcall` not currently used
- `means` means of the response variable on the focus factor.
- `lmat`, `lmat.rows` `simint.mmc` object for the same model formula.
- `eps` absolute error tolerance as double. See `simint`.

5
Value

as.multicomp is a generic function to change its argument to a "multicomp" object.
as.multicomp.hmtest changes an "hmtest" object to a "multicomp" object.
print.mmc.multicomp prints a "mmc.multicomp" object.

Note

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

ci.plot

Plot confidence and prediction intervals for simple linear regression

description

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

Usage

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

# S3 method for class lm :
ci.plot(lm.object,

xlim=range(data[,as.character(formula.lm[[3]]))],
newdata=data.frame(seq(xlim[1], xlim[2], length=51)),
conf.level=.95,
data=model.frame(lm.object),

6
Arguments

**lm.object**  
Linear model for one y and one x variable.

**xlim**  
**xlim** for plot. Default is based on data from which **lm.object** was constructed.

**newdata**  
**data.frame** containing data for which predictions are wanted. Defaults to vector spanning range of observed data.

**conf.level**  
Confidence level for intervals, defaults to .95

**data**  
**data** extracted from the **lm.object**

**newfit**  
Constructed **data.frame** containing the predictions, confidence interval, and prediction interval for the **newdata**.

**ylim**  
**ylim** for plot. Default is based on the constructed prediction interval.

**pch**  
Plotting character for observed points.

**main.cex**  
Font size for main title.

**main**  
Main title for plot

...  
Additional arguments to be passed to panel function.

Value

"trellis" object containing the plot.

Note

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

**lm**, **predict.lm**
Examples

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

---

**col.hh**  
*Initializing Trellis Displays*

### Description

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

#### Usage

```
col.hh()
```

#### Value

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

#### Author(s)

Richard M. Heiberger \(<rmh@temple.edu>\)

#### See Also

`trellis.device`, `trellis.par.get`

#### Examples

```r
## function name or result of function call
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
do.formula.trellis.xysplom(formula, data, na.action = na.pass)

Arguments

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

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

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

Author(s)
Richard M. Heiberger <rmh@temple.edu>

See Also
formula, na.action

Examples

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

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

---

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

**Description**

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

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

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

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

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

Arguments

at, label, main, gp, draw, vp
See link[grid]{grid.xaxis}.

labels
label values if you don’t want the defaults

Value

See link[grid]{grid.xaxis}.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

link[grid]{grid.xaxis}

HH-package

Support software for Statistical Analysis and Data Display by
Richard M. Heiberger and Burt Holland

Description

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

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

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

Author(s)

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

References


See Also

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

Examples

## interaction2wt()
## multicomp.mmc() ## S-Plus
## simint.mmc() ## R
## ancova()
## xysplom()
## plot.case()
## bwplot() ## with position
## tsacfplots() ## at this writing, only S-Plus
## tsdiagplot() ## 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 \texttt{options(HH.ROOT.DIR)} directory. \texttt{file} is the name exactly as specified in Heiberger and Holland (2004). The \texttt{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 \texttt{hh} directory, which must be stored as \texttt{options(HH.ROOT.DIR)}.

If you installed the HH library from an R or S-Plus package, then option \texttt{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 \texttt{HH.ROOT.DIR} option in your working directory (for example, "/c:/HOME/yourname/.Data" in S-Plus, or the \texttt{.GlobalEnv} in R). You may do this in your \texttt{.First} function in either system. Or you may enter at the command prompt a statement of the form \texttt{options(HH.ROOT.DIR="c:/HOME/hh")} in Windows or \texttt{options(HH.ROOT.DIR="/usr/users/hh")} in Unix. See Appendix B of Heiberger and Holland (2004) for further details. The \texttt{options} statement may need to be modified to match the location of the \texttt{hh} directory on your machine. If you use more than one computer, you may need a different value for the \texttt{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 \texttt{hh} function itself is unchanged.

Value

Fully expanded, absolute pathname for the input filename.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


Examples

\begin{verbatim}
hotdog <- read.table(hh("datasets/hotdog.dat"), header=TRUE)
\end{verbatim}
Homogeneity of Variance

Description

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

Usage

```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

\texttt{aov, plot.hov}

Examples

\begin{verbatim}
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)
\end{verbatim}

\begin{tabular}{ll}
\textbf{if.R} & \textit{Conditional Execution for R or S-Plus} \\
\end{tabular}

Description

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

Usage

\texttt{if.R(r, s)}

Arguments

\texttt{r} \hspace{1cm} Any R expression, including a group of expressions nested in braces. Assignments made in this expression are available to the enclosing function.

\texttt{s} \hspace{1cm} 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 \texttt{lattice} in R (based on \texttt{grid} graphics) are very different from panel functions for \texttt{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>
interaction2wt  
Plot all main effects and two-way interactions in a multi-factor design

Description

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

Usage

interaction2wt(x, ...)

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

## Default S3 method:
interaction2wt(x,
response.var,
resposnelab = deparse(substitute(y)),
relation = list(x = "free", y = "same"),
x.relation = relation$x,
y.relation = relation$y,
digits = 3,
x.between = c(1, rep(0, ncol(x) - 1)),
y.between = NULL,
cex = 0.75,
panel.input = panel.interaction2wt,
strip.input = strip.interaction2wt,
par.strip.text.input = list(cex = 0.7),
scales.additional,
main.in =
  paste(resposnelab,
        ": main effects and 2-way interactions",
        sep = ""),
xlab.in = TRUE,
xlab = list(labels = ""),
ylab = list(labels = ""),
...
...,
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 \( y \sim g_1 + g_2 + \ldots \).
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.

response.lab
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.

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 relation argument.

y.between
y value of relation argument.

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

panel.input
panel function. Default is panel.interaction2wt.

strip.input
strip function. Default is strip.interaction2wt.

par.strip.text.input
par.strip.text argument.

scales.additional
additional arguments to scales argument.

main.in
Text of main title.

xlab.in
S-Plus: FALSE suppresses x labels in trellis. R: no effect.

xlab
No effect.

ylab
No effect.

main.cex
cex for main title.

Value

"trellis" object containing the plot.
Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

panel.interaction2wt

Examples

anova(aov(len ~ supp*dose, data=ToothGrowth))
interaction2wt(len ~ supp + dose, data=ToothGrowth)
bwplot(len ~ supp | dose, data=ToothGrowth, layout=c(3,1))

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)
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,
ladder.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.xplot. Any replacement panel function must have those argu-
ment 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.

<table>
<thead>
<tr>
<th></th>
<th>notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x^p - 1)/p</td>
<td></td>
</tr>
<tr>
<td>(1/x - 1)/(-1)</td>
<td>-1.0</td>
</tr>
<tr>
<td>(1/sqrt(x)-1)/(-.5)</td>
<td>-0.5</td>
</tr>
<tr>
<td>log(x)</td>
<td>0.0</td>
</tr>
<tr>
<td>((sqrt(x)-1)/.5)</td>
<td>0.5</td>
</tr>
<tr>
<td>x-1</td>
<td>1.0</td>
</tr>
<tr>
<td>(x^2 - 1)/2</td>
<td>2.0</td>
</tr>
</tbody>
</table>

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,
```

21
lm.case  case statistics for regression analysis

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

lm.case(fit, lms = summary.lm(fit), lmi = lm.influence(fit))

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

22
panel.case(x, y, subscripts, rownames, group.names,
  nn, pp, ss, cex.threshold,
  panel.number, ## R only. S-Plus ignores this argument
  par.settings, ## R only. S-Plus ignores this argument
  obs.large, obs.large.env,
  ...)  

Arguments

fit "lm" object
lms summary.lm(fit)
lmi 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 between trellis/lattice argument.
oma In S-Plus, change par()$oma to make room for the threshold values. A warning is printed when par()$oma is changed as the delayed printing of trellis objects implies we can’t return it to the original value automatically. In R, this argument is ignored. Instead, we use the par.settings argument to xyplot inside plot.case. The par.settings becomes one component of the "trellis" object that is the value of plot.case and is therefore automatically applied every time the object is printed.
cex.threshold Multiplier for cex for the threshold values.
main.in main title for xyplot. The default main title displays the linear model formula from fit.
sigma.in standard error for the fit.
p.in The number of degrees of freedom associated with the fitted model.
obs.large Object name where the names of all observations for which the standard thresholds are exceeded will be stored. The default name is .lm.case.large.
ob.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.
panel.number R only, which panel are we in? Ignored in S-Plus.
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

kidney2.lm <- lm(clearance ~ concen + age + weight + concen*age, data=kidney)
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

---

**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**

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

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 "hmtest" 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.hmtest` in R, `simint.reverse` in R, `multicomp.reverse` in S-Plus

Examples

```r
## continue with the example in simint.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=
   simint(concent ~ catalyst, data=catalystm, type="Tukey")
catalystm.mca <-
   simint.mmc(concent ~ catalyst, data=catalystm, type="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.

as.hmtest(catalystm.mmc$mca)

## now simplify the contrast names by removing the string "catalyst"

tmp <- multicomp.label.change(catalystm.mmc$mca, "catalyst", "")
as.hmtest(tmp)

## for consistency with the S-Plus example, we change all factor level
## "A" to "control".

as.hmtest(multicomp.label.change(tmp, "A", "control"))

}, s=

## continue with the example in simint.mmc in R, or mmc in S-Plus

```r

# catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
catalystm.mca <- multicomp(catalystm1.aov, method="Tukey")
catalystm.mmc <- multicomp.mmc(catalystm1.aov, method="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.
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".
multicomp.label.change(catalystm.mmc$mca, "A", "control")
```

### norm.curve

**plot a normal curve with both x and z axes.**

**Description**

Plot a normal curve with both x (with `mean` and `se` as specified) and z (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 axis. The existence of any of the arguments `se`, `sd`, `n` forces dual x and z 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.

**Usage**

```r

norm.setup(xlim.in=c(-2.5,2.5),
           ylim.in = c(0, 0.4)/se,
           mean=0,
```

27
main.in=ifelse(
  !(missing(se) && missing(sd) && missing(n)),
  paste("normal density: se =", round(se,3)),
  "Standard Normal Density N(0,1)"),
  se=sd/sqrt(n), sd=1, n=1, ...)

norm.curve(mean=0, se=sd/sqrt(n),
  critical.values=mean + se*c(-1.96, 1.96),
  z=do.call("seq",
    as.list(c((par()$usr[1:2]-mean)/se, length=109))),
  shade, col=par("col"), axis.name="z", sd=1, n=1, ...)

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.

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.

z
  z-values (standardized to N(0,1)) used as base of plot.

shade
  Valid values for shade are "right", "left", "inside", "outside". Default is
  "right" for one-sided critical.values and "outside" for two-sided critical val-
  ues.

col
  color of the shaded region.

axis.name
  "z" for the standard normal scale centered on the null hypothesi
  s value of the mean. "z1" for the standard normal scale centered on the alternate
  hypothesis value of the mean.

... 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.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=2)

norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(100, 5, 100+5*c(-1.96, 1.96))

norm.setup(xlim=c(-3, 6))
norm.curve(crit=1.645, mean=1.645+1.281552, col=3,
          shade="left", axis.name="z1")
norm.curve(crit=1.645, col=2)

norm.setup(xlim=c(-6, 12), se=2)
norm.curve(crit=2*1.645, se=2, mean=2*(1.645+1.281552),
          col=3, shade="left", axis.name="z1")
norm.curve(crit=2*1.645, se=2, mean=0,
          col=2, 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))

## naively scaled,
## areas under the curve are numerically the same but visually different
norm.setup(n=1)
norm.curve(n=1)
norm.setup(n=2)
norm.curve(n=2)
norm.setup(n=4)
norm.curve(n=4)
norm.setup(n=10)
norm.curve(n=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.setup(n=2, ylim=c(0,1.3))
norm.curve(n=2)
norm.setup(n=4, ylim=c(0,1.3))
norm.curve(n=4)
norm.setup(n=10, ylim=c(0,1.3))
norm.curve(n=10)
mtext("all areas under the curve are numerically and visually the same",
    side=3, outer=TRUE)
par(mfrow=c(1,1))
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**

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

```r
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.

- `...`  
  other arguments, currently ignored.

- `legend.x`  
  x position of left-hand side of legend.

- `oma`  
  outer margin `par()$oma`, needed to make room for legend.

**Value**

`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.

- `omegal1, omega2`  
  odds for each row, p/(1-p)

- `psihat`  
  odds ratio, omega2/omega1

- `s.ln.psihat`  
  standard deviation of ln(psihat)

- `ci.ln.psihat`  
  confidence interval for ln(psihat) using normal approximation

- `ci.psihat`  
  confidence interval for psihat calculated ase p(ci.ln.psihat)

- `prob1`  
  seq(0,1,.05), set of p1 values for plotting.

- `odds1`  
  p1/(1-p1)
odds2  odds for the second row needed to retain psihat with the specified odds1, calculated as odds1*psihat.
ci.odds2  confidence interval for odds2
prob2  odds2 / (odds2+1)
ci.prob2  ci.odds2 / (ci.odds2+1)

Author(s)
Richard M. Heiberger <rmh@temple.edu>

References

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)

panel.bwplot.hh  Panel functions for bwplot.

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

Usage
panel.bwplot.hh(x, y, box.ratio = 1, horizontal = TRUE, pch = box.dot$pch,
  col = box.dot$col, cex = box.dot$cex, fill = box.rectangle$fill,
  varwidth = FALSE, levels.fos = NULL, coef = 1.5, at, ...)

panel.bwplot.intermediate.hh(x, y, box.ratio = 1, horizontal = TRUE,
  pch = box.dot$pch, col = box.dot$col, cex = box.dot$cex,
  fill = box.rectangle$fill, levels.fos = NULL, at, ...)

32
Arguments

- `x`, `y`, `pch`, `col`, `cex`, `fill`, `varwidth`, `horizontal`, `levels.fos`, `box.ratio`

- `coef` see `boxplot.stats`
- `at` x location of the individual boxes.
- `...` Extra arguments, if any, for 'panel.bwplot'.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

- `panel.xyplot`, `bwplot`, `interaction2wt`

panel.cartesian  
trellis panel function, with labeled rows and columns and without strip labels.

Description

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

Usage

```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

```r
#$---- Should be DIRECTLY executable !! ----#$
#$- == Define data, use random, #$
#$- or do help(data=index) for the standard data sets.

rent <- read.table(hh("datasets/rent.dat"), ## Weisbergs 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)
xysplom(resid(rent.lm) ~ rnt.till + cow.dens | lime, data=rent,
layout=c(2,2))
```

34
xysplom(resid(rent.lm) ~ rnt.till + cow.dens | lime, data=rent,
layout=c(2,2),
xbab="", 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.lm) ~ rnt.till + cow.dens | lime, data=rent,
layout=c(2,2),
xbab="", 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.lm) ~ rnt.till + cow.dens | lime, data=rent,
layout=c(2,2),
xbab="", ylab="",
x.label="", y.label="",
group.label.side="",
par.strip.text=list(cex=1.2),
panel=panel.cartesian,
axis3.line=3.6,
scales=list(
  relation="free",
  alternating=FALSE, labels=FALSE, ticks=FALSE),
between=list(x=1, y=3))

xysplom(resid(rent.lm) ~ rnt.till + cow.dens | lime, data=rent,
layout=c(2,2),
xbab="", ylab="",
y.label="resid",
group.label.side="top",
par.strip.text=list(cex=1.2),
panel=panel.cartesian,
axis3.line=3.6,
scales=list(alternating=FALSE, labels=FALSE, ticks=FALSE),
rescale=list(x=FALSE, y=FALSE),
between=list(x=1, y=5))
panel.ci.plot  Default Panel Function for ci.plot

Description

This is the default panel function for ci.plot.

Usage

panel.ci.plot(x, y, newdata, newfit = newfit, ...)

Arguments

x Observed values of predictor variable.
y Observed values of response variable.
newdata x values for which predictions are calculated.
newfit data.frame containing six components: "fit", "se.fit", "residual.scale", "df", "ci.fit", "pi.fit". In S-Plus these are the output from the predict.lm function. In R they are a rearrangement of the output of the predict.lm function.
...
other arguments passed to panel.xyplot.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

ci.plot, xyplot, lm

panel.interaction2wt  Plot all main effects and twoway interactions in a multifactor design

Description

This is the panel function for interaction2wt. The main diagonal displays boxplots for the main effects of each factor. The off-diagonals show the interaction plots for each pair of factors. The i,j panel shows the same factors as the j,i but with the trace- and x-factor roles interchanged.
Usage

panel.interaction2wt(x, y, subscripts, responselab, trace.values, factor.levels, fun = mean, ...)

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 \texttt{mean}
... extra arguments, primarily color, to be passed to \texttt{panel.bwplot.intermediate.hh}
which.given, which.panel, var.name, factor.levels, shingle.intervals see documentation for \texttt{strip.default}
strip.names Force \texttt{strip.names=TRUE}
style force \texttt{style=1}

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

interaction2wt, panel.bwplot.intermediate.hh

Examples

\begin{verbatim}
anova(aov(len ~ supp*dose, data=ToothGrowth))
interaction2wt(len ~ supp + dose, data=ToothGrowth)
bwplot(len ~ supp | dose, data=ToothGrowth, layout=c(3,1))
\end{verbatim}
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`.  

---

38
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.

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.

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
y.name
group.name

... 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
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$ylabel,
    focus=none$focus,
    main= main.method.phrase,
    main2=main2.method.phrase,
    main.method.phrase=
        paste("multiple comparisons of means of", ylab),
    main2.method.phrase=paste("simultaneous ",
        100*(1-none$alpha),"% confidence limits", ",
        method, " method", sep=" "),
    ry.mmc=TRUE,
    key.x=par()$usr[1]+ diff(par()$usr[1:2])/20,
    key.y=par()$usr[3]+ diff(par()$usr[3:4])/3,
    method=if (is.null(mca)) lmat$method else mca$method,
    print.lmat=!is.null(lmat),
    print.mca=(!is.null(mca) && (!print.lmat)),
    iso.name=TRUE,
    x.offset=0,
    col.mca.signif=8, col.mca.not.signif=1,
    lty.mca.signif=1, lty.mca.not.signif=6,
    lwd.mca.signif=1, lwd.mca.not.signif=1,
    col.lmat.signif=8, col.lmat.not.signif=1,
    lty.lmat.signif=1, lty.lmat.not.signif=6,
    lwd.lmat.signif=1, lwd.lmat.not.signif=1,
    lty.iso=7, col.iso=1, lwd.iso=1,
    lty.contr0=2, col.contr0=1, lwd.contr0=1,
    decdigits.ybar=2,
)

Arguments

x mmc.multicomp object

xlab "contrast value". An alternate "" can help unclutter a figure when
several MMC plots are displayed together.

ylab name of response variable

focus define the factor to compute contrasts of.

main, main2 main and second line of title of plot
main.method.phrase, main2.method.phrase

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 \texttt{simint} for the list.

\texttt{print.lmat}

logical. If \texttt{TRUE}, then display the user-specified contrasts.

\texttt{print.mca}

logical. If \texttt{TRUE}, then display the pair-wise contrasts.

\texttt{iso.name}

logical. If \texttt{TRUE}, label the isomeans grid with the factor levels. If \texttt{FALSE},
label the isomeans grid with sequential numbers and display a key relating
the numbers to the factor levels.

\texttt{x.offset}

amount to move the vertical 0 line to the left or right to reduce overprint-
ing of labels and plotted lines.

\texttt{col.mca.signif}, \texttt{lty.mca.signif}, \texttt{lwd.mca.signif}

\texttt{col.mca.not.signif}, \texttt{lty.mca.not.signif}, \texttt{lwd.mca.not.signif}

\texttt{col.lmat.signif}, \texttt{lty.lmat.signif}, \texttt{lwd.lmat.signif}

\texttt{col.lmat.not.signif}, \texttt{lty.lmat.not.signif}, \texttt{lwd.lmat.not.signif}

color, line type, line width for significant user-specified contrasts.

\texttt{col.iso}, \texttt{col.contr0}, \texttt{lwd.iso}

color, line type, line width for the isomeans grid.

\texttt{lty.iso}, \texttt{col.contr0}, \texttt{lwd.contr0}

color, line type, line width for the vertical contrast=0 line.

\texttt{decdigits.ybar}

number of decimal digits in the left-axis labels.

\texttt{...}

other arguments, currently ignored.

\textbf{Note}

\texttt{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.

\textbf{Author(s)}

Richard M. Heiberger <rmh@temple.edu>

\textbf{References}


Richard M. Heiberger and Burt Holland (2006, accepted), "Mean–Mean Multiple Compari-
on Displays for Families of Linear Contrasts", \textit{Journal of Computational and Graphical
Statistics}.

See Also

`simint.mmc` in R, `mmc` in S-Plus.

Examples

```r
## continue with the example in simint.mmc in R, or 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)
class(catalystm1.aov) <- rev(class(catalystm1.aov))
catalystm.mca <- 
  if.R(r=simint(concent ~ catalyst, data=catalystm, type="Tukey"), 
    s=multicomp(catalystm1.aov))

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))
dimnames(catalystm.lmat)[[1]] <-
  if.R(s=dimnames(catalystm.mca$lmat)[[1]],
    r=c("(Intercept)", dimnames(catalystm.mca$cmatrix)[[2]][-1]))

if.R(r={
  catalystm.mmc <- simint.mmc(concent ~ catalyst, data=catalystm,
    lmat=catalystm.lmat, lmat.rows=2:5,
    type="Tukey", whichf="catalyst")
    catalystm.mmc <- multicomp.label.change(catalystm.mmc, "catalyst", ")
  },
  s={
    catalystm.mmc <- multicomp.mmc(catalystm1.aov, lmat=catalystm.lmat)
  })

gray <- if.R(r="gray", s=16)
red <- if.R(r="red", s=8)
blue <- if.R(r="blue", s=6)

old.mar <- if.R(s=par(mar=c(5,12,4,6)+.1),
               r=par(mar=c(5,6,4,4)+.1))

## 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, 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),
```

---

**43**
## both pairwise contrasts and user-specified contrasts

```r
plot(catalystm.mmc, x.offset=1.6, ry.mmc=c(50,58), lty.iso=2,
     col.iso=gray, print.mca=TRUE,
     col.lmat.signif=blue, col.mca.signif=red)

par(old.mar)
```

---

### 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

```r
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 straight line fit and display of squared residuals

### Usage

```r
regr1.plot(x, y, model=lm(y~x), coef.model=coef(model),
            main="put a useful title here",
            xlab=deparse(substitute(x)),
            ylab=deparse(substitute(y)),
            jitter.x=FALSE,
            resid.plot=FALSE,
            points.yhat=TRUE,
            ..., length.x.set=51,
            err=-1)
```

### Arguments

- **x**: x variable  
- **y**: y variable  
- **model**: Defaults to the simple linear model `lm(y ~ x)`. Any linear model object with one x variable, such as the quadratic `lm(y ~ x + I(x^2))` can be used.  
- **coef.model**: Defaults to the coefficients of the `model` argument. Other coefficients can be entered to illustrate the sense in which they are not "least squares".  
- **main**, **xlab**, **ylab**: arguments to `plot`.  
- **jitter.x**: logical. If `TRUE`, the x is jittered before plotting. Jittering is often helpful when there are multiple y-values at the same level of x.  
- **resid.plot**: If `FALSE`, then do not plot the residuals. If "square", then call `resid.squares` to plot the squared residuals. If `TRUE` (or anything else), then call `resid.squares` to plot straight lines for the residuals.  
- **points.yhat**: logical. If `TRUE`, the predicted values are plotted.  
- **...**: other arguments.  
- **length.x.set**: number of points used to plot the predicted values.  
- **err**:  

### Note

This plot is designed as a pedagogical example for introductory courses. When `resid.plot="square"`, then we actually see the set of squares for which the sum of their areas is minimized by the method of "least squares".

45
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))
```

---

**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, xlab=c(22,73), ylab=c(0,3400))
abline(hardness.lin.lm)
resid.squares(hardness$density, hardness$hardness, predict(hardness.lin.lm), resid.plot = "line")
simint.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

```r
simint.mean(y, n, ybar, s, alpha=.05, ## R
ylabel="ylabel", focus.name="focus.factor", plot=FALSE,
1mat, labels=NULL,
method="Tukey",
bounds="both",
df=sum(n) - length(n),
sigmahat=(sum((n-1)*s^2) / df)^.5,
contrasts, ..., group=y)
```

```r
simint.mmc.mean(y, n, ybar, s, alpha=.05, ## R
ylabel="ylabel", focus.name="focus.factor", plot=FALSE,
1mat, labels=NULL,
method="Tukey",
bounds="both",
df=sum(n) - length(n),
sigmahat=(sum((n-1)*s^2) / df)^.5,
estimate.sign=1,
order.contrasts=TRUE, ..., group=y)
```

```r
cmulticomp.mean(group, n, ybar, s, alpha=.05, ## S-Plus
ylabel="ylabel", focus.name="focus.factor", plot=FALSE,
1mat, labels=NULL, ...

df=sum(n) - length(n),
sigmahat=(sum((n-1)*s^2) / df)^.5)
```

```r
cmulticomp.mmc.mean(group, n, ybar, s, ylabel, focus.name, ## S-Plus
lmat, ...
comparisons="mca",
1mat.rows=seq(length=length(ybar)),
ry,
plot=TRUE,
crit.point, iso.name=TRUE,
estimate.sign=1,
x.offset=0,
```

48
order.contrasts=TRUE,
method="tukey",
df=sum(n)-length(n),
sigmahat=(sum((n-1)*s^2)/df)^.5)

Arguments

group, y  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
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 simint
bounds  type of intervals to compute. This is the "bounds" argument to multicomp and the alternative argument to simint. Values are: the default "both" for two-sided intervals; "lower" for intervals with infinite upper bounds; and, "upper" for intervals with infinite lower bounds. In R, the S-Plus values are translated to "two.sided", "greater", and "less". Or the user can enter the values "two.sided", "greater", and "less".
df  scalar, residual degrees of freedom
sigmahat  sqrt(MSE) from the ANOVA table
contrasts  logical, argument in R to contr.Dunnett when method="Dunnett".
...  other arguments
comparisons  argument to S-Plus multicomp only.
estimate.sign, order.contrasts, lmat.rows
See lmat.rows in multicomp.mmc or simint.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 simint.
iso.name, x.offset
See plot.mmc.multicomp.

Value

simint.mmc.mean and multicomp.mmc.mean return a "mmc.multicomp" object.
simint.mean returns a "hmtest" 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 `simint.mmc` based on `simint`. 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


See Also

`simint.mmc`, `multicomp.mmc`

Examples

```r
## This example is from Hsu and Peruggia
pulmonary <- read.table(hh("datasets/pulmonary.dat"), header=TRUE,
                        row.names="group")
pulmonary

anova.mean(row.names(pulmonary),
           pulmonary$n,
pulmonary$ybar,
pulmonary$s,
           ylabel="pulmonary")

## simint or multicomp object
pulmonary.mca <-
  if.R(r=
    simint.mean(row.names(pulmonary),
                pulmonary$n,
pulmonary$ybar,
pulmonary$s,
                ylabel="pulmonary",
                focus="smoker")
  ,s=
    multicomp.mean(row.names(pulmonary),
                   pulmonary$n,
pulmonary$ybar,
                   "pulmonary",
                   focus="smoker")
```
pulmonary$s,
ylabel="pulmonary",
focus="smoker")
}

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

pulm.lmat <- cbind("npnl-mh"=c( 1, 1, 1, 1,-2,-2), ## not.much vs lots
"n-ndl" =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)
pulm.lmat
## mmc.multicomp object
pulmonary.mmc <-
if.R(r=
simint.mmc.mean(row.names(pulmonary),
pulmonary$n,
pulmonary$ybar,
pulmonary$s,
ylabel="pulmonary",
focus="smoker",
lmat=pulm.lmat)
,
,s=
multicomp.mmc.mean(row.names(pulmonary),
pulmonary$n,
pulmonary$ybar,
pulmonary$s,
ylabel="pulmonary",
focus="smoker",
lmat=pulm.lmat,
plot=FALSE)
)

gray <- if.R(r="gray", s=16)
red <- if.R(r="red", s=8)
blue <- if.R(r="blue", s=6)
old.par <- if.R(s=par(mar=c(5,4,4,4)+.1),
 r=par(mar=c(15,4,4,4)+.1))
## pairwise comparisons
plot(pulmonary.mmc, print.mca=TRUE, print.lmat=FALSE,
col.mca.signif=red, col.iso=16)
## tiebreaker plot, with contrasts ordered to match MMC plot,
## with all contrasts forced positive, and with names also reversed.
if.R(r={
simint.mmc

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

**Description**

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

**Usage**

R

```r
simint.mmc(y, ## R
data, 
type = "Tukey", 
lmat=NULL, 
lmat.rows=2:nrow(mca.lmat), 
lmat.scale.abs2=TRUE, 
estimate.sign=1, 
order.contrasts=TRUE, 
whichf, 
cmatrix=t(lmat), 
covariates, 
...)
```

S-Plus

```r
multicomp.mmc(..., comparisons="mca", ##S-Plus
lmat, lmat.rows=-1, 
lmat.scale.abs2=TRUE, 
ry, 
plot=TRUE, 
crit.point, 
iso.name=TRUE,
```

52
estimate.sign=1,
x.offset=0,
order.contrasts=TRUE,
main,
main2)

"[..mmc.multicomp"(x, ..., drop = TRUE)

Arguments

y  Analysis of variance formula.
data  data.frame
type  type of contrasts. See simint for the list.
lmat  contrast matrix as in the S-Plus multicomp. The convention for lmat in R is to use the transpose of the cmatrix component produced by simint. Required for user-specified contrasts.
lmat.rows  rows in lmat for the whichf factor.
whichf  define the factor to compute contrasts of. See simint. This argument is called focus in multicomp.
cmatrix  transpose of the "lmat" argument.
covariates  The current version of multicomp (0.4-8 in R-2.3.1) doesn't handle covariates correctly.
...  other arguments. alternative and base are frequently used with simint.
comparisons  argument to multicomp
lmat.scale.abs2  logical, scale the contrasts in the columns of lmat to make the sum of the absolute values of each column equal 2.
estimate.sign  numeric. If 0, leave contrasts in the default lexicographic direction. If positive, force all contrasts to positive, reversing their names if needed (if contrast A-B is negative, reverse it to B-A). If negative, the force all contrasts to positive.
order.contrasts  sort the contrasts in the (mca, none, lmat) components by height on the MMC plot. This will place the contrasts in the multicomp plots in the same order as in the MMC plot.
crit.point  critical value for the tests. The value from the specified multicomp method is used for the user-specified contrasts when lmat is specified. This argument is not available with simint in R.
plot  logical, display the plot if TRUE.
ry, iso.name, x.offset, main, main2  arguments to plot.mmc.multicomp.
x, drop  See "[".
Details

By default we plot the isomeans grid and the pairwise comparisons. We get the right contrasts automatically if the aov is oneway. If we specify an lmat for oneway it must have a leading row of 0.

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

simint in R multcomp version 0.4-8 doesn’t work correctly with covariates.

simint.mmc works from a formula, not from an "aov" object as multicomp.mmc in S-Plus does.

Value

In R, an "mmc.multicomp" object containing either the first three, or all five, of the following components:

- **mca**: "multicomp" object containing the pairwise comparisons.
- **none**: "multicomp" object comparing each mean to 0.
- **hmtest**: "hmtest" object from simint for the pairwise comparisons.
- **lmat**: "multicomp" object for the contrasts specified in the lmat argument.
- **lmat.hmtest**: "hmtest" object from simint 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 simint.mmc based on simint. 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


See Also

`as.multicomp`

Examples

```r
## simint is strictly for 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 ~ concent, data=catalystm, 
    scales=list(cex=1.5), 
    xlab=list("concentration", cex=1.5), 
    ylab=list("catalyst",cex=1.5)))
}

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

catalystm.mca <-
  if.R(r=simint(concent ~ catalyst, data=catalystm, type="Tukey"), 
    s=multicomp(catalystm1.aov, plot=F))
plot(catalystm.mca)
catalystm.mca

## pairwise comparisons

catalystm.mmc <-
  if.R(r=simint.mmc(concent ~ catalyst, data=catalystm), 
    s=multicomp.mmc(catalystm1.aov, plot=F))
  if.R(r=catalystm.mmc <-
        multicomp.label.change(catalystm.mmc, "catalyst", ""), 
    s={})
catalystm.mmc
plot(catalystm.mmc)

if.R(r=plot(catalystm.mmc, col.mca.signif="red"), 
  s={})
plot(catalystm.mmc$mca)
plot(catalystm.mmc$none)
```

55
## $none works for oneway ANOVA, not sure yet beyond that

### 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))
dimnames(catalystm.lmat)[[1]] <- dimnames(catalystm.mmc$mca$lmat)[[1]]
zapsmall(catalystm.lmat)
if.R(s=dimnames(catalystm.mca$lmat)[[1]],
 r=c("(Intercept)"*, dimnames(catalystm.mca$cmatrix)[[2]][-1]))

if.R(r={
catalystm.mmc <- simint.mmc(concent ~ catalyst, data=catalystm,
 lmat=catalystm.lmat, lmat.rows=2:5,
 type="Tukey", whichf="catalyst")
catalystm.mmc <- multicomp.label.change(catalystm.mmc, "catalyst", ")
},
s={
catalystm.mmc <- multicomp.mmc(catalystm1.aov, lmat=catalystm.lmat,
 plot=FALSE)
})
catalystm.mmc
plot(catalystm.mmc)
if.R(r=plot(catalystm.mmc, col.lmat.signif="red"),
 s={})

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

### Dunnett s test

weightloss <- read.table(hh("datasets/weightloss.dat"), header=TRUE)
weightloss <- data.frame(loss=unlist(weightloss),
                        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)
tmp.dunnett <-
if.R(r=
  simint(loss ~ group, data=weightloss,
    type="Dunnett", alternative="greater", base=4)
  ,s=
  multicomp(weightloss.aov,
    method="dunnett", comparisons="mcc",
    bounds="lower", control=4,
    valid.check=FALSE)
  )
plot(tmp.dunnett)
if.R(r={
  tmp.dunnett.mmc <-
  simint.mmc(loss ~ group, data=weightloss,
    type="Dunnett", alternative="greater", base=4)
  tmp.dunnett.mmc <- multicomp.label.change(tmp.dunnett.mmc, "group", "")
},s=
  tmp.dunnett.mmc <-
  multicomp.mmc(weightloss.aov,
    method="dunnett", comparisons="mcc",
    bounds="lower", control=4,
    valid.check=FALSE, plot=FALSE)
  )

plot(tmp.dunnett.mmc)

---

**simint.reverse**

*Force all comparisons in a "hmtest" from simint to have the same sign.*

### Description

Force all comparisons in a "hmtest" from simint 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

```r
simint.reverse(y, estimate.sign = 1, ...)  # R

multicomp.reverse(y, estimate.sign = 1, ...)  # S-Plus
```

### Arguments

- `y`  
  "simint" 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 "simint" 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

```r
## see example in multicomp.order

strip.xysplom
```

Description

strip function that is able to place the correlation or regression coefficient into the strip label.

Usage

```r
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

t.trellis

Interchange the x- and y-axes for an S-Plus trellis object. Identity function for an R lattice object.

Description

In S-Plus, change the "trellis" object to effectively, and after-the-fact convert formulas from \((y ~ x | g)\) to \((x ~ y | g)\). This is needed in S-Plus because most S-Plus trellis functions do not permit factors on the right-hand side of the formula.

In R lattice functions, factors are permitted on the right-hand side of the formula. Therefore we make t.trellis an identity operator in R.

Usage

t.trellis(x)

Arguments

x any "trellis" object.

Details

In S-Plus, a warning is generated if the panel component of the "trellis" object is a function. No warning is generated when the panel component of the "trellis" object is a character string naming the function. We interchange the x and y values of the \(c("x", "y", "xlab", "ylab", "xlim", "ylim", "adj.xlim", "adj.ylim")\) components of the "trellis" object, and are aware of special features of the panel functions listed in the S-Plus version of t.trellis. The transpose will usually work correctly for other panel functions.

Value

In S-Plus, a "trellis" object with all x and y components interchanged. In R, the argument "trellis" object is returned unchanged.
vif  

*Calculate the Variance Inflation Factor*

**Description**

The VIF for predictor $i$ is $1/(1 - R^2_i)$, where $R^2_i$ is the $R^2$ from a regression of predictor $i$ against the remaining predictors.

**Usage**

```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))
)
```

```r
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
```

**Referenced**


**See Also**

In R: `xyplot`. In S-Plus: `xyplot`.
Arguments

- `x` : data.frame, or formula, or lm object.
- `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 $1/(1-R^2_i)$, where $R^2_i$ is the $R^2$ from a regression of predictor $i$ against the remaining predictors. If $R^2_i$ 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`

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)
```
vif(usair.lm)

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

vif(usair)

vif(usair, y.name="lnSO2")

---

X.residuals

Residuals from the regression of each column of a data.frame against all the other columns.

Description

Calculate the residuals from the regression of each column of a data.frame against all the other columns.

Usage

X.residuals(x, ...)

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

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

## S3 method for class lm :
X.residuals(x, na.action = na.exclude, ...) ## x is a "lm" object

Arguments

x  data.frame, or formula, or lm object.
na.action  See na.action.
...  additional arguments.
y.name  Name of Y-variable to be excluded from the computations.
data  A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.

Value

Data.frame of residuals, one column from each regression.

Author(s)

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

See Also
lm, vif, residual.plots.

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.residuals(usair.lm)
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**

```r
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 num-
ber 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 ~ x + 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, corr=TRUE, beta=TRUE)
xysplom(y + w ~ x , data=tmp, abline=TRUE)
xysplom(y + w ~ x , data=tmp, corr=TRUE, abline=FALSE)
```r
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, 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))

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

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=
    {})
```

66