# Package ‘HH’

October 12, 2022

**Type** Package

**Title** Statistical Analysis and Data Display: Heiberger and Holland

**Version** 3.1-49

**Date** 2022-08-09

**Author** Richard M. Heiberger

**Maintainer** Richard M. Heiberger &lt;rmh@temple.edu&gt;

**Depends** R (&gt;= 3.0.2), lattice, stats, grid, latticeExtra, multcomp, gridExtra (&gt;= 2.0.0), graphics

**Imports** reshape2, leaps, vcd, colorspace, RColorBrewer, shiny (&gt;= 0.13.1), Hmisc, abind, Rmpfr (&gt;= 0.6.0), grDevices, methods

**Suggests** mvtnorm, car, Rcmdr, RcmdrPlugin.HH, TeachingDemos, microplot

**Description** Support software for Statistical Analysis and Data Display (Second Edition, Springer, ISBN 978-1-4939-2121-8, 2015) and (First Edition, Springer, ISBN 0-387-40270-5, 2004) by Richard M. Heiberger and Burt Holland. This contemporary presentation of statistical methods features extensive use of graphical displays for exploring data and for displaying the analysis. The second edition includes redesigned graphics and additional chapters. The authors emphasize how to construct and interpret graphs, discuss principles of graphical design, and show how accompanying traditional tabular results are used to confirm the visual impressions derived directly from the graphs. Many of the graphical formats are novel and appear here for the first time in print. All chapters have exercises. All functions introduced in the book are in the package. R code for all examples, both graphs and tables, in the book is included in the scripts directory of the package.

**License** GPL (&gt;= 2)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-08-09 16:10:07 UTC

---

## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>HH-package</td>
<td>5</td>
</tr>
<tr>
<td>ae.dotplot</td>
<td>12</td>
</tr>
<tr>
<td>R topics documented</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
</tr>
<tr>
<td>AEdotplot</td>
<td>17</td>
</tr>
<tr>
<td>AEdotplot.data.frame</td>
<td>20</td>
</tr>
<tr>
<td>ancova</td>
<td>25</td>
</tr>
<tr>
<td>ancova-class</td>
<td>28</td>
</tr>
<tr>
<td>ancovaplot</td>
<td>29</td>
</tr>
<tr>
<td>anova.ancovaplot</td>
<td>32</td>
</tr>
<tr>
<td>anovaMean</td>
<td>33</td>
</tr>
<tr>
<td>aovSufficient</td>
<td>34</td>
</tr>
<tr>
<td>arima.diag.hh</td>
<td>36</td>
</tr>
<tr>
<td>arma.loop</td>
<td>37</td>
</tr>
<tr>
<td>as.likert</td>
<td>39</td>
</tr>
<tr>
<td>as.matrix.listOfNamedMatrices</td>
<td>41</td>
</tr>
<tr>
<td>as.multicomp</td>
<td>44</td>
</tr>
<tr>
<td>as.vector.trellis</td>
<td>47</td>
</tr>
<tr>
<td>axis.i2wt</td>
<td>48</td>
</tr>
<tr>
<td>bivariateNormal</td>
<td>48</td>
</tr>
<tr>
<td>ci.plot</td>
<td>49</td>
</tr>
<tr>
<td>CIplot</td>
<td>51</td>
</tr>
<tr>
<td>col.hh</td>
<td>54</td>
</tr>
<tr>
<td>col3x2</td>
<td>55</td>
</tr>
<tr>
<td>combineLimits.trellisvector</td>
<td>56</td>
</tr>
<tr>
<td>cp.calc</td>
<td>57</td>
</tr>
<tr>
<td>cplx</td>
<td>60</td>
</tr>
<tr>
<td>datasets</td>
<td>60</td>
</tr>
<tr>
<td>dchisq.intermediate</td>
<td>61</td>
</tr>
<tr>
<td>defunct</td>
<td>62</td>
</tr>
<tr>
<td>diag.maybe.null</td>
<td>63</td>
</tr>
<tr>
<td>diagplot5new</td>
<td>64</td>
</tr>
<tr>
<td>diagQQ</td>
<td>65</td>
</tr>
<tr>
<td>Discrete4</td>
<td>66</td>
</tr>
<tr>
<td>do.formula.trellis.xysplom</td>
<td>67</td>
</tr>
<tr>
<td>EmphasizeVerticalPanels</td>
<td>68</td>
</tr>
<tr>
<td>emptyMainLeftAxisLeftStripBottomLegend</td>
<td>69</td>
</tr>
<tr>
<td>export.eps</td>
<td>70</td>
</tr>
<tr>
<td>extra</td>
<td>71</td>
</tr>
<tr>
<td>F.curve</td>
<td>72</td>
</tr>
<tr>
<td>glhtWithMCP993</td>
<td>75</td>
</tr>
<tr>
<td>gof.calculation</td>
<td>76</td>
</tr>
<tr>
<td>grid.yaxis.hh</td>
<td>77</td>
</tr>
<tr>
<td>GSremove</td>
<td>78</td>
</tr>
<tr>
<td>HH.regsubsets</td>
<td>78</td>
</tr>
<tr>
<td>hhpdf</td>
<td>80</td>
</tr>
<tr>
<td>HHscriptnames</td>
<td>81</td>
</tr>
<tr>
<td>hov</td>
<td>82</td>
</tr>
<tr>
<td>hovBF</td>
<td>84</td>
</tr>
<tr>
<td>if.R</td>
<td>85</td>
</tr>
<tr>
<td>InsertVerticalPanels</td>
<td>86</td>
</tr>
<tr>
<td>interaction.positioned</td>
<td>87</td>
</tr>
</tbody>
</table>
topics documented:

interaction2wt ........................................................................... 88
interval .................................................................................... 92
intxplot .................................................................................... 93
ladder ....................................................................................... 96
latex.array ............................................................................... 99
latticeresids ............................................................................ 103
legendGrob2wt ......................................................................... 104
likert ......................................................................................... 104
likertColor ............................................................................... 120
likertMosaic ............................................................................ 123
LikertPercentCountColumns ..................................................... 128
likertWeighted ......................................................................... 130
likertWeighted47 ...................................................................... 133
lm.case ....................................................................................... 136
lm.regsubsets .......................................................................... 139
lmatPairwise ............................................................................ 140
lmatRows .................................................................................. 141
lm.plot ....................................................................................... 142
logit ........................................................................................ 143
matrix.trellis ........................................................................... 144
mcalinfect ................................................................................. 146
mmc .......................................................................................... 147
mmc.mean .................................................................................. 155
mmc_aspect ............................................................................. 159
mmcisomeans ........................................................................... 159
mmcplot ..................................................................................... 162
mmcPruneIsomeans .................................................................. 164
multicomp.order ...................................................................... 165
multicomp.reverse ................................................................... 167
norm.curve ............................................................................... 169
NormalAndTplot ......................................................................... 175
NormalAndTPower ................................................................... 182
normalApproxBinomial .............................................................. 184
np.arima .................................................................................... 185
NTplot ........................................................................................ 186
objip .......................................................................................... 189
OddsRatio .................................................................................. 190
OneWayVarPlot ......................................................................... 192
orthog.complete ....................................................................... 193
panel.acf .................................................................................. 196
panel.axis.right ......................................................................... 196
panel.bwplot.intermediate.hh .................................................. 197
panel.bwplot.superpose ............................................................ 198
panel.bwplott ............................................................................ 201
panel.cartesian ........................................................................ 202
panel.ci.plot ............................................................................. 204
panel.confintMMC .................................................................... 205
panel.dotplot.tb ....................................................................... 206
R topics documented:

panel.interaction2wt .................................................. 208
panel.isomeans ............................................................. 210
panel.likert ................................................................. 211
panel.pairs.hh ............................................................... 213
panel.xysplom ............................................................... 214
partial.corr ................................................................. 214
pdf.latex ................................................................. 215
pdiscunif ................................................................. 216
perspPlane ................................................................. 217
plot.hov ................................................................. 218
plot.mmc.multicomp ..................................................... 220
plot.multicomp ........................................................... 223
position ................................................................. 226
positioned-class ........................................................ 230
print.lattice.resids ..................................................... 231
print.NormalAndTplot .................................................. 232
print.tsdiagplot ........................................................ 233
print.TwoTrellisColumns ............................................. 234
push.vp.hh ................................................................. 238
pyramidLikert ............................................................ 239
rbind.trellis .............................................................. 241
regr1.plot ................................................................. 244
regr2.plot ................................................................. 246
regrresidplot ............................................................. 248
resid.squares .............................................................. 249
residual.plots ............................................................ 250
residual.plots.lattice ................................................... 252
residVSfitted ............................................................. 253
ResizeEtc ................................................................. 254
ResizeEtc.likertPlot .................................................... 257
rowPcts ................................................................. 258
seqplot ................................................................. 259
seqplot.forecast ........................................................ 260
strip.background0 ....................................................... 261
strip.useOuterStrips.first ............................................. 262
strip.xysplom ............................................................. 263
sufficient ................................................................. 264
summary.arma.loop ...................................................... 265
ToBW.likert .............................................................. 266
toCQxR ................................................................. 267
tsacfplots ................................................................. 268
tsdia.gplot ................................................................. 270
useOuterScales .......................................................... 273
useOuterStripsT2L1 ...................................................... 278
vif ................................................................. 279
X.residuals ................................................................. 281
xysplom ................................................................. 282

Index ................................................................. 286
**Description**

Support software for Statistical Analysis and Data Display (Second Edition, Springer, ISBN 978-1-4939-2121-8, 2015) and (First Edition, Springer, ISBN 0-387-40270-5, 2004) by Richard M. Heiberger and Burt Holland. This contemporary presentation of statistical methods features extensive use of graphical displays for exploring data and for displaying the analysis. The second edition includes redesigned graphics and additional chapters. The authors emphasize how to construct and interpret graphs, discuss principles of graphical design, and show how accompanying traditional tabular results are used to confirm the visual impressions derived directly from the graphs. Many of the graphical formats are novel and appear here for the first time in print. All chapters have exercises. All functions introduced in the book are in the package. R code for all examples, both graphs and tables, in the book is included in the scripts directory of the package.

**Details**

The DESCRIPTION file:

- **Package**: HH
- **Type**: Package
- **Title**: Statistical Analysis and Data Display: Heiberger and Holland
- **Version**: 3.1-49
- **Date**: 2022-08-09
- **Author**: Richard M. Heiberger
- **Maintainer**: Richard M. Heiberger <rmh@temple.edu>
- **Depends**: R (>= 3.0.2), lattice, stats, grid, latticeExtra, multcomp, gridExtra (>= 2.0.0), graphics
- **Imports**: reshape2, leaps, vcd, colorspace, RColorBrewer, shiny (>= 0.13.1), Hmisc, abind, Rmpfr (>= 0.6.0), grDevices, methods
- **Suggests**: mvtnorm, car, Rcmdr, RcmdrPlugin.HH, TeachingDemos, microplot
- **Description**: Support software for Statistical Analysis and Data Display (Second Edition, Springer, ISBN 978-1-4939-2121-8, 2015) and (First Edition, Springer, ISBN 0-387-40270-5, 2004) by Richard M. Heiberger and Burt Holland. This contemporary presentation of statistical methods features extensive use of graphical displays for exploring data and for displaying the analysis. The second edition includes redesigned graphics and additional chapters. The authors emphasize how to construct and interpret graphs, discuss principles of graphical design, and show how accompanying traditional tabular results are used to confirm the visual impressions derived directly from the graphs. Many of the graphical formats are novel and appear here for the first time in print. All chapters have exercises. All functions introduced in the book are in the package. R code for all examples, both graphs and tables, in the book is included in the scripts directory of the package.
- **License**: GPL (>= 2)

Index of help topics:

- **AEdotplot**: AE (Adverse Events) dotplot of incidence and relative risk
- **AEdotplot.data.frame**: AE (Adverse Events) dotplot of incidence and relative risk, support functions
- **CIplot**: Illustration of the meaning of confidence levels.
- **Discrete4**: Discrete with four levels color dataset.
- **EmphasizeVerticalPanels**: Helper function for likertWeighted(). used for vertical spacing and horizontal borders of grouped panels.
F.curve  plot a chisquare or a F-curve.
GSremove  Remove selected GraphSheetPages in the S-Plus Windows GUI Graphsheet
HH-defunct  Defunct Functions in Package 'HH'
HH-package  Statistical Analysis and Data Display: Heiberger and Holland
HH.regsubsets  Display tabular results for Best Subsets Regression.
HH.scriptnames  Find absolute pathname of a script file for the HH book in the HH package.
InsertVerticalPanels  Expand a 3D array on the second dimension, inserting empty layers where the input vector has a '0' value. A 2D argument 'x' with 'dim(x)==c(r,c)' is first extended to 3D with 'dim(x)==c(1,r,c)', and then the result is collapsed back to 2D.
LikertPercentCountColumns  Display likert plots with percents in the first column of panels and counts in the second column of panels.
NTplot  Specify plots to illustrate Normal and t Hypothesis Tests or Confidence Intervals, including normal approximation to the binomial.
NormalAndTPower  Construct a power graph based on the NTplot.
NormalAndTplot  Specify plots to illustrate Normal and t Hypothesis Tests or Confidence Intervals.
OddsRatio  Calculate or plot the odds ratio for a 2x2 table of counts.
OneWayVarPlot  Displays a three-panel 'bwplot' of the data by group, of the group means, and of the entire dataset. This is an approximate visualization of the Mean Square lines from the ANOVA table for a one-way ANOVA model.
ResizeEtc  Display multiple independent trellis objects on the same coordinated scale.
ResizeEtc.likertPlot  Display multiple independent trellis objects, representing likert plots, on the same coordinated scale.
ToBW.likert  Change colors in a likert plot to shades of Black and White.
X.residuals  Residuals from the regression of each column of a data.frame against all the other columns.
ae.dotplot  AE (Adverse Events) dotplot of incidence and relative risk.
ancova  Compute and plot oneway analysis of covariance.
ancova-class  Class "ancova" Analysis of Covariance
ancovaplot  Analysis of Covariance Plots
anova.ancovaplot  ANOVA table for a c("ancovaplot","trellis")
HH-package

anovaMean
ANOVA table from the group sample sizes, means, and standard deviations.
anovSufficient
Analysis of variance from sufficient statistics for groups.
arma.diag.hh
Repair design error in S-Plus arima.diag
arma.loop
Loop through a series of ARIMA models and display coordinated tables and diagnostic graphs.
as.likert
Support functions for diverging stacked barcharts for Likert, semantic differential, and rating scale data.
as.matrix.listOfNamedMatrices
Convert a list of numeric matrices to a single matrix
as.multicomp
Support functions in R for MMC (mean-mean multiple comparisons) plots.
as.rts
Miscellaneous functions that I wish were in or consistent between S-Plus and R.
as.vector.trellis
Convert a two-dimensional trellis object into a one-dimensional trellis object. Change the order of panels in a trellis object.
axis.i2wt
specialized axis function for interaction2wt.
bivariateNormal
Plot the bivariate normal density using wireframe for specified rho.
case
case statistics for regression analysis
ci.plot
Plot confidence and prediction intervals for simple linear regression
col.hh
Initializing Trellis Displays
col3x2
col3x2 color dataset
combineLimits.trellisvector
Combine limits on a one-dimensional trellis object.
cp.calc
Rearranges and improves the legibility of the output from the stepwise function in S-Plus.
cplex
Generate a sequence spanning the xlim of a lattice window.
datasets
Datasets for Statistical Analysis and Data Display, Heiberger and Holland
dchisq.intermediate
Intermediate f and chisq functions to simplify writing for both R and S-Plus.
diag.maybe.null
Returns a value for the diagonal of NA and NULL arguments.
diagQQ
QQ plot of regression residuals.
diagplot5new
Transpose of ECDF for centered fitted values and residuals from a linear model.
do.formula.trellis.xysplom
Interprets model formulas for xysplom and
HH-package

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

export.eps
Exports a graph to an EPS file.

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

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

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

hhpdf
R tools for writing HH2: hhpdf, hhdev.off, hhcapture, hhcode, hhpng, hhlatex

hov
Homogeneity of Variance

hovBF
Homogeneity of Variance: Brown-Forsyth method

hovPlot
Homogeneity of Variance Plot

if.R
Conditional Execution for R or S-Plus

interaction.positioned
interaction method for positioned factors.

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

interval
Prediction and Confidence Intervals for glm Objects

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

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

latex.array
Generate the latex code for an "array" or "table" with 3, 4, or more dimensions.

latticeresids
Subroutine used by residual.plots.lattice

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

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

likertColor
Selection of colors for Likert plots.

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

likertWeighted
Special case wrapper for likert() when multiple columns are to have the same bar thicknesses. Uses formula with one or two conditioning variables.

likertWeighted47
Special case wrapper for likert() when multiple columns are to have the same bar thicknesses. Uses formula.
**lm.regsubsets**  Evaluate lm model with highest adjusted $R^2$.

**lm.matPairwise**  lm.matPairwise

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

**lm.plot**  Four types of residual plots for linear models.

**logit**  Logistic and odds functions and their inverses.

**matrix.trellis**  Convert a one-dimensional trellis object to a two-dimensional trellis object. This permits combineLimits and useOuterStrips to work.

**mca.infect**  MCA multiple comparisons analysis (pairwise)

**mmc**  MMC (Mean-mean Multiple Comparisons) plots.

**mmc.mean**  MMC (Mean-mean Multiple Comparisons) plots from the sufficient statistics for a one-way design.

**mmc.aspect**  Control aspect ratio in MMC plots to maintain isomeans grid as a square.

**mmc.prune.isomeans**  MMC plots in lattice-suppress isomeans grid lines for specified levels of the factor.

**mmc.isomeans**  Functions used by mmcplot.

**mmc.plot**  MMC (Mean-mean Multiple Comparisons) plots in lattice.

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

**multicomp.reverse**  Force all comparisons in a "multicomp" object to have the same sign.

**norm.curve**  plot a normal or a t-curve with both x and z axes.

**normalApproxBinomial**  Plots to illustrate Normal Approximation to the Binomial-hypothesis tests or confidence intervals.

**npar.arma**  Count the number of parameters in an ARIMA model specification.

**objip**  loop through all attached directories looking for pattern, possibly restricting to specified class or mode.

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

**panel.acf**  Panel functions for tsdiagplot.

**panel.axis.right**  Right-justify right-axis tick labels.

**panel.bwplot.intermediate.hh**  Panel functions for bwplot.

**panel.bwplot.superpose**  Panel function for bwplot that displays an entire box in the colors coded by groups.

**panel.bwplott**  Extension to S-Plus trellis to allow transposed plots.

**panel.cartesian**  trellis panel function, with labeled rows and columns and without strip labels.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>panel.ci.plot</td>
<td>Default Panel Function for ci.plot</td>
</tr>
<tr>
<td>panel.confintMMC</td>
<td>Confidence interval panel for MMC tiebreaker plots, or confidence interval plot.</td>
</tr>
<tr>
<td>panel.dotplot.tb</td>
<td>Dotplot with evenly spaced tiebreakers.</td>
</tr>
<tr>
<td>panel.interaction2wt</td>
<td>Plot all main effects and two-way interactions in a multifactor design</td>
</tr>
<tr>
<td>panel.isomeans</td>
<td>isomeans grid for MMC plots.</td>
</tr>
<tr>
<td>panel.likert</td>
<td>Panel functions for likert that include a stackWidth argument</td>
</tr>
<tr>
<td>panel.pairs.hh</td>
<td>Function based on S-Plus panel.pairs to add the subpanel.scales and panel.cex arguments.</td>
</tr>
<tr>
<td>panel.xysplom</td>
<td>Panel method for xysplom.</td>
</tr>
<tr>
<td>partial.corr</td>
<td>partial correlations</td>
</tr>
<tr>
<td>pdf.latex</td>
<td>Construct a pdf file from a &quot;latex&quot; file. See Hmisc::latex for concepts.</td>
</tr>
<tr>
<td>pdiscunif</td>
<td>Discrete Uniform Distribution</td>
</tr>
<tr>
<td>perspPlane</td>
<td>Helper functions for regr2.plot</td>
</tr>
<tr>
<td>plot.mmc.multicomp</td>
<td>MMC (Mean-mean Multiple Comparisons) plot.</td>
</tr>
<tr>
<td>plot.multicomp</td>
<td>Multiple comparisons plot that gives independent user control over the appearance of the significant and not significant comparisons.</td>
</tr>
<tr>
<td>position</td>
<td>Find or assign the implied position for graphing the levels of a factor. A new class &quot;positioned&quot;, which inherits from &quot;ordered&quot; and &quot;factor&quot;, is defined.</td>
</tr>
<tr>
<td>positioned-class</td>
<td>Class &quot;positioned&quot;, extends &quot;ordered&quot; to specify the position for graphing the levels of a factor.</td>
</tr>
<tr>
<td>print.NormalAndTplot</td>
<td>Print method for Normal and t plots from NTplot.</td>
</tr>
<tr>
<td>print.TwoTrellisColumns5</td>
<td>Print two conformable trellis plots in adjacent columns with user control of widths.</td>
</tr>
<tr>
<td>print.latticeresids</td>
<td>Print a 'latticeresids' object.</td>
</tr>
<tr>
<td>print.tsdiagplot</td>
<td>Print a &quot;tsdiagplot&quot; object.</td>
</tr>
<tr>
<td>push.vp.hh</td>
<td>push and pop a grid viewport, turn clipping off, change scale.</td>
</tr>
<tr>
<td>pyramidLikert</td>
<td>Print a Likert plot as a Population Triangle</td>
</tr>
<tr>
<td>rbind.trellis</td>
<td>Extend matrix reshaping functions to trellis objects.</td>
</tr>
<tr>
<td>regr1.plot</td>
<td>plot x and y, with optional straight line fit and display of squared residuals</td>
</tr>
<tr>
<td>regr2.plot</td>
<td>3D plot of z against x and y, with regression plane fit and display of squared residuals.</td>
</tr>
<tr>
<td>regrresidplot</td>
<td>Draw a plot of y vs x from a linear model object, with residuals indicated by lines or squares.</td>
</tr>
</tbody>
</table>
HH-package

resid.squares  plot squared residuals in inches to match the y-dimension
residVSfitted  Draw plots of resid ~ y.hat and sqrt(abs(resid)) ~ y.hat
residual.plots  Residual plots for a linear model.
residual.plots.lattice  Construct four sets of regression plots: Y against X, residuals against X, partial residuals against X, partial residuals against each X adjusted for all the other X columns.
rowPcts  Row and columns percents
seqplot  Time series plot.
seqplotForecast  seqplot with confidence bands for the forecast region.
strip.background0  Turn off the coloring in the trellis strip labels. Color 0 is the background color.
strip.useOuterStrips.first  Functions based on strip.default for use with the useOuterScales function.
strip.xysplom  strip function that is able to place the correlation or regression coefficient into the strip label.
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.
summary.arma.loop  summary and print and subscript methods for tsdiagplot and related objects.
toCQxR  Reshape a 3-way array to a 2-way data.frame that can be used with a trellis conditioning formula to get the three-way behavior. Used with likertWeighted().
tsacfplots  Coordinated time series and ACF and PCF plots.
tsdiagplot  Times series diagnostic plots for a structured set of ARIMA models.
useOuterScales  Put scales for axes only on the bottom and left panels of a lattice display, and give fine control over the placement of strips
useOuterStripsT2L1  Three-factor generalization of latticeExtra::useOuterStrips
vif  Calculate the Variance Inflation Factor
xysplom  scatterplot matrix with potentially different sets of variables on the rows and columns.
ae.dotplot

Author(s)

Richard M. Heiberger

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

References


See Also

ancovaplot, ci.plot, interaction2wt, ladder, case.lm, NTplot for Normal and t plots, hov, resid.squares, MMC,
AE.dotplot, likert, tsacfplots, tsdiagplot
demo(package="HH")

Examples

## In addition to the examples for each function,
## there are seven interactive shiny apps in the HH package:
## Not run:
if (interactive()) NTplot(mean0=0, mean1=1, shiny=TRUE)
if (interactive()) shiny::runApp(system.file("shiny/bivariateNormal", package="HH"))
if (interactive()) shiny::runApp(system.file("shiny/bivariateNormalScatterplot", package="HH"))
if (interactive()) shiny::runApp(system.file("shiny/PopulationPyramid", package="HH"))
if (interactive()) shiny::runApp(system.file("shiny/AEdotplot", package="HH"))
if (interactive()) shiny::runApp(system.file("shiny/likert", package="HH"))
## End(Not run)

ae.dotplot

AE (Adverse Events) dotplot of incidence and relative risk

Description

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

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

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

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

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

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

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

aeReshapeToLong(aewide)
```

Arguments

- **ae**
  - For `ae.dotplot`, either a data.frame containing the Adverse Event data in long format as described by the detail for `xr` below, or a data.frame containing the Adverse event data in wide format as described by the detail for `aewide` below.
  - For `logrelrisk`, a data.frame containing the first 4 columns of `xr` described below.

- **...**
  - For `ae.dotplot`, all the arguments listed in the calling sequence for `ae.dotplot.long` and possibly standard panel function arguments.
  - For the other functions, just standard panel function arguments.

- **xr**
  - `RAND`: treatment as randomized (factor).
  - `PREF`: adverse event symptom name (factor).
  - `SN`: number of patients in treatment group.
  - `SAE`: number of patients in each group for whom the event `PREF` was observed.
  - `PCT`: `SAE/SN` as a percent.
  - `relrisk`: Relative risk defined as `PCT` for the B treatment divided by `PCT` for the A treatment.
  - `logrelrisk`: natural logarithm of `relrisk`.
  - `ase.logrelrisk`: asymptotic standard error of `logrelrisk`.
  - `logrelriskCI.lower`, `logrelriskCI.upper`: confidence interval for
• logrelrisk.
• relriskCI.lower, relriskCI.upper: back transform of the CI for the log relative risk into the relative risk scale.

aewide
• Event: adverse event symptom name (factor).
• N.A, N.B: number of patients in treatment groups A and B.
• AE.A, AE.B: number of patients in treatment groups A and B for whom the event Event was observed.
• PCT.A, PCT.B: AE.A/N.A and AE.B/N.B as a percent.
• Relative.Risk: Relative risk defined as PCT.B divided by PCT.A.
• logrelrisk: natural logarithm of relrisk.
• ase.logrelrisk: asymptotic standard error of logrelrisk.
• logrelriskCI.lower, logrelriskCI.upper: confidence interval for logrelrisk.
• relriskCI.lower, relriskCI.upper: back transform of the CI for the log relative risk into the relative risk scale.

A.name, B.name Names of treatment groups (in x$RAND).
col.AB, pch.AB, cex.AB.points color, plotting character and character expansion for the individual points on the left plot.
cex.AB.y.scale Character expansion for the left tick labels (the symptom names).
main.title, main.cex Main title and character expansion for the combined plot in ae.dotplot.
cex The character expansion for the points in the left and right plots.
position.left, position.right position of the left and right plots. This argument is use in S-Plus only, not in R. See the discussion of position in print.trellis.
key.y Position of the key (legend) in the combined plot. This is the y argument of the key. See the discussion of the key argument to xyplot in xyplot.
crit.value Critical value used to compute confidence intervals on the log relative risk. Defaults to 1.96. User is responsible for specifying both crit.value and CI.percent consistently.
CI.percent Confidence percent associated with the crit.value Defaults to 95. User is responsible for specifying both crit.value and CI.percent consistently.
x, y, groups, lwd standard panel function arguments.
lower, upper xr$logrelriskCI.lower and xr$logrelriskCI.upper inside the panel functions.
Details

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

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

Value

logrelrisk takes an input data.frame of the form x described in the argument list and returns a data.frame consisting of the input argument with additional columns as described in the argument xr. The result column of symptom names PREF is an ordered factor, with the order specified by the relative risk.
ae.leftplot returns a "trellis" object containing a horizontal dotplot of the percents against each of the symptom names.
ae.rightplot returns a "trellis" object containing a horizontal plot on the log scale of the relative risk confidence intervals against each of the symptom names.
ae.dotplot calls both ae.leftplot and ae.rightplot and combines their plots into a single display with a single set of left axis labels, a main title, and a key. The value returned invisibly is a list of the full left trellis object and the right trellis object with its left labels blanked out. Printing the value will not usually be interesting as the main title and key are not included. It is better to call ae.dotplot directly, perhaps with a change in some of the positioning arguments.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

AEdotplot for a three-panel version that also has an associated shiny app.

Examples

```r
## variable names in the input data.frame aeanonym
## RAND treatment as randomized
## PREF adverse event symptom name
## SN number of patients in treatment group
## SAE number of patients in each group for whom the event PREF was observed
##
## Input sort order is PREF/RAND
```
## Calculate log relative risk and confidence intervals (95% by default). 
## logrelrisk sets the sort order for PREF to match the relative risk.
```
aeanonymr <- logrelrisk(aeanonym) ## sorts by relative risk
```
```
head(aeanonymr)
```
## construct and print plot on current graphics device
```
ae.dotplot(aeanonymr,
            A.name="TREATMENT A (N=216)",
            B.name="TREATMENT B (N=431)"

## export.eps(h2("stdt/figure/aerelrisk.eps"))
## This looks great on screen and exports badly to eps.
## We recommend drawing this plot directly to the postscript device:
##
```
trellis.device(postscript, color=TRUE, horizontal=TRUE,
                colors=ps.colors.rgb[c("black", "blue", "red", "green",
                "yellow", "cyan", "magenta", "brown"),],
                onefile=FALSE, print.it=FALSE,
                file=h2("stdt/figure/aerelrisk.ps"))
```
```
ae.dotplot(aeanonymr,
            A.name="TREATMENT A (N=216)",
            B.name="TREATMENT B (N=431)"
```
```
## dev.off()
```
## To change the sort order, redefine the PREF factor.
## For this example, to plot alphabetically, use the statement
```
aeanonymr$PREF <- ordered(aeanonymr$PREF, levels=sort(levels(aeanonymr$PREF)))
```
```
ae.dotplot(aeanonymr,
            A.name="TREATMENT A (N=216)",
            B.name="TREATMENT B (N=431)",
            main.title="change the main title to reflect the new sort order")
```
## Not run:
## to restore the order back to the default, use
```
relrisk <- aeanonymr[seq(1, nrow(aeanonymr), 2), "relrisk"]
```
```
PREF <- unique(aeanonymr$PREF)
aeanonymr$PREF <- ordered(aeanonymr$PREF, levels=PREF[order(relrisk)])
ae.dotplot(aeanonymr,
            A.name="TREATMENT A (N=216)",
            B.name="TREATMENT B (N=431)",
            main.title="back to the original sort order")
```
## smaller artificial example with the wide format
```
aewide <- data.frame(Event=letters[1:6],
                      N.A=c(50,50,50,50,20,20),
                      N.B=c(90,90,90,90,90,90),
                      AE.A=2*(1:6),
                      AE.B=1:6)
```
aewtol <- aeReshapeToLong(aewide)
xr <- logrelrisk(aewtol)
AEdotplot

AE (Adverse Events) dotplot of incidence and relative risk

Description

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

Usage

AEdotplot(xr, ...)

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

Arguments

xr For the formula method, a formula of the form AE ~ nAE/nTRT ~ OrgSys, where the condition variable is optional. For the formula method only, the variable names are not restricted. See AEdotplot.data.frame for the support methods.
groups Variable containing the treatment levels.
data data.frame containing at least four variables: containing the AE name as a factor, the treatment level as a factor, the number of observed AE in that treatment level, the number of patients in that treatment group. It may also contain a fifth variable containing a condition variable used to split the data.frame into partitions. It may be used to partition the plot, for example by organ system or by gender. The treatment factor must have exactly two levels. Each AE name must appear exactly once for each level of the treatment.
sortbyRelativeRisk logical. If TRUE, then make the Adverse Events an ordered factor ordering by relative risk. If FALSE, then make the Adverse Events an ordered factor retaining the order of the input levels.
AEdotplot

Sub

... Any of the arguments (such as the sorting options) listed in the calling sequence for the methods documented in AEdotplot.data.frame.

Details

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

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

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

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

Value

The primary interest is in the display of the plot.

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

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

Note

Ann Liu-Ferrara was a beta tester for the shiny app.

Author(s)

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

References


See Also

AEdotplot.data.frame

Examples

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

AEdotplot(AE ~ nAE/nTRT, groups = TRT, data = AEdata) ## sort by Relative Risk
AEdotplot(AE ~ nAE/nTRT | OrgSys, groups = TRT, data = AEdata) ## conditioned on Organ System

## Not run:
AEdotplot(AE ~ nAE/nTRT, groups = TRT, data = AEdata, sortbyVar="PCT") ## PCT A
AEdotplot(AE ~ nAE/nTRT, groups = TRT, data = AEdata, sortbyVar="PCT", sortbyVarBegin=2) ## PCT B
AEdotplot(AE ~ nAE/nTRT, groups = TRT, data = AEdata, sortbyRelativeRisk=FALSE) ## levels(AE)
AEdotplot(AE ~ nAE/nTRT | OrgSys, groups = TRT, data = AEdata, sortbyVar="ase.logrelrisk")

## End(Not run)

## Not run:
AEdotplot(AE ~ nAE/nTRT | OrgSys, groups = TRT, data = AEdata[c(AEdata$OrgSys %in% c("GI","Resp"),)])

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

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

## End(Not run)

## Not run:
## run the shiny app
if (interactive()) shiny::runApp(system.file("shiny/AEdotplot", package="HH"))

## End(Not run)


### Description

Support functions for the `AEdotplot`.

### Usage

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

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

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

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

```r
c(..., panel.widths=attr(aedp[[1]], "panel.widths"),
   par.strip.text=list(cex=.7))
```

`AElogrelrisk(ae,
   name=levels(ae$RAND)[1],
   B.name=levels(ae$RAND)[2],
   crit.value=1.96,
   sortbyRelativeRisk=TRUE, ...,
   sortbyVar=c("PREF", ## Event name
     "PCT",       ## Percent
     "SN",        ## Number of Patients
     "SAE",       ## Number of Observed Events
     "relrisk",   ## Relative Risk (RR)
     "ase.logrelrisk", ## Asymptotic Standard Error(log(RR))
     "relriskCI.lower", ## Confidence Interval Bounds
     "relriskCI.upper"),
   sortbyVarBegin=1) ## 1 for A treatment, 2 for B treatment
```

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

## S3 method for class 'AEdotplot'

```r
update(object, ...)
```

### Arguments

- **ae**
  - For `AElogrelrisk`, a `data.frame` containing at least the first 4 columns of `xr`.
  - For the formula method documented in `AEdotplot`, a formula of the form `AE ~ nAE/nTRT | OrgSys`, where the condition variable is optional. For the formula method only, the variable names are not restricted.

- **xr**
  - For the other methods, `xr` is a `data.frame` containing the Adverse Event data in long format. It must have variables named
    - **RAND**: treatment as randomized (factor with exactly two levels).
    - **PREF**: adverse event symptom name (factor).
    - **SN**: number of patients in treatment group.
    - **SAE**: number of patients in each group for whom the event `PREF` was observed.
    - If the `xr` object is a `AElogrelrisk` object, then it must also have variables
      - **PCT**: number of patients in each group for whom the event `PREF` was observed.
      - **relrisk**: Relative risk defined as `PCT` for the B treatment divided by `PCT` for the
A treatment.

logrelrisk: natural logarithm of relrisk.
ase.logrelrisk: asymptotic standard error of logrelrisk.
logrelriskCI.lower, logrelriskCI.upper: confidence interval for logrelrisk.
relriskCI.lower, relriskCI.upper: back transform of the CI for the log relative risk into the relative risk scale.

sortbyRelativeRisk

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

conditionVariable

Vector of same length as number of rows in xr, it may be one of the columns in xr in which case its full name in the form xr$varname must be used. It will be used to split the data.frame into partitions. It may be used to partition the plot, for example by organ system or by gender.

conditionName

Character. Name to be used in left.strip.

useCondition

logical. If FALSE, then a non-NULL ConditionVariable won't be used.

x

object to be printed.

panel.widths

Vector of three non-negative numerics that sum to 1. These are the widths of each of the three panels in the output plot. The left panel contains the AE names as y-tick labels and the Percent plot. The middle panel contains the Relative Risk plot. The right panel contains a table of the numerical values of number of patients for each treatment, number of adverse events for each treatment, and relative risk. Setting the third value to 0 suppresses the table of numerical values from the display.

AEtable

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

main, sub

Main title and subtitle for the combined plot in AEdotplot.

main.second

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

A.name, B.name

Names of treatment groups (in x$RAND).

col.AB, pch.AB, cex.AB.points

color, plotting character and character expansion for the individual points on the left plot.

cex.AB.y.scale

Character expansion for the left tick labels (the Adverse Effects names).

cex.x.scale

Character expansion for the x-axis tick labels.

key.y

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

ae.key

is a key as described in xyplot.

AEstandard, AEsecond, AEsecond.AEtable, levels.order

Arguments that force the Adverse Events in the panels of AEsecond to have the same sort order levels.order of PREFERENCES as the panels of AEstandard. AEstandard and AEsecond are two "AEdotplot" objects with the same set of panels and the same Adverse Events in corresponding panels. AEsecond.AEtable is the AEtable object from AEsecond. levels.order is the new order for AEsecond; normally the same order as in AEpri.
crit.value  Critical value used to compute confidence intervals on the log relative risk. De-
faults to 1.96. User is responsible for specifying both crit.value and CI.percent consistently.
CI.percent  Confidence percent associated with the crit.value Defaults to 95. User is
responsible for specifying both crit.value and CI.percent consistently.
...  For AEdotplot and AEdotplot.data.frame, all the arguments listed in the
calling sequence for AEdotplot.AErelrisk.. For c.AEdotplot, one or more
“AEdotplot” objects. For print.AEdotplot, the ... arguments are ignored.
sortbyVar  Specify which variable will be used to provide the sort order in the plot. The
names are the internal names for the variables.
sortbyVarBegin 1 for A treatment, 2 for B treatment.
object  An AEdotplot object. The update method updates the components of each
of the constituent trellis objects. It does not update the "main" and "sub"
attributes (nor any other attribute) of the AEdotplot object.
par.strip.text  Default value for strip labels. See xyplot for details.

Details
The first panel is an ordinary dotplot of the percent of AE observed for each treatment by AE.
The second panel shows relative risk of an event on the Treatment B arm (usually the active com-
 pound) relative to the Treatment A arm (usually the placebo), with 95% confidence intervals for a
2 × 2 table. Confidence intervals on the log relative risk are calculated using the asymptotic standard
error formula given as Equation 3.18 in Agresti A., Categorical Data Analysis. Wiley: New York,
1990.

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

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

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

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

Richard M. Heiberger <rmh@temple.edu>

References


See Also

AEdotplot

Examples

```r
## Not run:
## variable names in the input data.frame aeanonym
## RAND treatment as randomized
## PREF adverse event symptom name
## SN number of patients in treatment group
## SAE number of patients in each group for whom the event PREF was observed
## OrgSys Organ System
##
## Input sort order is PREF/RAND

data(aeanonym)
head(aeanonym)

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

data(aeanonym)
head(aeanonym)

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

AEdotplot(aeanonym)

AEdotplot(aeanonym, sort=FALSE)

AEdotplot(aeanonym, conditionVariable=aeanonym$OrgSys)

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

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

Compute and plot oneway analysis of covariance

Description

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

Usage

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

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

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

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

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

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

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

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

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

### Arguments

- **formula**: A formula specifying the model.
- **data.in**: A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.
- **...**: Arguments to be passed to aov, such as subset or na.action.
- **x**: Covariate in ancova, needed for plotting when the formula does not include x. "aov" object in print.ancova, to match the argument of the print generic function. Variable to plotted in "panel.ancova".
- **groups**: Factor. Needed for plotting when the formula does not include groups after the conditioning bar "|".
- **transpose**: S-Plus: The axes in each panel of the plot are transposed. The analysis is identical, just the axes displaying it have been interchanged. R: no effect.
- **display.plot.command**: The default setting is usually what the user wants. The alternate value TRUE prints on the console the command that draws the graph. This is strictly for debugging the ancova command.
- **superpose.level.name**: Name used in strip label for superposed panel.
- **ignore.groups**: When TRUE, an additional panel showing all groups together with a common regression line is displayed.
- **ignore.groups.name**: Name used in strip label for ignore.groups panel.
**ancova**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pch</code></td>
<td>Plotting character for groups.</td>
</tr>
<tr>
<td><code>blocks</code></td>
<td>Additional factor used to label points in the panels.</td>
</tr>
<tr>
<td><code>blocks.pch</code></td>
<td>Alternate set of labels used when a blocks factor is specified.</td>
</tr>
<tr>
<td><code>blocks.cex</code></td>
<td>Alternate set of cex used when a blocks factor is specified.</td>
</tr>
<tr>
<td><code>layout</code></td>
<td>The layout of multiple panels. The default is a single row. See details.</td>
</tr>
<tr>
<td><code>between</code></td>
<td>Space between the panels for the individual group levels and the superpose panel including all groups.</td>
</tr>
<tr>
<td><code>main</code></td>
<td>Character with a main header title to be done on the top of each page.</td>
</tr>
<tr>
<td><code>y,subscripts</code></td>
<td>In &quot;panel.ancova&quot;, see <code>panel.xyplot</code>.</td>
</tr>
<tr>
<td><code>object</code></td>
<td>An &quot;aov&quot;object. The functions using this argument are methods for the similarly named generic functions.</td>
</tr>
<tr>
<td><code>coef, contrasts, classes</code></td>
<td>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.</td>
</tr>
</tbody>
</table>

**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. See `ancovaplot` for a newer set of functions that keep the graph and the `aov` object separate.
Examples

data(hotdog)

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

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

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

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

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

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

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

ancova-class

Class "ancova" Analysis of Covariance

Description

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

Objects from the Class

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

Extends

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

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

ancova

Description

Analysis of Covariance Plots. Any of the ancova models

- \( y \sim x \times t \)
- \( y \sim t \times x \)
- \( y \sim x + t \)
- \( y \sim t + x \)
- \( y \sim x, \text{groups}=t \)
- \( y \sim t, x=x \)
- \( y \sim x \times t, \text{groups}=b \)
- \( y \sim t \times x, \text{groups}=b \)
- \( y \sim x \times t, \text{groups}=b \)
- \( y \sim t + x, \text{groups}=b \)

Usage

ancovaplot(object, ...)

## S3 method for class 'formula'
ancovaplot(object, data, groups=NULL, x=NULL, ..., 
  formula=object, 
  col=rep(tpg$col, 
    length=length(levels(as.factor(groups)))))
  pch=rep(c(15,19,17,18,16,20, 0:14), 
    length=length(levels(as.factor(groups))))),
slope, intercept,
layout=c(length(levels(cc)), 1),
col.line=col, lty=1,
superpose.panel=TRUE,
between=if (superpose.panel)
  list(x=c(rep(0, length(levels(cc))-1), 1))
else
  list(x=0),
col.by.groups=FALSE ## ignored unless groups= is specified
)

panel.ancova.superpose(x, y, subscripts, groups,
slope, intercept,
col, pch, ..., col.line, lty,
superpose.panel,
col.by.groups,
condition.factor,
groups.cc.incompatible,
plot.resids=FALSE,
print.resids=FALSE,
mean.x.line=FALSE,
col.mean.x.line="gray80")

Arguments

formula, object
  formula specifying the aov model. The function modifies it for the xyplot
  specification.

data
data.frame

groups
  If the treatment factor is included in the formula, then groups is not needed.
  By default groups will be set to the treatment factor, but the user may specify
  another factor for groups, usually a blocking factor. The pch will follow the
  value of groups. If the treatment is not included in the formula, then groups is
  required.

x
  Covariate. Required by ancovaplot.formula if the covariate is not included in
  the formula.

For panel.ancova.superpose, see panel.superpose.

...  
  Other arguments to be passed to xyplot.

col, pch
  Standard lattice arguments. pch follows the value of groups. When col.by.groups
  is TRUE, then col follow the value of groups. When col.by.groups is FALSE,
  then col follows the value of the treatment factor, and is constant in each panel.

slope, intercept
  Vector, the length of the number of treatment levels, containing slope and in-
  tercept of the abline in each panel. This is by default calculated based on the
  formula. The user may override each independently.
layout, between

Standard lattice arguments.

col.line, lty

Standard lattice arguments. By default, they follow the value of the treatment factor in the formula. col.line is recycled to the number of panels in the plot.

y, subscripts

See panel.xyplot.

superpose.panel

logical. If TRUE (the default), there is an additional panel on the right containing the superposition of the points and lines for all treatment levels.

col.by.groups

logical. See the discussion in argument col.

condition.factor, groups.cc.incompatible

These are both internal variables. condition.factor contains a copy of the treatment factor. groups.cc.incompatible is a logical which is set to TRUE when the groups argument is explicitly set by the user.

plot.resids, print.resids, mean.x.line, col.mean.x.line

logical, logical, logical or numeric, color name. When plot.resids==TRUE then vertical line segments connecting the data points and the fitted line are drawn. The other two arguments are interpreted only when plot.resids==TRUE. When print.resids==TRUE then the values of the residuals are printed on the console. When is.numeric(mean.x.line) then a vertical reference line is drawn at the specified value, which will normally be specified by the user as the mean of the full set of x values. The reference line will have color specified by col.mean.x.line.

Details

ancova=aov specification  xyplot specification  abline

y ~ x * t  y ~ x | t, groups=t  lm(y[t] ~ x[t])  ## separate lines
y ~ t * x  y ~ x | t, groups=t  lm(y[t] ~ x[t])  ## separate lines
y ~ x + t  y ~ x | t, groups=t  lm(y ~ x + t)  ## parallel lines
y ~ t + x  y ~ x | t, groups=t  lm(y ~ x + t)  ## parallel lines
y ~ x, groups=t  y ~ x | t, groups=t  lm(y ~ x)  ## single regression line
y ~ t, x=x  y ~ x | t, groups=t  mean(t)  ## separate horizontal lines
y ~ x * t, groups=b  y ~ x | t, groups=b  lm(y[t] ~ x[t])  ## sep lines, pch&col follow b
y ~ t * x, groups=b  y ~ x | t, groups=b  lm(y[t] ~ x[t])  ## sep lines, pch&col follow b
y ~ x + t, groups=b  y ~ x | t, groups=b  lm(y ~ x + t)  ## par lines, pch&col follow b
y ~ t + x, groups=b  y ~ x | t, groups=b  lm(y ~ x + t)  ## par lines, pch&col follow b

Value

ancovaplot returns a c("ancova","trellis") object. panel.ancova.superpose is an ordinary lattice panel function.

Author(s)

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


See Also

See the older ancova.

Examples

```r
data(hotdog, package="HH")
ancovaplot(Sodium ~ Calories + Type, data=hotdog)
ancovaplot(Sodium ~ Calories * Type, data=hotdog)
ancovaplot(Sodium ~ Calories, groups=Type, data=hotdog)
ancovaplot(Sodium ~ Type, x=Calories, data=hotdog)

## Please see demo("ancova", package="HH") to coordinate placement
## of all four of these plots on the same page.

ancovaplot(Sodium ~ Calories + Type, data=hotdog, plot.resids=TRUE)
```

## aov.ancovaplot

ANOVA table for a c("ancovaplot","trellis") object.

### Description

ANOVA table for a c("ancovaplot","trellis") object.

### Usage

```r
## S3 method for class 'ancovaplot'
anova(object, ...)
aov.ancovaplot(object, warn=TRUE)
aovStatement(object, ...)
## S3 method for class 'ancovaplot'
aovStatement(object, ...)
aovStatementAndAnova(object, ...)
## S3 method for class 'ancovaplot'
aovStatementAndAnova(object, ...)
## S3 method for class 'ancovaplot'
model.tables(x, ...)
```
Arguments

object, x c("ancovaplot","trellis") object.
warn, ... warn is logical with default TRUE. See the Details section for the interpretation of warn. When ... is received by aov.ancovaplot, it is evaluated if it is warn and ignored for all other values. When ... is received by model.tables it is interpreted normally.

Details

The aov.ancovaplot modifies the call item into an aov call with the same formula and data. If there are groups in the call specified as a name, the groups factor is included in the constructed aov call only if there are both a factor and a covariate in the right-hand-side of the formula. In that case they the groups will be interpreted as a block factor and will be placed first. If the groups are specified as a vector of values in the call, the groups are ignored with a warning. If there is only one term in the right-hand-side, then the groups factor will not be placed into the aov formula. In this case, there will be a warning if the argument warn is TRUE, and no warning if the warn argument is FALSE.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

ancovaplot

anovaMean

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

Description

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

Usage

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

Arguments

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

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

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

See Also
anova.lm, plot.mmc.multicomp

Examples

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

data(pulmonary)

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

---

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

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

Usage

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

- `formula`, `data`, `projections`, `qr`, `contrasts`, ...
  
  See `aov`.

- `weights`
  
  See `lm`.

- `sd` vector of within-group standard deviations.

- `object` "aov" object constructed by `aovSufficient`. It also works with regular `aov` objects.

Value

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

Note

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

- `MMC` and `aov`

Examples

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

if.R(s={}

r={

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

  ## Not run:
  pulmonary.mmc <- mmc(pulmonary.aov,
```
linfct=mcp(smoker="Tukey"),
  df=pulmonary.aov$df.residual,
  vcov.=vcovSufficient)
mmcplot(pulmonary.mmc, style="both")

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

dimnames(pulm.lmat)[[1]] <- row.names(pulmonary)
pulm.lmat

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

mmcplot(pulmonary.mmc, style="both", type="lmat")

## End(Not run)

---

**arima.diag.hh**

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

**Description**

Repair design error in S-Plus arima.diag.

**Usage**

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

**Arguments**

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

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

This function is a no-op in R.

Value

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

tsdiaigplot in both systems and arima.diag in S-Plus.

Description

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

Usage

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

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

rearrange.diag.arma.loop(z)
Arguments

- **x**  
  Time series vector. In S-Plus, x must be an "rts".

- **model**  
  A valid S-Plus model for
  `arima.mle`.

- **order, seasonal**  
  A valid R order and seasonal for
  `arima`.

- **series**  
  Character string describing the time series.

- **...**  
  Additional arguments for `arima.mle` or `arima`.

- **z**  
  For `diag.arma.loop`, an "arma.loop" object. For `rearrange.diag.arma.loop`, an "diag.arma.loop" object.

- **lag.max**  
  Maximum lag for the acf and pacf plots.

- **gof.lag**  
  Maximum lag for the gof plots.

Details

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

Value

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

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

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

Author(s)

Richard M. Heiberger (rmh@temple.edu)

References


See Also

- `tsdiagplot`

Examples

```r
## see tsdiagplot for the example
```
Support functions for diverging stacked barcharts for Likert, semantic differential, and rating scale data.

Description

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

Usage

is.likert(x)

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

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

is.likertCapable(x, ...)

Arguments

x For the as.likert methods, a numeric object stored as a vector, matrix, two-dimensional table, two-dimensional ftable, two-dimensional structable (as de-
defined in the vcd package), or list of named matrices. For functions \texttt{is.likert} and \texttt{is.likertCapable}, any object. This is the only required argument.

rowlabel, collabel

\texttt{names(dimnames(x))}, where \texttt{x} is the argument to the \texttt{as.likert} functions. These will become the \texttt{xlab} and \texttt{ylab} of the likert plot.

... other arguments. They will be ignored by the \texttt{as.likert} method.

\textbf{ReferenceZero} Please see discussion of this argument in \texttt{likert}.

\textbf{xlimEqualLeftRight}

Logical. The default is \texttt{FALSE}. If \texttt{TRUE}, then the left and right x limits are set to negative and positive of the larger of the absolute value of the original x limits.

\textbf{xtickLabelsPositive}

Logical. The default is \texttt{TRUE}. If \texttt{TRUE}, then the tick labels on the negative side are displayed as positive values.

\textbf{padding, reverse.left}

\texttt{padding} is \texttt{FALSE} for \texttt{likert} and \texttt{TRUE} for \texttt{likertMosaic}. \texttt{reverse.left} is \texttt{TRUE} for \texttt{likert} and \texttt{FALSE} for \texttt{likertMosaic}. \texttt{likert} is based on \texttt{barchart} and requires that the sequencing of negative values be reversed. \texttt{likertMosaic} is based on \texttt{mosaic} and needs padding on left and right to fill the rectangle implied by the convex hull of the plot.

\section*{Details}

Please see \texttt{likert} for information on the plot for which \texttt{as.likert} prepares the data.

\section*{Value}

For the \texttt{as.likert} methods, a \texttt{likert} object, which is a matrix with additional attributes that are needed to make the \texttt{barchart} method used by the \texttt{plot.likert} methods work with the data. Columns for respondents who disagree have negated values. Any NA values in the argument \texttt{x} are changed to 0. The column of the original data for respondents who neither agree nor disagree is split into two columns, each containing halved values—one positive and one negative. Negative columns come first in the sequence of "No Opinion"(negative)–"Strongly Disagree", followed by "No Opinion"(positive)–"Strongly Agree". There are four attributes: "\texttt{even.col}" indicating whether there were originally an even number of columns, "\texttt{n.levels}" the original number of levels, "\texttt{levels}" the original levels in the original order, "\texttt{positive.order}" The sequence in which to display the rows in order to make the right hand sides progress with high values on top.

\texttt{is.likert} returns a \texttt{TRUE} or \texttt{FALSE} value.

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

\section*{Author(s)}

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


See Also

likert

Examples

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

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

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

as.matrix.listOfNamedMatrices

(Convert a list of numeric matrices to a single matrix)

Description

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

Usage

## S3 method for class 'listOfNamedMatrices'
as.matrix(x, abbreviate = TRUE, minlength = 4, ...)
is.listOfNamedMatrices(x, xName=deparse(substitute(x)))
## S3 method for class 'listOfNamedMatrices'
as.data.frame(x, ...)
as.listOfNamedMatrices(x, xName=deparse(substitute(x)), ...)
## S3 method for class 'listOfNamedMatrices'

as.listOfNamedMatrices(x, xName=deparse(substitute(x)), ...)

## S3 method for class 'array'

as.listOfNamedMatrices(x, xName=deparse(substitute(x)), ...)

## S3 method for class 'MatrixList'

as.listOfNamedMatrices(x, xName=deparse(substitute(x)), ...)

## S3 method for class 'listOfNamedMatrices'

print(x, ...)

as.MatrixList(x)

## S3 method for class 'array'

as.MatrixList(x)

## S3 method for class 'MatrixList'

print(x, ...)

as.likertDataFrame(x, xName=deparse(substitute(x)))

## S3 method for class 'listOfNamedMatrices'

as.likertDataFrame(x, xName=deparse(substitute(x)))

## S3 method for class 'array'

as.likertDataFrame(x, xName=deparse(substitute(x)))

## S3 method for class 'MatrixList'

as.likertDataFrame(x, xName=deparse(substitute(x)))

### Arguments

- **x**
  Named list of numeric matrices. All matrices in the list should have the same number of columns and the same column names. The names of the list items will normally be long; NA, as introduced by the `addNA`, is a valid name. The row names will normally be long. The number of rows and their names will normally differ across the matrices. Each named item in the list may be a vector, matrix, array, data.frame, two-dimensional table, two-dimensional ftable, or two-dimensional structable. For the `as.MatrixList` methods, an array.

- **...**
  Other arguments. Not used.

- **abbreviate**
  Logical. If TRUE, then use the `abbreviate` function on the item names and row names.

- **minlength**
  The minimum length of the abbreviations.

- **xName**
  Name of the argument in its original environment.

### Value

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

The result of `is.listOfNamedMatrices` is logical value.
print.listOfNamedMatrices prints as.matrix.listOfNamedMatrices of its argument and returns the original argument.

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

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert

Examples

data(ProfChal)

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

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

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

## Not run:
## NA as a dimname value
tmp <- structure(c(0, 0, 0, 6293, 18200, 2122,
0, 0, 0, 2462, 7015, 5589,
6908, 5337, 842, 0, 0, 0),
.Dim = c(3L, 2L, 3L),
.Dimnames = list(c("A", "B", "C"),
c("D", "E"),
c("F", "G", NA)))
tmp
as.MatrixList(tmp)

## End(Not run)

## Not run:
sapply(as.MatrixList(tmp3), as.likert, simplify=FALSE) ## odd number of levels.
data(NZScienceTeaching)
likert(Question ~ ., NZScienceTeaching)
likert(Question ~ . | Subtable, data=NZScienceTeaching)
likert(Question ~ . | Subtable, data=NZScienceTeaching,
        layout=c(1,2), scales=list(y=list(relation="free")))

## End(Not run)

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

Description

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

Usage

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

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

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

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

as.multicomp(x, ...)

## S3 method for class 'glht'
as.multicomp(x,       ## glht object
             focus=x$focus,
             ylabel=deparse(terms(x$model)[[2]]),
             means=model.tables(x$model, type="means",
                                 cterm=focus)$tables[[focus]],
             height=rev(1:nrow(x$linfct)),
             lmat=t(x$linfct),
             lmat.rows=lmatRows(x, focus),
             lmat.scale.abs2=TRUE,
             estimate.sign=1,
as.multicomp

order.contrasts=TRUE,
contrasts.none=FALSE,
level=0.95,
calpha=NULL,
method=x$type,
df,
vcov.,
...)

as.glht(x, ...)

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

Arguments

x "glht" object for as.multicomp. A "mmc.multicomp" object for print.mmc.multicomp.
A "multicomp" object for as.glht and print.multicomp.

... other arguments.
focus name of focus factor.
ylabel response variable name on the graph.
means means of the response variable on the focus factor.
lmat, lmat.rows

mmc

lmat.scale.abs2 logical, almost always TRUE. If it is not TRUE, then the contrasts will not be
properly placed on the MMC plot.
estimate.sign numeric. 1: force all contrasts to be positive by reversing negative contrasts.
$-1$: force all contrasts to be negative by reversing positive contrasts. Leave
contrasts as they are constructed by glht.

order.contrasts, height

logical. If TRUE, order contrasts by height (see mmc).

contrasts.none logical. This is an internal detail. The “contrasts” for the group means are not
real contrasts in the sense they don’t compare anything. mmc.glht sets this
argument to TRUE for the none component.

level Confidence level. Defaults to 0.95.
calpha R only. User-specified critical point. See
confint.glht.
df, vcov. R only. Arguments forwarded through glht to
modelparm.

method R only. See type in
confint.glht.

width.cutoff See deparse.
Details

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

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

Value

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

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

Note

The multiple comparisons calculations in R and S-Plus use completely different functions. 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`. 
as.vector.trellis

Convert a two-dimensional trellis object into a one-dimensional trellis object. Change the order of panels in a trellis object.

Description

as.vector.trellis converts a two-dimensional trellis object into a one-dimensional trellis object. reorder.trellis changes the order of the panel.args component in a trellis object. These are mostly used as utilities by matrix.trellis.

Usage

## S3 method for class 'trellis'
as.vector(x, mode = "any")

## S3 method for class 'trellis'
reorder(x, X, ...)

Arguments

x trellis object.
mode We are hijacking the mode argument. It is used here for the names of the panels.
... Other arguments are ignored.
X Subscript vector specifying the new order of the panels.

Value

trellis object with length(dim(x)) == 1. as.vector retains the original order of the panels. reorder changes the order to the one specified by using the X argument as a subscript.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

tmp <- data.frame(a=letters[c(1:3,1:3,1:3)],
  b=1:9,
  d=1:9,
  e=LETTERS[c(4,4,4,5,5,5,6,6,6)])
tmp
a6 <- xyplot(b ~ d | a*e, data=tmp, pch=19)
a6
dim(a6)
a62 <- as.vector(a6)
a62
dim(a62)
a63 <- reorder(a6, c(1,4,7, 2,5,8, 3,6,9))
a63
dim(a63)
a64 <- matrix.trellis(a63, nrow=3, ncol=3, dimnames=dimnames(a6), byrow=TRUE)
a64
dim(a64)

---

**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`

---

**bivariateNormal**

*Plot the bivariate normal density using wireframe for specified rho.*

**Description**

Plot the bivariate normal density using wireframe for specified rho. There is a shiny app that allows this to be done dynamically.

**Usage**

`bivariateNormal(rho = 0, layout = c(3, 3), lwd = 0.2, angle = c(22.5, 67.5, 112.5, 337.5, 157.5, 292.5, 247.5, 202.5), col.regions = trellis.par.get("regions")$col, ...)"
ci.plot 49

Arguments

rho  Correlation between $x$ and $y$.
layout, lwd  Standard lattice arguments.
angle  This is used as the $z$ component of the screen argument to panel.wireframe.
col.regions, ...  See wireframe.

Details

The default setting shows the view as seen from a series of eight angles. To see just a single view, see the example.

Value

"trellis" object.

Note


Author(s)

Richard M. Heiberger (rmh@temple.edu)

Examples

bv8 <- bivariateNormal(.7) ## all views on one page
bv8
update(bv8[3], layout=c(1,1)) ## one panel
## Not run:
if (interactive())
  shiny::runApp(file.path(system.file(package="HH"), "shiny/bivariateNormal")) ## 3D
if (interactive())
  shiny::runApp(system.file("shiny/bivariateNormalScatterplot", package="HH")) ## scatterplot
## End(Not run)


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

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

```r
## S3 method for class 'lm'
ci.plot(lm.object,
    xlim=range(data[, x.name]),
    newdata,
    conf.level=.95,
    data=model.frame(lm.object),
    newfit,
    ylim,
    pch=19,
    lty=c(1,3,4,2),
    lwd=2,
    main.cex=1,
    main=list(paste(100*conf.level,
        "% confidence and prediction intervals for ",
        substitute(lm.object), sep=""), cex=main.cex), ...
)
```

Arguments

- **lm.object**: Linear model for one y and one x variable.
- **xlim**: xlim for plot. Default is based on data from which lm.object was constructed.
- **newdata**: data.frame containing data for which predictions are wanted. The variable name of the column must be identical to the name of the predictor variable in the model object. Defaults to a data.frame containing a vector spanning the range of observed data. User-specified values are appended to the default vector.
- **conf.level**: Confidence level for intervals, defaults to .95
- **data**: data extracted from the lm.object
- **newfit**: Constructed data.frame containing the predictions, confidence interval, and prediction interval for the newdata.
- **ylim**: ylim for plot. Default is based on the constructed prediction interval.
- **pch**: Plotting character for observed points.
- **lty, lwd**: Line types and line width for fit and intervals.
- **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)

CIplot
Illustration of the meaning of confidence levels.

Description
Illustration of the meaning of confidence levels. Generate sets of confidence intervals for independent
randomly generated sets of normally distributed numbers. Low confidence levels give narrow
intervals that are less likely to bracket the true value. Higher confidence levels increase the prob-
ability of bracketing the true value, and are also much wider and therefore less precise. The shiny
app can animate how the increase in confidence level and width leads to a consequent decrease in
precision.

Usage

CIplot(n.intervals = 100,
       n.per.row = 40,
       pop.mean = 0,
       pop.sd = 1,
       conf.level = 0.95,
       ...
)

confintervaldata(n.intervals = 100,
                 n.per.row = 40,
                 pop.mean = 0,
                 pop.sd = 1,
                 conf.level = 0.95,
                 seed,
                 ...
confinterval.matrix(x,  
   conf.level = attr(x, "conf.level"),  
   ...)  
  
confintervalplot(x.ci,  
   n.intervals = nrow(x.ci),  
   pop.mean = attr(x.ci, "pop.mean"),  
   pop.sd = attr(x.ci, "pop.sd"),  
   n.per.row = attr(x.ci, "n.per.row"),  
   xlim, ylim, ...)  

shiny.CIplot(height = "auto")

**Arguments**

- **n.intervals**: Number of sets of observations to generate. Each set leads to one confidence interval on the plot.
- **n.per.row**: Number of observations in each set.
- **pop.mean, pop.sd**: Population mean and standard deviation for generated set of n.per.row independent normally distributed random numbers.
- **conf.level**: Confidence level of each of the n.per.row confidence intervals calculated from the generated datasets.
- **seed**: Standard argument to `rnorm`.
- **x**: Output matrix from `confintervaldata`.
- **x.ci**: Output data.frame from `confinterval.matrix`.
- **xlim, ylim**: Standard `xyplot` arguments.
- **height**: Height of graph on web page in pixels.
- **...**: Additional arguments. For CIplot, seed will be forwarded to `confintervaldata`, and xlim and ylim will be forwarded to `confintervalplot`. Any other additional arguments will be ignored.

**Details**

The shiny app has sliders for the n.intervals, n.per.row, pop.mean, pop.sd, and conf.level. Changes in the conf.level slider, either manually by animation, use the same set of generated data to show how increasing the confidence level increases the width of the confidence interval and consequently decreases the precision of the interval estimator.

**Value**

CIplot and confintervalplot return a "trellis" plot containing a plot of Confidence Intervals. confintervaldata returns a matrix of n.intervals rows by n.per.row columns of independent normally distributed random numbers. The matrix has a set of attributes recording the arguments to the function.
confinterval.matrix returns a data.frame of n.intervals with three columns containing the lower bound, center, and upper bound of the confidence interval for each row of its input matrix. The data.frame has a set of attributes recording the arguments to the function.

shiny.CIplot returns a shiny app object which, when printed, runs a shiny app displaying the Confidence Interval plot and several slider controls.

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**Examples**

```r
## A. from the console
## example 1
CIplot()

## example 2
## Not run:
CIplot(n.intervals=100,
n.per.row=40,
pop.mean=0,
pop.sd=1,
conf.level=.95)

## End(Not run)

## example 3
## Not run:
tmp.data <- confintervaldata()
tmp.ci <- confinterval.matrix(tmp.data)
confintervalplot(tmp.ci)

## End(Not run)

## example 4
## Not run:
tmp.data <- confintervaldata(n.intervals=100,
n.per.row=40,
pop.mean=0,
pop.sd=1,
conf.level=.95)
tmp.ci <- confinterval.matrix(tmp.data)
confintervalplot(tmp.ci)

## End(Not run)

## B. shiny, initiated from the console
## example 5
```
## Not run:
    if (interactive())
       shiny.CIplot()

## End(Not run)

## example 6
## Not run:
    if (interactive())
       shiny.CIplot(height=800)  ## px
       ## take control of the height of the graph in the web page

## End(Not run)

---

### 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
## Not run:
    trellis.device(theme="col.hh")  ## Open a device with the theme
    trellis.device(theme=col.hh())  ## Open a device with the theme
    trellis.par.set(theme=col.hh())## Add theme to already open device

## End(Not run)
```
Description

col3x2 color dataset.

Usage

data("col3x2")

Format

The format is: chr [1:6] "#1B9E77" "#D95F02" "#7570B3" "#66C2A5" "#FC8D62" "#8DA0CB"

Details

3x2 color scheme. These colors look like a 3x2 color array when run through the vischeck simulator to see how they look for the three most common color vision deficiencies: Deuteranope, Protanope, Tritanope.

References

About 10% of the population have color deficient vision. Your job is make your graphs legible to everyone. Download ImageJ from https://imagej.net/Downloads and VischeckJ from http://vischeck.com and follow the instructions in those sites. This program will allow you to simulate color deficient vision on your computer.

On my Mac, I need to doubleclick ij.jar to open the program. Then open the "Vischeck Panel" on the Plugins menu and navigate to a png file with the "File Open" menu. Click on each of the three types of color deficiency.

Examples

data(col3x2)
col3x2

```
## Not run:
library(RColorBrewer)
library(lattice)
col3x2 <- c(brewer.pal(n=3, "Dark2"), brewer.pal(n=3, "Set2"))
col3x2

## save(col3x2, file="col3x2.rda") ## data(col3x2, package="HH")

## End(Not run)

## Not run:
barchart(~ 1:6, col=col3x2, lwd=0, origin=0, horizontal=FALSE, scales=list(x=list(at=1:6, labels=col3x2)))
```
Combine limits on a one-dimensional trellis object.

Combine limits on a one-dimensional trellis object.

combineLimits.trellisvector(x, margin.x = 1:2, margin.y = 1:2,
layout = x$layout,
ncol=x$layout[1], nrow=x$layout[2],
condlevels = x$condlevels[[1]], ...)  

Arguments

x trellis object.
margin.x, margin.y See combineLimits.
layout See xyplot.
condlevels Character. Names of each panel of the result. Defaults to the names of the panels of the argument.
... Other arguments are ignored.
nrow, ncol See matrix.trellis. These arguments default to the levels of x$layout if it is non-null. Otherwise nrow==1 and ncol==dim(x).

Details

The one-dimensional object is converted to a two-dimensional object which is forwarded to the standard combineLimits function. The result is converted back to a one-dimensional object.
Value

One-dimensional trellis object with combined xlim and ylim values across all panels.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

combineLimits

Examples

tmp <- data.frame(a=1:3,
                  b=c(4,5,7),
                  c=5:7,
                  d=c(8, 9, 12),
                  e=9:11)
tmp

a2 <- xyplot(a + b ~ c + d + e, data=tmp, outer=TRUE,
             scales=list(relation="free"), main="a2")
a2
dim(a2)
combineLimits.trellisvector(a2)
a21 <- combineLimits.trellisvector(update(a2, layout=c(3,2)))
a21
dim(a21)

---

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

Description

Rearranges and improves the legibility of the output from the stepwise function in S-Plus. The output can be used for the Cp plot. cp.calc works only in S-Plus. Use regsubsets in R. The example below works in both languages.

Usage

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

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

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

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

Value

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

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


Examples

## This example is from Section 9.15 of Heiberger and Holland (2004).
data(usair)
if.R(s=(usair <- usair), r={})
splom(~usair, main="U.S. Air Pollution Data with SO2 response", cex=.5)
## export.eps(hh("regb/figure/regb.f1.usair.eps"))
usair$lnSO2 <- log(usair$SO2)
usair$lnmfg <- log(usair$mfgfirms)
usair$lnpopn <- log(usair$popn)
usair[1:3,]  ## lnSO2 is in position 8, SO2 is in position 1  
## lnmfg is in position 9, lnpopn is in position 10

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

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

    old.par <- par(mar=par()$mar+c(0,1,0,0))
    tmp <- (usair.cp$cp <= 10)
    plot(cp ~ p, data=usair.cp[tmp,], ylim=c(0,10), type="n", cex=1.3)
    abline(b=1)
    text(x=usair.cp$p[tmp], y=usair.cp$cp[tmp],
        row.names(usair.cp)[tmp], cex=1.3)
    title(main="Cp plot for usair.dat, Cp<10")
    par(old.par)
    ## export.eps(hh("regb/figure/regb.f3.usair.eps"))
},r=
    usair.regsubsets <- leaps::regsubsets(lnSO2~lnmfg+lnpopn+precip+raindays+temp+wind,
        data=usair, nbest=2)
    usair.subsets.Summary <- summaryHH(usair.regsubsets)
    tmp <- (usair.subsets.Summary$cp <= 10)
    usair.subsets.Summary[tmp,]
    plot(usair.subsets.Summary[tmp,], statistic='cp', legend=FALSE)

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

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

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

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

Generate a sequence spanning the xlim of a lattice window.

Description

Generate a sequence of length points spanning the current.panel.limits()$xlim of a lattice window.

Usage

cplx(length)

Arguments

length  Integer number of points.

Value

One-column matrix containing length rows. The first value is the x-value at the left side of the window. The last value is the x-value at the right side of the window. The in between points are evenly spaced.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

scale

datasets

Datasets for Statistical Analysis and Data Display, Heiberger and Holland
**dchisq.intermediate**

**Description**

Most of the datasets are described in the book *Statistical Analysis and Data Display*. For ProfChal, see `plot.likert`.

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

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


`PoorChildren` is from “Poor Children, Working Parents”, Analysis of data from the Census Bureau’s American Community Survey. Comparison of Census areas of 100,000 or more people, based on samples from 2005 to 2009. Source: Data from the U.S. Census Bureau’s American Community Survey; analysis by Andrew A. Beveridge, Queens College. Copyright 2011 The New York Times Company


**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**References**


**dchisq.intermediate**

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

**Description**

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

Usage

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

Arguments

x,p,q, df,df1,df2, ncp, log,log.p, lower.tail

See pchisq and pf. Some arguments don’t exist in S-Plus. That is why these functions are needed.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

defunct

Defunct Functions in Package ‘HH’

Description

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

Usage

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

... other arguments.

Details

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

## Description

Returns a value for the diagonal of NA and NULL arguments.

## Usage

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

## Arguments

x  
matrix, vector, NA,

...  
Other arguments to diag.

## Author(s)

Richard M. Heiberger (rmh@temple.edu)

## See Also

diag.

## Examples

diag.maybe.null(NULL)
diag.maybe.null(NA)
diag.maybe.null(1:5)
diagplot5new

Description

Transpose of ECDF (Empirical CDF) for centered fitted values and residuals from a linear model.

Usage

diagplot5new(linearmodel, ..., pch = 19)

Arguments

- linearmodel "lm" object.
- pch, ... Arguments to `xyplot`.

Details

This is an implementation in `xyplot` of the "r-f spread" plot.

Value

"trellis" object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


Examples

```r
## See ?residVSfitted
## Not run:
data(fat)
fat.lm <- lm(bodyfat ~ abdomin, data=fat)
diagplot5new(fat.lm)
## End(Not run)
```
diagQQ

QQ plot of regression residuals.

Description

QQ plot of regression residuals. The panel.qqmathline is displayed.

Usage

diagQQ(lm.object, ...)

Arguments

lm.object "lm" object.
...

Additional arguments to qqmath.

Value

"trellis" object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

qqmath

Examples

## See ?residVSfitted
## Not run:
data(fat)
fat.lm <- lm(bodyfat ~ abdomin, data=fat)
diagQQ(fat.lm)

## End(Not run)
Discrete4

Discrete with four levels color dataset.

Description

Discrete with four levels color dataset. These colors look like four distinct colors when run through the vischeck simulator to see how they look for the three most common color vision deficiencies: Deuteranope, Protanope, Tritanope.

Usage

data("Discrete4")

Format

The format is: chr [1:4] "#E31A1C" "#1F78B4" "#FB9A99" "#A6CEE3"

Details

4x1 color scheme

Examples

data(Discrete4)
## Not run:
library(RColorBrewer)
library(lattice)
Discrete4 <- brewer.pal(n=12, "Paired")[c(6,2,5,1)]
Discrete4
## save(Discrete4, file="Discrete4.rda") ## data(Discrete4, package="HH")
##
## barchart(~ 1:4, col=Discrete4, lwd=0, origin=0, horizontal=FALSE,
## xlab="Colors", scales=list(x=list(labels=Discrete4), y=list(labels=NA)),
## main=paste("These colors look like four distinct colors when run through",
## "the vischeck simulator to see how they look for the three most",
## "common color vision deficiencies: Deuteranope, Protanope, Tritanope.",
## sep="\n")

## End(Not run)
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)
EmphasizeVerticalPanels

Helper function for likertWeighted(). used for vertical spacing and horizontal borders of grouped panels.

Description

Helper function for likertWeighted() used for vertical spacing and horizontal borders of grouped panels. Horizontal rules between panels are suppressed by default by likertWeighted unless y.between is non-zero. See examples.

Usage

EmphasizeVerticalPanels(x, y.between)

Arguments

x "trellis" object, normally one constructed by likertWeighted.
y.between The between=list(y=numericvector) argument applied to a trellis object.

Value

Revised trellis object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likertWeighted

Examples

tmp1 <- array(1:60, c(5, 4, 3), list(letters[1:5],letters[6:9],letters[10:12]))
tmp2 <- toCQxR(tmp1)
colnames(tmp2)
likertWeighted(~ . | group + row, tmp2)
likertWeighted(~ . | group + row, tmp2, h.resize.panels=1:5,
between=list(y=c(0,0,3,0)),
h.resizePanels=1:5,
ylab=c("Bottom","Top"),
xlab.top=c("First","Second","Third"),
auto.key.title="Response Level",
main="Three Questions by Five Levels of Classification")

likertObject <- likertWeighted(~ . | group + row, tmp2)
emptyMainLeftAxisLeftStripBottomLegend

EmphasizeVerticalPanels(likertObject, y.between=c(0,0,1,0))

emptyMainLeftAxisLeftStripBottomLegend

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

Description

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

Usage

desc = emptyMainLeftAxisLeftStripBottomLegend(x)

Arguments

x

A "trellis" object.

Details

We manipulate the items inside the trellis object.

Value

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

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

See Also

plot.likert
Examples

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

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

Empl <- ProfChal.list[[2]]

pct <- likert(Empl, as.percent="noRightAxis", xlab="Percent")

count <- likert(Empl, rightAxis=TRUE,
               xlab="Count", ylab.right="Row Count Totals",
               scales=list(x=list(at=c(0, 100, 200))))

countEmptied <- HH:::emptyMainLeftAxisLeftStripBottomLegend(count)

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

export.eps

Exports a graph to an EPS file.

Description

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

Usage

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

Arguments

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

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

See Also
dev2.

Examples

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

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

Usage

as.rts(x, ...)
title.trellis(main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
  line = NA, outer = FALSE, axes=NULL, ...)
title.grob(main=NULL, y=.99, gp=gpar(cex=1.5))
## S3 method for class 'arima.model'
as.character(x, ...)
arima.model(x)
coefArimaHH(object, ...)
.arima.info.names.not.ordered (model)
F.curve

plot a chisquare or a F-curve.

Arguments

x vector or time series

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

Value

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

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

arma.loop

Description

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

Usage

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

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

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

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

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

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

 xlim.in, ylim.in
 Initial settings for xlim, ylim. The defaults are calculated for the degrees of
 freedom.

 df, df1, df2, ncp, log.p
 Degrees of freedom, non-centrality parameter, probabilities are given as log(p).
 See pchisq and pf.

 alpha
 Probability of a Type I error. alpha is a vector of one or two values. If one value,
 it is the right alpha. If two values, they are the c(left.alpha, right.alpha).

 critical.values
 Critical values. Initial values correspond to the specified alpha levels. A scalar
 value implies a one-sided test on the right side. A vector of two values implies
 a two-sided test.

 main.in, ylab.in
 Main title, default ylab.

 shade
 Valid values for shade are "right", "left", "inside", "outside", "none". Default is
 "right" for one-sided critical.values and "outside" for two-sided critical values.

 col
 color of the shaded region and the area of the shaded region.

 shaded.area
 Numerical value of the area. This value may be cumulated over two calls to the
 function (one call for left, one call for right). The shaded. area is the return
 value of the function. The calling program is responsible for the cumulation.

 display.obs
 Logical. If TRUE, print the numerical value of the observed value, plot a vertical
 abline at the value, and use it for showing the p-value. If FALSE, don’t print or
 plot the observed value; just use it for showing the p-value.

 f, chisq
 Values used to draw curve. Replace them if more resolution is needed.

 f.obs, chisq.obs
 Observed values of statistic. p-values are calculated for these values.

 axis.name
 Axis name.

 Other arguments which are ignored.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

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

old.omd <- par(omd=c(.05,.88, .05,1))
chisq.setup(df=12)
chisq.curve(df=12, col='blue', alpha=c(.05, .05))
par(old.omd)
```
old.omd <- par(omd=c(.05,.88, .05,1))
F.setup(df1=5, df2=30)
F.curve(df1=5, df2=30, col='blue')
F.observed(3, df1=5, df2=30)
par(old.omd)

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

---

### glhtWithMCP.993

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

#### Description

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

#### Usage

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

#### Arguments

- `model, linfct, ...`
  
  See [glht](#)

#### Details

`mcp2matrix` is taken from from `multcomp_0.993-2.tar.gz/R/mcp.R`. `glhtWithMCP.993` is based on `glht.mcp` in `multcomp_1.0-0/R/glht.R` with the call to `mcp2matrix` replaced by a call to `mcp2matrix.993`.

#### Value

See [glht](#)

#### Author(s)

Richard M. Heiberger <rmh@temple.edu>

#### See Also

- [mmc](#)
gof.calculation

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

Description

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

Usage

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

Arguments

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

details

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

Value

See the gof value described in arima.diag in S-Plus.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

arima.diag in S-Plus.

Examples

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

Description

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

Usage

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

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

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

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

Arguments

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

labels label values if you don’t want the defaults

Value

See link[grid]{grid.xaxis}.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

link[grid]{grid.xaxis}
GSremove

Remove selected GraphSheetPages in the S-Plus Windows GUI GraphSheet.

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

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

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

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

See Also
graphsheet in S-Plus.

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

## End(Not run)

HH.regsubsets

Display tabular results for Best Subsets Regression.

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

`summaryHH(object, ...)`

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

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

**Arguments**

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

**Value**

`summaryHH` produces a table of models, with p, rsq, rss, adjr2, cp, bic, stderr for each. `plot.summaryHH.regsubsets` plots the specified statistic from the summary. All the others are support functions.
R tools for writing HH2: hhpdf, hhdev.off, hhcapture, hhcode, hhpng, hhlatex.

The files in HHscriptnames() contain R code for all examples and figures in the book. The examples can all be directly executed by the user. The code examples all use these functions. The versions of these functions here are essentially placeholders. Functions hhpdf, hhpng, and hhdev.off are no-ops and return NULL. As a consequence, the code between them will execute and display on the default graphics device. Function hhcapture sources its text argument and prints the
output to the console. Function hhcode prints its text argument to the console. Function hhlatex
prints the latex source to the console and returns NULL.
While writing the book, these placeholder functions are replaced by more elaborate functions with
the same names that write the graphs onto pdf or png files, the console output to text files, and the
latex code to a file.

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

HHscriptnames

Find absolute pathname of a script file for the HH book in the HH
package.

Description
Find absolute pathname of a script file for the HH book in the HH
package.

Usage
HHscriptnames(chapternumbers=NULL, edition=2)

WindowsPath(x, display=TRUE)

Arguments
chapternumbers A number or letter name for a chapter in the HH book. For the Second edi-
tion, the valid values are from the set c(1:18, LETTERS[1:15]). For the First edi-
tion, the valid values are from the set c(1:18). The argument may be a vec-
tor of one or more items. The file basename for the corresponding chapter is
also accepted. If the chapternumbers is NULL (the default) then the directory
containing the script files for the edition is returned.
edition Either 2 or 1, for the second or first edition of the book Statistical Analysis and
Data Display.
x A vector or matrix of pathnames as generated by R, with "/" as the separator
character.
display Logical. With the default TRUE, the WindowsPath function prints the pathname
on the console with a single \ character as the separator suitable for copy and
paste into a Windows program and returns its result invisibly. With FALSE the
WindowsPath function does not print anything; it returns its result visibly.

Value
For HHscriptnames, matrix of full pathnames to script files in the HH package.
For WindowsPath, a vector or matrix of full pathnames with all "/" characters changed to "\"
(which displays as \ by the cat function). When display is TRUE the function also prints at the
console the pathnames with a single \ character suitable for copy and paste into a Windows program.
Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


Examples

```r
## Not run:
## All Operating Systems

## Second Edition
HHscriptnames()
HHscriptnames(6)
HHscriptnames("6")
HHscriptnames("oway")

HHscriptnames("H")
HHscriptnames("RApx")

HHscriptnames(c(1:18, LETTERS[1:15]))

## with Windows pathname separators
WindowsPath(HHscriptnames())
WindowsPath(HHscriptnames(6))
WindowsPath(HHscriptnames(6), display=FALSE)
WindowsPath(HHscriptnames(6:8))
WindowsPath(HHscriptnames(6:8), display=FALSE)

## First Edition
HHscriptnames(6, edition=1)

## End(Not run)
```

hov 

**Homogeneity of Variance**

Description

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

hov(x, data=NULL, method = "bf")  ## x is a formula

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

Arguments

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

Value

"htest" object for the hov test.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

Heiberger, Richard M. and Holland, Burt (2015). *Statistical Analysis and Data Display: An Inter-


See Also

aov, hovPlot

Examples

data(turkey)

hov(wt.gain ~ diet, data=turkey)
hovPlot(wt.gain ~ diet, data=turkey)
Homogeneity of Variance: Brown–Forsyth method

**Description**

Homogeneity of Variance: Brown–Forsyth method

**Usage**

```r
hovBF(x, data=NULL, ..., na.rm = TRUE)
hovplotBF(x, data, ..., na.rm = TRUE,
        main = "Brown-Forsyth Homogeneity of Variance", plotmath = TRUE)
```

**Arguments**

- `x`  
  Model formula with one response variable and one factor.
- `data`  
  data.frame
- `...`  
  Other arguments. `hovplotBF` sends them on to the panel function. `hovBF` ignores them.
- `na.rm`  
  A logical value indicating whether 'NA' values should be stripped before the computation proceeds. See `median`.
- `main`  
  main title for the plot.
- `plotmath`  
  Logical. When TRUE (the default) the strip labels use `plotmath`. When FALSE the strip labels use ASCII.

**Value**

`hovplotBF` returns a three-panel `trellis` object. `hovBF` returns an `htest` object.

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**References**


**Examples**

```r
data(batch)
batch1.aov <- aov(Calcium ~ Batch, data=batch)
anova(batch1.aov)
hovBF(Calcium ~ Batch, data=batch)
hovplotBF(Calcium ~ Batch, data=batch)
```
if.R

Conditional Execution for R or S-Plus

Description

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

Usage

if.R(r, s)

Arguments

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

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

Details

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

Value

The result of the executed expression.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

is.R

Examples

if.R(r="This is R.",
     s="This is S-Plus")]
**InsertVerticalPanels**

Expand a 3D array on the second dimension, inserting empty layers where the input vector has a 0 value. A 2D argument `x` with `dim(x)==c(r,c)` is first extended to 3D with `dim(x)==c(1,r,c)`, and then the result is collapsed back to 2D.

**Description**

Expand a 3D array on the second dimension, inserting empty layers where the input vector has a 0 value. A 2D argument `x` with `dim(x)==c(r,c)` is first extended to 3D with `dim(x)==c(1,r,c)`, and then the result is collapsed back to 2D.

**Usage**

`InsertVerticalPanels(x, expansion, newRowheights=5, newValue=NA)`

**Arguments**

- `x` Three-dimensional array, for example, one defined as a set of matrices for the `likert` and related functions. `x[1,]` and more generally `x[i,]` will be an argument to `likert`.
- `expansion` Vector of 0 and 1, with 1 indicating an existing layer in dimension 2, and 0 a placeholder for where a new layer in dimension 2 should be inserted.
- `newRowheights` Value to be used for inserted row by `likertWeighted` function.
- `newValue` Value to be inserted in all positions of inserted layer.

**Value**

Array similarly structured to the input array `x`, but with more layers on the second dimension. The "rowheights" attribute gives the rowheights used by `EmphasizeVerticalPanels`. The `newRows` gives the row (second dimension) numbers in the result that are the generated values. All data items in the `newRows` will have value in the `newValue` argument.

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**See Also**

`EmphasizeVerticalPanels`
interaction.positioned

Examples

```r
x <- array(1:24, c(3, 4, 2),
           dimnames = list(letters[1:3], LETTERS[4:7], letters[8:9]))
x

expansion <- c(1, 1, 0, 1, 0, 1)
result <- InsertVerticalPanels(x, expansion)
result

Pop.labels1 <- result[, ,]
Pop.labels1[ attr(result, "newRows"),] <- " "

Pop.labels2 <- result[, ,]
Pop.labels2[ attr(result, "newRows"),] <- " "

Pct.labels1 <- format(round(HH::rowPcts(result[, ,])))
Pct.labels1[ attr(result, "newRows"),] <- " "

Pct.labels2 <- format(round(HH::rowPcts(result[2, ,])))
Pct.labels2[ attr(result, "newRows"),] <- " ">
```

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.
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**

Plot all main effects and two-way interactions in a multifactor design

Description

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

Usage

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

## S3 method for class 'formula'
```r
interaction2wt(x, data=NULL, responselab, ...)
```

## Default S3 method:
```r
interaction2wt(x, 
  response.var, 
  responselab = deparse(substitute(response.var)), 
  responselab.expression = responselab, 
  relation = list(x = "same", y = "same"), 
  x.relation = relation$x, 
  y.relation = relation$y, 
  digits = 3, 
  x.between=1, 
  y.between=1, 
  between, 
)```
cex = 0.75,
rot=c(0,0),
panel.input = panel.interaction2wt,
strip.input =
  if (label.as.interaction.formula) strip.default
  else strip.interaction2wt,
par.strip.text.input = trellis.par.get()$add.text,
scales.additional,
main.in =
  paste(responcelab,
    ": ", c("main", "simple")[1+simple],
    ": effects and 2-way interactions",
    sep=""),
  xlab = "",
ylab = "",
simple=FALSE,
box.ratio=if (simple) .32 else 1,
label.as.interaction.formula=TRUE,
..., 
main.cex,
key.cex.title=.8,
key.cex.text=.7,
factor.expressions=names.x,
simple.pch=NULL
}

Arguments

Arguments when x is a formula.

The object on which method dispatch is carried out.

For the "formula" method, a formula describing the response variable and factors. The formula is generally of the form \( y \sim g_1 + g_2 + \ldots \). 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.

responcelab

Character name of response variable, defaults to the name of the response variable in the formula.

responcelab.expression

plotmath or character name of response variable, defaults to responcelab.

...

additional arguments, primarily trellis arguments.

response.var

For the "default" method, the response variable. This is usually constructed by formula method from the input data and the input formula.
simple logical. TRUE if simple effects are to be displayed. Arguments `simple.offset`, `simple.scale`, and `col.by.row` may also be needed. See `panel.interaction2wt` for details.

`box.ratio` `xyplot`.
Trellis/Lattice arguments. Default values are set by the `formula` method. The user may override the defaults. See also `xyplot`.

`relation` `trellis` argument.
`x.relation` `x` value of `relation` argument.
`y.relation` `y` value of `relation` argument.

digits doesn’t do anything at the moment

`x.between` `x` value of `between` argument.
`y.between` `y` value of `between` argument.

`between` trellis/lattice between argument. If used, `between` has precedence over both the `x.between` and `y.between` arguments.

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

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

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

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

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

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

`main.in` Text of main title.
`xlab` No effect.
`ylab` No effect.

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

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

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

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

`simple.pch` Named list containing plotting characters for each level of one or more of the factors. `simple.pch` is used only when `simple==TRUE`. If the argument `simple.pch` is missing, then the integers for the levels of the factors are used. The characters are used for the median of the box plots in the diagonal panels. They match the trace factor of the interaction panel in the same column of the display.
interaction2wt

Value

"trellis" object containing the plot.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

panel.interaction2wt

Examples

data(vulcan)
interaction2wt(wear ~ filler + pretreat + raw, data=vulcan,
  par.strip.text=list(cex=.7))
interaction2wt(wear ~ filler + pretreat + raw, data=vulcan)
interaction2wt(wear ~ filler + raw, data=vulcan,
  simple=TRUE)
interaction2wt(wear ~ filler + raw, data=vulcan,
  simple=TRUE, simple.scale=c(filler=.15, raw=.2),
  xlim=c(.3, 5.6))

ToothGrowth$dose <- positioned(ToothGrowth$dose) ## modify local copy
anova(aov(len ~ supp*dose, data=ToothGrowth))
interaction2wt(len ~ supp + dose, data=ToothGrowth)

esoph$ntotal <- with(esoph, ncases + ncontrols) ## modify local copy
esoph$rate <- with(esoph, ncases/ntotal) ## modify local copy

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

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

esoph.aov <- aov(rate ~ agegp + alcgp + tobgp, data=esoph)
anova(esoph.aov)

---

interval Prediction and Confidence Intervals for glm Objects

Description

Prediction and Confidence Intervals for glm Objects

Usage

```r
interval(glm.object, ...)
## S3 method for class 'glm'
interval(glm.object, linkfit.object,
  type = c("link", "response"),
  conf.level = 0.95, ...)
```

Arguments

- **glm.object**: result from a call to the glm function.
- **linkfit.object**: result from a call to the predict function for the glm.object with type="link", se.fit=TRUE.
- **type**: Either "link" or "response". See predict.glm for details.
- **conf.level**: Confidence level, for example .95 for 95%.
- **...**: Other arguments to be passed to predict.glm.

Value

Matrix with five columns: fit, ci.low, ci.hi, pi.low, pi.hi and as many rows as predict.glm returns.

Author(s)

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

```r
data(spacshu)
spacshu.bin.glm <- glm(damage ~ tempF, data=spacshu, family=binomial)

## observed data
spacshu.interval <- interval(spacshu.bin.glm)

## new data, link
spacshu.interval.link <- interval(spacshu.bin.glm, newdata=data.frame(tempF=30:85))

## new data, response
spacshu.interval.response <- interval(spacshu.bin.glm, newdata=data.frame(tempF=30:85),
                                        type="response")
```

### 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**

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

```
panel.intxplot(x, y, subscripts, groups, type = "l", ..., se, cv=1.96,
```
intxplot

```
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.
\texttt{intxplot}

\begin{verbatim}
y, subscripts, groups, type
\end{verbatim}
Standard arguments for panel functions.

\begin{verbatim}
cv
\end{verbatim}
critical value for confidence intervals. Defaults to 1.96.

\begin{verbatim}
offset.use
\end{verbatim}
logical. If \texttt{TRUE}, offset the endpoints of each group.

\begin{verbatim}
offset.scale
\end{verbatim}
Scale number indicating how far apart the ends of the groups will be placed.
Larger numbers make them closer together.

\begin{verbatim}
offset
\end{verbatim}
Actual numbers by which the end of the groups are offset from their nominal
location which is the \texttt{as.numeric} of the group levels.

\begin{verbatim}
rug.use
\end{verbatim}
logical. If \texttt{TRUE}, display a rug for the endpoints of each group.

\textbf{Value}

\begin{verbatim}
"trellis" object.
\end{verbatim}

\textbf{Author(s)}

Richard M. Heiberger <rmh@temple.edu>

\textbf{See Also}

\texttt{sufficient}

\textbf{Examples}

\begin{verbatim}
## This uses the same data as the HH Section 12.13 rhizobium example.
data(rhiz.clover)

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

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

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

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

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

## change distance between endpoints
intxplot(Npg ~ strain, groups=comb, data=rhiz.clover,
se=TRUE, offset.scale=20)
\end{verbatim}
### When data includes the nobs and sd variables, data.is.summary=TRUE is needed.

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

**Usage**

```r
ladder(formula.in, data=NULL, 
  main.in="Ladders of Powers", 
  panel.in=panel.cartesian, 
  xlab=deparse(formula.in[[3]]), 
  ylab=deparse(formula.in[[2]]), 
  scales=list(alternating=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=2, 
  strip.names, 
  strip.style=1, 
  strip, 
  oma=c(0,0,0,0), ## S-Plus 
  axis3.line=.61, 
  layout=c(length(tmp$x.power), length(tmp$y.power)), 
  axis.key.padding = 10, ## R right axis 
  key.axis.padding = 10, ## R top axis 
  useOuter=TRUE, ## R useOuterStrips(combineLimits(result))
  ...) 
```

```r
ladder3(x, y, 
  dsx=deparse(substitute(x)),
```

---

---
dsy=deparse(substitute(y)),
ladder.function=ladder.f)

ladder.f(x, name.prefix="")
ladder.fstar(x, name.prefix="")

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

name.prefix Base name used for column names of powers. The default is empty (""). An alternative must include the power symbol "^", for example, "abc^".

strip.number Number of strip labels in each panel of the display. 0: no strip labels; 1: one strip label of the form y^p ~ x^q; 2: two strip labels of the form ylab: y^p and xlab: x^q, where p and q are the powers returned by ladders; y and x are the arguments dsy and dsx.

useOuter logical, defaults to TRUE. In R, this implies that strip.number is forced to 2 and that the resulting "trellis" object will be sent through useOuterStrips(combineLimits(result)). This argument is ignored by S-Plus.
strip.style  style argument to strip.
oma         argument to par in S-Plus.
...         other arguments to xyplot.
axis3.line  extra space to make the top axis align with the top of the top row of panels. Trial
            and error to choose a good value.
axis.key.padding
            Extra space on right of set of panels in R.
key.axis.padding
            Extra space on top of set of panels in R.
x, y         variables.
which.given, which.panel, var.name, factor.levels, shingle.intervals, par.strip.text
            See strip.default.
strip.names, style
            We always print the strip.names in style=1. Multicolored styles are too busy.

Details

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

Value

ladder returns a "trellis" object.

The functions ladder.fstar and ladder.f take an input vector x of non-negative values and con-
struct 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 transforma-
tion.

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

panel.cartesian

Examples

data(tv)

## default
ladder(life.exp ~ ppl.per.phys, data=tv, main="Ladder of Powers for Life Expectancy and People per Physician")

## Not run:
## one strip label
ladder(life.exp ~ ppl.per.phys, data=tv, strip.number=1, useOuter=FALSE, dsx="ppp", dsy="le")

## two strip labels
ladder(life.exp ~ ppl.per.phys, data=tv, strip.number=2, useOuter=FALSE)

## outer strip labels
ladder(life.exp ~ ppl.per.phys, data=tv, useOuter=TRUE)

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

## End(Not run)

latex.array

Generate the latex code for an "array" or "table" with 3, 4, or more dimensions.

Description

Generate the latex code for an "array" or "table" with 3, 4, or more dimensions.
Usage

## S3 method for class 'array'
latex(object, ..., 
  var.sep = "\tabularnewline{\bfseries ", value.sep = ": ", 
  use.ndn = TRUE, cgroup = NULL, 
  ## rgroup here captures and ignores any incoming rgroup argument 
  rgroup = NULL, n.rgroup = NULL, 
  title = first.word(deparse(substitute(object))), 
  rrowlabel=title, 
  rsubgroup=NULL, n.rsubgroup=NULL)

## S3 method for class 'matrix'
latex(object, ..., 
  use.ndn=TRUE, cgroup=NULL, 
  title=first.word(deparse(substitute(object))), 
  rrowlabel=title)

## S3 method for class 'table'
latex(object, ...) ## prepend c("matrix", "array") to the 
## class of the input object, and then call latex.default

Arguments

object A c(“matrix”, “array”) or “table” object.

... Arguments forwarded to the ”default” method for latex.

use.ndn Logical. ndn is an abbreviation for ”Names of DimNames”. When TRUE (the default), the rowlabel, cgroup, and rgroup values will be taken from the names(dimnames(object)).

rgroup, n.rgroup These are the standard arguments for latex. Incoming values for rgroup and n.rgroup are ignored by latex.array and replaced with values constructed from the names of the dimnames of the third and higher dimensions of the input array object. Each item in rgroup is assigned the appropriate combination of names(dimnames(object))[-(1:2)].

rsubgroup, n.rsubgroup These are based on the standard arguments for latex. Incoming values for rsubgroup and n.rsubgroup are applied to the rows of each rgroup.

title, rowlabel, cgroup 
These are the standard arguments for latex. When use.ndn is TRUE (the default), then rowlabel is assigned the names(dimnames(object))[1] and cgroup is assigned the names(dimnames(object))[2].

value.sep When use.ndn is TRUE (the default), and length(dim(object)) >= 3 then this string is used in the constructed rgroup values to separate the factor name from the factor level of the specified dimension, for example ABC: 5.

var.sep When use.ndn is TRUE (the default), and length(dim(object)) >= 4 then this string is used in the in the constructed rgroup values to separate the name and
level of each dimension, for example ABC: 5 ; DEF: 6. The default value is exactly what \texttt{Hmisc::latex} needs in order to place two or more lines (one for each dimension) in boldface.

Details

latex.matrix calls latex.default directly. When use.ndn is \texttt{TRUE} (the default), rowlabel and cgroup are constructed from names(dimnames(object)) unless the user explicitly specified them.

latex.array appends all two-dimensional layers object[,, one, at, a, time] into a single long "matrix", ignores any incoming rgroup and n.rgroup (with a warning), and constructs rgroup and n.rgroup to label the layers. When use.ndn is \texttt{TRUE} (the default), rowlabel and cgroup are constructed from names(dimnames(object)) unless the user explicitly specified them.

latex.table prepends c("matrix", "array") to the class of the "table" object, then calls the generic "latex". This step is necessary because the survey package creates objects whose class includes the value "table" but not the values c("matrix", "array"). Should this object be sent directly to latex.default, it would cause on error for any table with dimension larger than two.

Value

See latex.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

latex

Examples

```r
## Not run:
## These are the recommended options. See #Hmisc::latex for details.
options(latexcmd="pdflatex")
options(dviExtension='pdf')
options(xdvicmd='open') ## Macintosh, Windows, SMP linux

## End(Not run)

## This sets up the defaults for latex to write to a pdf file
microplot::latexSetOptions()
## It is needed for R CMD check.
## It is recommended if you normally use pdflatex.
## If you want some other destination for latex, use a non-default argument.

tmp3 <- array(1:8, c(2,2,2),
    list(letters[1:2],
        letters[3:4],
        letters[5:6]))
```
```r
## Not run:
try(  ## warning: Input rgroup and n.rgroup are ignored
    latex(tmp3, rgroup=letters[1:3], n.rgroup=c(1,1,2), file="ignorergroup.tex")
)

names(dimnames(tmp3)) <- LETTERS[24:26]
latex(tmp3, file="LETTERS3.tex")
ltmp3 <- latex(tmp3, rowlabel="Something Else", file="SomethingElse.tex")

## with rsubgroup and n.rsubgroup
latex(tmp4, var.sep=" ; ", file="LETTERS4sub.tex",
    rsubgroup=c("Three","Two"), n.rsubgroup=c(3,2))

## with rsubgroup and n.rsubgroup and cgroup and n.cgroup
latex(tmp4, var.sep=" ; ", file="LETTERS4sub.tex",
    rsubgroup=c("Three","Two"), n.rsubgroup=c(3,2),
    cgroup=c("FGH","I"), n.cgroup=c(3,1))

# tmp2 <- array(1:6, c(3,2),
#     list(Rows=letters[1:3],
#          Columns=letters[4:5]))
```

```
## Input rgroup honored for "matrix"
latex(tmp2, rgroup=c("Two","One"), n.rgroup=c(2,1), file="rgroup.tex")
latex(tmp2, rowlabel="something else", file="something.tex")

## tableDemo is based on a table constructed from
## survey::svytable(~ FactorA + FactorB + FactorC, Survey.Design.Object)

```r
tableDemo <- structure(c(28, 25, 33, 12, 6, 22, 8, 12, 23, 24, 6, 32,
    32, 31, 59, 11, 2, 33, 10, 3, 23, 7, 2, 26),
  .Dim = c(3L, 4L, 2L),
  .Dimnames = list(FactorA = c("a", "b", "c"),
                   FactorB = c("d", "e", "f", "g"),
                   FactorC = c("h", "i")),
  class = "table")
```

class(tableDemo)
latex(tableDemo)

## End(Not run)

---

### latticeresids

*Subroutine used by residual.plots.lattice*

**Description**

Subroutine used by residual.plots.lattice

**Usage**

```r
latticeresids(x, data, 
  main = "please use an appropriate main title",
  par.strip.text, scales.cex, y.relation, ...)
```

**Arguments**

- `x`, `data`, `main`, `par.strip.text`, ...
  - `lattice` arguments. See `xyplot`.
- `scales.cex` cex for the scales argument in `xyplot`.
- `y.relation` relation for the y argument to scales argument in `xyplot`.

**Value**

"trellis" object.

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>
See Also

residual.plots.lattice

---

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

Description

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

Usage

legendGrob2wt(...)

Arguments

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

Value

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

Author(s)

Richard M. Heiberger, with assistance from Deepayan Sarkar.

See Also

interaction2wt

---

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

Description

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

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

## S3 method for class 'formula'
plot.likert(x, data, ReferenceZero=NULL, value, levelsName="",  
scales.in=NULL, ## use scales=
 between=list(x=1 + (horizontal), y=.5 + 2*(!horizontal)),
 auto.key.in=NULL, ## use auto.key=
 panel.in=NULL, ## use panel=
 horizontal=TRUE,
 par.settings.in=NULL, ## use par.settings=
 ...,  
 as.percent = FALSE,
 ## titles
 ylab= if (horizontal) {  
 if (length(x)==3)
 deparse(x[[2]])
 else  
 "Question"
 }
 else
 if (as.percent != FALSE) "Percent" else "Count",

 xlab= if (!horizontal) {  
 if (length(x)==3)
 deparse(x[[2]])
 else  
 "Question"
 }
 else
 if (as.percent != FALSE) "Percent" else "Count",

 main = x.sys.call,

 ## right axis
 rightAxisLabels = rowSums(data.list$Nums),  
 rightAxis = !missing(rightAxisLabels),
 ylab.right = if (rightAxis) "Row Count Totals" else NULL,  
 xlab.top = NULL,
 right.text.cex =  
 if (horizontal) { ## lazy evaluation
 if (!is.null(scales$y$cex)) scales$y$cex else .8
 }  
 else
 {
if (!is.null(scales$x$cex)) scales$x$cex else .8

## scales
xscale.components = xscale.components.top.HH,
yscale.components = yscale.components.right.HH,
xlimEqualLeftRight = FALSE,
xTickLabelsPositive = TRUE,

## row sequencing
as.table=TRUE,
positive.order=FALSE,
data.order=FALSE,
reverse=ifelse(horizontal, as.table, FALSE),

## resizePanels arguments
h.resizePanels=sapply(result$y.used.at, length),
w.resizePanels=sapply(result$x.used.at, length),

## color options
reference.line.col="gray65",
col.strip.background="gray97",
key.border.white=TRUE,
col=likertColor(Nums.attr$nlevels,
    ReferenceZero=ReferenceZero,
    colorFunction=colorFunction,
    colorFunctionOption=colorFunctionOption),
colorFunction="diverge_hcl",
colorFunctionOption="lighter"

## S3 method for class 'matrix'
plot.likert(x,
    positive.order=FALSE,
ylab=names(dimnames(x)[1]),
xlab=if (as.percent != FALSE) "Percent" else "Count",
main=xName,
reference.line.col="gray65",
col.strip.background="gray97",
col=likertColor(attr(x, "nlevels"),
    ReferenceZero=ReferenceZero,
    colorFunction=colorFunction,
    colorFunctionOption=colorFunctionOption),
colorFunction="diverge_hcl",
colorFunctionOption="lighter",
as.percent=FALSE,
par.settings.in=NULL,
horizontal=TRUE,
ReferenceZero=NULL,
..., key.border.white=TRUE,
  xName=deparse(substitute(x)),
  rightAxisLabels=rowSums(abs(x)),
  rightAxis=!missing(rightAxisLabels),
  ylab.right=if (rightAxis) "Row Count Totals" else NULL,
  panel=panel.barchart,
  xscale.components=xscale.components.top.HH,
  yscale.components=yscale.components.right.HH,
  xlimEqualLeftRight=FALSE,
  xTickLabelsPositive=TRUE,
  reverse=FALSE)

## Default S3 method:
plot.likert(x, ...) ## calls plot.likert.matrix

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

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

## S3 method for class 'list'
plot.likert(x, ## named list of matrices, 2D tables,
  ## or all-numeric data.frames
  condlevelsName="ListNames",
  xName=deparse(substitute(x)),
  main=paste("List items of", xName, "by", condlevelsName),
  layout=if (length(dim.x) > 1) dim.x else {
    if (horizontal) c(1, length(x)) else c(length(x), 1)),
    positive.order=FALSE,
    strip=!horizontal,
    strip.left=horizontal,
    strip.left.values=names(x),
    strip.values=names(x),
    strip.par=list(cex=1, lines=1),
    strip.left.par=list(cex=1, lines=1),
    horizontal=TRUE,
    ...
  rightAxisLabels=sapply(x, function(x) rowSums(abs(x)), simplify = FALSE),
  rightAxis=!missing(rightAxisLabels),
  resize.height.tuning=-.5,
Arguments

x

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

The . on the right-hand-side is expanded to the formula containing the sum of all remaining (after the response and the conditioning variables) numeric columns in data. An empty left-hand-side is interpreted as the rownames(data). See the examples for all possible forms of formula recognized by the likert function. Otherwise, any numeric object stored as a vector, matrix, array, data.frame, table, ftable, structable (as defined in the vcd package), or as a list of named two-dimensional objects. This is the only required argument. See the Details section for restrictions on the form of data.frame, list, ftable, and structable arguments.

data

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

ReferenceZero

Numeric scalar or NULL. The position in the range seq(0, attr(x, "nlevels")+.5,.5) where the reference line at 0 will be
likert

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

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

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

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

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

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

as.table Standard lattice argument. See barchart.

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

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

ylab, xlab, ylab.right, xlab.top, main

Standard lattice graph labels in barchart.

right.text.cex The right axis, as used here for the "Row Count Totals", has non-standard controls. It's cex follows the cex of the left axis, unless this argument is used to override that value. When horizontal=FALSE, then the top axis defaults to follow the bottom axis unless overridden by right.text.cex.

between Standard lattice argument.

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

colorFunction, colorFunctionOption
See `likertColor`.

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

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

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

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

... other arguments. These will be passed to the `barchart` function by the `plot.likert` method. The most useful of these is the `border` argument which defaults to make the borders of the bars the same color as the bars themselves. A scalar alternative (border="white" being our first choice) puts a border around each bar in the stacked barchart. This works very well when the ReferenceZero line is between two levels. It gives a misleading division of the central bar when the ReferenceZero is in the middle of a level. See the example in the examples section. Arguments to the `lattice auto.key=list()` argument (described in `barchart`) will be used in the legend. See the examples.

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

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

xName
Name of the argument in its original environment.

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

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

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

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

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

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

panel
panel function eventually to be used by barchart.

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

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

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

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

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

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

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

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

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

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

Value

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

Note

The current version of the likert function uses the default diverging palette from diverge_hcl as
the default. Previous versions used the RColorBrewer palette "RdBu" as the default color palette.
The previous color palette is still available with an explicit call to likertColorBrewer, for example
col=likertColorBrewer(nc, ReferenceZero=ReferenceZero,
BrewerPaletteName="RdBu", middle.color="gray90")

Note

Ann Liu-Ferrara was a beta tester for the shiny app.
**Note**

**Documentation note:** Most of the plots drawn by `plot.likert` have a long left-axis tick label. They therefore require a wider window than R’s default of a nominal 7in × 7in window. The comments with the examples suggest aesthetic window sizes.

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

1. `likert(x)` # recommended
   `likert` is an alias for `plot.likert()`.

2. `plot.likert(x)`
   `plot.likert` is both a method of `plot` for "likert" objects, and a generic function in its own right. There are methods of `plot.likert` for "formula", "matrix", "array", "table", and several other classes of input objects.

3. `plot(as.likert(x))`
   Both `likert` and `plot.likert` work by calling the `as.likert` function on their argument `x`. Once `as.likert` has converted its argument to a "likert" object, the method dispatch technology for the generic `plot.likert` is in play. The user can make the explicit call `as.likert(x)` to see what a "likert" object looks like, but is very unlikely to want to look a second time.

**Author(s)**

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

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

**References**

doi:10.18637/jss.v057.i05.


**See Also**

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

**Examples**

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

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

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

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

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

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

## End(Not run)

## Not run:
## formula notation with long data arrangement
NZScienceTeachingLong <- reshape2::melt(NZScienceTeaching,
    id.vars=c("Question", "Subtable"))
names(NZScienceTeachingLong)[3] <- "Agreement"
head(NZScienceTeachingLong)
likert(Question ~ Agreement | Subtable, value="value", data=NZScienceTeachingLong,
       ylab=NULL,
       scales=list(y=list(relation="free")), layout=c(1,2))

## End(Not run)

## Examples with higher-dimensional arrays.
tmp3 <- array(1:24, dim=c(2,3,4),
              dimnames=list(A=letters[1:2], B=LETTERS[3:5], C=letters[6:9]))
## positive.order=FALSE is the default. With arrays
## the rownames within each item of an array are identical.

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

## plot.likert interprets vectors as single-row matrices.
## http://survey.cvent.com/blog/customer-insights-2/box-scores-are-not-just-for-baseball
Responses <- c(15, 13, 12, 25, 35)
names(Responses) <- c("Strongly Disagree", "Disagree", "No Opinion",
                   "Agree", "Strongly Agree")
## Not run:
likert(Responses, main="Retail-R-Us offers the best everyday prices.",
      sub="This plot looks better in a 9in x 2.6in window.")
## End(Not run)
## reverse=TRUE is needed for a single-column key with
## horizontal=FALSE and with space="right"
likert(Responses, horizontal=FALSE,
      aspect=1.5,
      main="Retail-R-Us offers the best everyday prices.",
      auto.key=list(space="right", columns=1,
                     reverse=TRUE, padding.text=2),
      sub="This plot looks better in a 4in x 3in window.")
## Not run:
## Since age is always positive and increases in a single direction,
## this example uses colors from a sequential palette for the age
## groups. In this example we do not use a diverging palette that is
## appropriate when groups are defined by a characteristic, such as
## strength of agreement or disagreement, that can increase in two directions.
## Initially we use the default Blue palette in the sequential_hcl function.
data(AudiencePercent)
likert(AudiencePercent,
      auto.key=list(between=1, between.columns=2),
      xlab=paste("Percentage of audience younger than 35 (left of zero),
                  and older than 35 (right of zero)"),
      main="Target Audience",
      col=rev(colorspace::sequential_hcl(4)),
      sub="This plot looks better in a 7in x 3.5in window.")
## The really light colors in the previous example are too light.
## Therefore we use the col argument directly. We chose to use an
## intermediate set of Blue colors selected from a longer Blue palette.
likert(AudiencePercent,
      positive.order=TRUE,
      auto.key=list(between=1, between.columns=2),
      xlab=paste("Percentage of audience younger than 35",
      ..."Percentage of audience older than 35")...)
"(left of zero) and older than 35 (right of zero)",
main="Brand A has the most even distribution of ages",
col=colorspace::sequential_hcl(11)[5:2],
scales=list(x=list(at=seq(-90,60,10),
            labels=as.vector(rbind("",seq(-80,60,20))))),
sub="This plot looks better in a 7in x 3.5in window.")

## End(Not run)

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

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

## End(Not run)

Pop <- rbind(a=c(3,2,4,9), b=c(6,10,12,10))
dimnames(Pop)[[2]] <- c("Very Low", "Low", "High", "Very High")
likert(as.listOfNamedMatrices(Pop),
    as.percent=TRUE,
    resize.height="rowSums",
    strip=FALSE,
    strip.left=FALSE,
## Professional Challenges example.

The data for this example is a list of related likert scales, with each item in the list consisting of differently named rows. The data is from a questionnaire analyzed in a recent Amstat News article. The study population was partitioned in several ways. Data from one of the partitions (Employment sector) was used in the first example in this help file. The examples here show various options for displaying all partitions on the same plot.

```{r}
data(ProfChal)
levels(ProfChal$Subtable)[6] <- "Prof Recog"  # reduce length of label

## 1. Plot counts with rows in each panel sorted by positive counts.

likert(Question ~ . | Subtable, ProfChal,
  positive.order=TRUE,
  main="This works, but needs more specified arguments to look good")

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

## Not run:

likert(Question ~ . | Subtable, ProfChal,
  positive.order=TRUE,
  main="This works, but needs more specified arguments to look good")

## (Not run)

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

ProfChalCountsPlot

## Not run:

## 2. Plot percents with rows in each panel sorted by positive percents.
```
## This is a different sequence than the counts. Row Count Totals are displayed on the right axis.

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

ProfChalPctPlot

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

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

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

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

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

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

## Not run:

`5. putting the x-axis tick labels on top for horizontal plots
   putting the y-axis tick labels on right for vertical plots`

This non-standard specification is a consequence of using the right axis labels for different values than appear on the left axis labels with horizontal plots, and using the top axis labels for different values than appear on the bottom axis labels with vertical plots.

## Percent plot calculated automatically from Count data

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

tmpv <- likert(Question ~ ., ProfChal[ProfChal$Subtable=="Employment sector",], as.percent=TRUE, main="Is your job professionally challenging?", sub="likert plots with long Question names look better horizontally. With effort they can be made to look adequate vertically.", horizontal=FALSE, scales=list(y=list(alternating=2), x=list(rot=c(90, 0))), ylab.right="Percent", ylab=NULL, xlab.top="Column Count Totals", par.settings=list( layout.heights=list(key.axis.padding=5), layout.widths=list(key.right=1.5, right.padding=0)
})
tmpv$y.scales$labels
names(tmpv$y.scales$labels) <- tmpv$y.scales$labels
```

tmpv
likertColor

Selection of colors for Likert plots.

Description

Selection of colors for Likert plots.

Usage

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

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

n, nc
Number of colors in the palette. If there are more levels than RColorBrewer normally handles, we automatically interpolate with colorRampPalette.

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

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

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

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

... Other arguments are ignored.

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

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

Details

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

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

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

Value

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

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

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

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

**Author(s)**
Richard M. Heiberger, with contributions from Naomi B. Robbins <naomi@nbr-graphs.com>. Maintainer: Richard M. Heiberger <rmh@temple.edu>

**See Also**
`plot.likert`

**Examples**

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

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

data(ProfDiv)
ProfDiv.df <- data.frame(ProfDiv)
likert(~ ., ProfDiv.df, horizontal=FALSE, positive.order=FALSE)
likert(~ ., ProfDiv.df, horizontal=FALSE, positive.order=FALSE, colorFunctionOption="default")
likert(~ ., ProfDiv.df, horizontal=FALSE, positive.order=FALSE, colorFunctionOption="flatter")
likert(~ ., ProfDiv.df, horizontal=FALSE, positive.order=FALSE, colorFunction="sequential_hcl")
likert(~ ., ProfDiv.df, horizontal=FALSE, positive.order=FALSE, colorFunction="sequential_hcl", colorFunctionOption="default")
likert(~ ., ProfDiv.df, horizontal=FALSE, positive.order=FALSE, colorFunction="sequential_hcl", colorFunctionOption="flatter")
likert(ProfDiv, horizontal=FALSE, positive.order=FALSE)
likert(ProfDiv, horizontal=FALSE, positive.order=FALSE, colorFunctionOption="default")
likert(ProfDiv, horizontal=FALSE, positive.order=FALSE,
```
## likertMosaic

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

### Description

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

### Usage

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

## S3 method for class 'array'  
likertMosaic(x, ReferenceZero = NULL, col = NULL, main = NULL, ... , as.percent = FALSE, variable.width = NULL, positive.order = FALSE, Conditions = NULL)
```

## Examples

```r
likertMosaic(ProfDiv.df)
```
likertMosaic

x.legend = list(text = list(dimnames(x)[[ndim]]),
    columns = x.dim[ndim],
    space = "bottom",
    size = 2,
    cex = 0.8,
    between = 0.6,
    rect = list(col = col, border = "white"),
    legend.y = 0.05,
    spacing = spacing_highlighting,
    split_vertical = c(TRUE, FALSE),
    margins = c(3, 2, 4, 22),
    keep_aspect = FALSE,
    rot_labels = c(0, 0, 90, 0),
    just_labels = c("center", "center", "center", "right"),
    labels = c(TRUE, TRUE, FALSE, TRUE),
    varnames = FALSE,
    zero_size = 0,
    gp = gpar(fill = col.extended, col = 0),
    colorFunction = "diverge_hcl",
    colorFunctionOption = "lighter")

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

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

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

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

Arguments

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

data For the formula method, a data.frame.

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

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

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

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

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

**colorFunction, colorFunctionOption** See `likertColor`.

**main** main title for the plot.

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

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

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

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

**legend.y** Adjust vertical location of legend.

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

**varnames, zero_size, gp** Please see `strucplot` for details.

**Details**

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

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

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

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

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

Value

Please see strucplot for a description of the returned object.

Note

The functions described here are currently missing the following features:

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

Richard M. Heiberger, with contributions from Naomi B. Robbins <naomi@nbr-graphs.com>.
Maintainer: Richard M. Heiberger <rmh@temple.edu>

References


See Also

likert.mosaic

Examples

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

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

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

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

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

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

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

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

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

See Also
likert

Examples

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

levels(ProfChal$Subtable)[6] <- "Prof Recog" ## reduce length of label
## See ?print.TwoTrellisColumns for this example using the original ordering
## Order both the plot of the count plot and the percent plot by the
## positive.order of the percent plot.
LikertPercentCountColumns(Question ~ . | Subtable, ProfChal,
    layout=c(1,6), scales=list(y=list(relation="free")),
    ylab=NULL, between=list(y=0),
    strip.left=strip.custom(bg="gray97"), strip=FALSE,
    par.strip.text=list(cex=.7),
    positive.order=TRUE,
    main="Is your job professionally challenging?")

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

## Order both the plot of the count plot and the percent plot by the
## positive.order of the percent plot.
## Just the "Employment sector".
LPCCEs <-
LikertPercentCountColumns(Question ~ . ,
    ProfChal[ProfChal$Subtable == "Employment sector", -7],
    ylab=NULL, between=list(y=0),
    par.strip.text=list(cex=.7),
    positive.order=TRUE,
```
likertWeighted

Special case wrapper for likert() when multiple columns are to have the same bar thicknesses. Uses formula with one or two conditioning variables.

Description

Special case wrapper for likert() when multiple columns are to have the same bar thicknesses. Uses formula with one or two conditioning variables.

Usage

likertWeighted(x, ...) ## generic

## S3 method for class 'array'
likertWeighted(x, ..., C = 1, Q = 3, R = 2) ## array

## Default S3 method:
likertWeighted(x, ...) ## matrix, table, data.frame

## S3 method for class 'formula'
likertWeighted(x, data,

xlim=c(-100, 100),
scales=list(y=list(relation="free", cex=1.3),

x=list(at=seq(-100, 100, 50),

labels=abs(seq(-100, 100, 50)), cex=.5)),

box.ratio=1000,

as.percent=TRUE, rightAxis=FALSE,

between=list(x=1, y=0),

strip=FALSE, strip.left=FALSE,

par.settings=list(clip=list(panel="off")),

h.resizePanels=1,

auto.key.title=NULL,

auto.key.columns=dim(data)[[2]] - NumberOfConditioningVariables(formula), ## excludes conditioning variables
Arguments

x
For the default method, a matrix or data.frame or two-dimensional table. For the array method, a two- or three-dimensional array. For the formula method, a formula.

formula
Standard trellis formula, usually ~ . | row + column or ~ . | row

data
A data.frame that has been constructed from a 2D object (matrix or table or data.frame) to include an additional column row, or constructed from a 3D array by toCQxR to include two additional columns group and row. The default and array methods do that construction.

C, R, Q
Integers, one each of 1,2,3; positions of the three dimensions. Used in array method. See toCQxR.

xlim, between, strip, strip.left, par.settings, ylab
See xyplot

scales
See xyplot. For likertWeighted, when scales for x is changed, scales for y must be stated also.

box.ratio
See panel.bwplot.

as.percent, rightAxis, ..., h.resizePanels
First see the formula method for likertWeighted, and then likert.

auto.key.title, auto.key.columns, auto.key.cex, auto.key.cex.title, auto.key.lines.title
Values which will be used in trellis argument auto.key=list(title=auto.key.title, columns=auto.key.columns, cex=auto.key.cex, cex.title=auto.key.cex.title, lines.title=auto.key.lines.title)

axis.top
Label to be placed at x=0 for top (and other specified) panel of each column.

axis.top.row
Which rows will have axis.top displayed.

Value

A likert plot as a "trellis" object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert
Examples

```r
## simplest 2D example
tmp <- matrix(1:12, 3, 4,
  dimnames=list(c("A","B","C"),
  c(letters[4:7]))) * c(1,2,3)
tmp
rowSums(tmp)
likertWeighted(tmp,
  h.resizePanels=rowSums(tmp),
  main="likertWeighted, simplest example,
  defaults to Percent,
  specified row thicknesses")

## Same example with explicit use of the formula method
## (default method does this for you).
tmpdd <- data.frame(tmp, row=row.names(tmp))
tmpdd
likertWeighted(~ . | row, tmpdd, ## tmpdd
  h.resizePanels=rowSums(tmp), ## tmp
  main="likertWeighted,
  same example but with explicit formula method")

## show subgroups
likertWeighted(tmp,
  h.resizePanels=rowSums(tmp),
  between=list(y=c(0,1)),
  ylab=c("C in its own group","A and B together"),
  main="between=list(y=c(0,1)) ## standard lattice between argument
Adjacent A and B with y.between = 0 are in the same bordered group.
Adjacent B and C with y.between != 0 are in different bordered groups.")

## simplest 3D example
## This is natural when multiple questions are asked of the
## same set of respondents in a survey.
## This example simulates that situation.
##
## tmp3D <- abind::abind(h=tmp, i=tmp, j=tmp, along=3)
tmp3D[1,"i"] <- tmp3D[1,c(4,2,1,3),"h"]
tmp3D[2,"i"] <- tmp3D[2,c(2,4,3,1),"h"]
tmp3D[3,"i"] <- tmp3D[3,c(4,1,2,3),"h"]
tmp3D[1,"j"] <- tmp3D[1,c(4,3,2,1),"h"]
tmp3D[2,"j"] <- tmp3D[2,c(1,4,3,2),"h"]
tmp3D[3,"j"] <- tmp3D[3,c(2,4,3,1),"h"]

## now
rowSums(tmp3D[,1]) == rowSums(tmp3D[,2])
rowSums(tmp3D[,1]) == rowSums(tmp3D[,3])
likertWeighted(tmp3D, h.resizePanels=rowSums(tmp3D[,1]),)
```

likertWeighted
Description

Special case wrapper for likert() when multiple columns are to have the same bar thicknesses. Uses formula. Each row of the data is placed in its own trellis panel. Optionally, adjacent panels can be grouped in a common border. likertWeighted47() was introduced in HH_3.1-47 with the name likertWeighted(), and is now deprecated in favor of a redesigned likertWeighted(). All examples in likertWeighted47() work unchanged with likertWeighted(), although with slightly different appearance. The likertWeighted47() appearance can be produced with adjusted arguments in the new likertWeighted().

Usage

likertWeighted47(x, data, xlim=c(-100,100),
   x.at=seq(-100,100,25), x.labels=abs(x.at),
   h.resizePanels=data$rowheights,
   rightAxisLabels=format(round(h.resizePanels, digits),
                           big.mark=","),
   ylab=NULL,
   axis.key.padding=10,
   y.tck=c(0,3),
   layout=c(1, nrow(data)),
   as.percent=TRUE,
   line.color="black",
   box.ratio=1000,
   digits=-3,
   scales.cex=1,
   between=list(y=0),
   layer=TRUE,
   ...)
Arguments

- **x**: A formula of the form `~ a + b + c | panelnames` where `a`, `b`, `c` are columns in `data` and `panelnames` is a column of data that will normally be a copy of `rownames(data)`. The values in `panelnames` must be unique. The formula `~ . | panelnames` where `.` stands for the `'+'` of all variables in the `data.frame` other than `panel.names` may be used.

  This formula will be passed to `likert`

- **data**: `data.frame`

- **x.at, x.labels**: Location of `x` tick-marks and value of corresponding labels. Same meaning as `scales=list(x=list(at=seq(-100,100,25), labels=abs(seq(-100,100,25))))`

- **between**: Standard argument to lattice functions, see `xyplot`. In addition, adjacent panels with zero-valued `y.between` between them will be grouped within the same borders, and adjacent panels with non-zero-valued `y.between` between them will be in separately bordered groups. See example EEE.

- **xlim, ylab, layout, box.ratio**: Standard arguments to lattice functions, see `xyplot`.

- **digits**: See `round`

- **axis.key.padding**: See `trellis.par.get`

- **as.percent, rightAxisLabels, h.resizePanels**

  See `likert`

- **scales.cex**: Abbreviation for `scales=list(cex=scales.cex)`

- **y.tck**: Abbreviation for `scales=list(y=list(tck=y.tck))`

- **line.color**: Color to be used to surround sub-panels when `between=list(y=v)` has a non-zero vector for `v`.

- **layer**: Logical, default `TRUE`. If `TRUE` then put a border around the sets of grouped panels. If `FALSE`, then don’t put a border.

- **...**: Additional arguments that will be passed through to `likert` and from there to `lattice`.

Details

In the standard `likert()` plot, all stacked bars have the same thickness. The `likertWeighted47` function weights the thickness of the bars by `h.resizePanels`. When used with `as.percent=TRUE`, the `i`th stacked bar is 100% wide and `h.resizePanels[i]` units high. As a consequence one square inch of plotting surface represents the same number of objects in all bars and sub-bars. Compare AAA, BBB, and CCC in the examples to see the effect.

Value

A “trellis” object containing the plot. The plot will be automatically displayed unless the result is assigned to an object.
Author(s)

Richard M. Heiberger, with contributions from Naomi B. Robbins <naomi@nbr-graphs.com>. Maintainer: Richard M. Heiberger <rmh@temple.edu>

See Also

likert

Examples

tmp <- data.frame(matrix(1:12, 3, 4,
    dimnames=list(c("A","B","C"),
        c(letters[4:7])))
tmpcc <- cbind(tmp, rowheights=rowSums(tmp))
AAA <- likert(~ . , data=tmp, rightAxisLabels=tmpcc$rowheights,
    main="The total width of each bar is the sum of its magnitudes.
The width of each sub-bar is its magnitude. All bars have the same thickness.")
AAA

BBB <- likert(~ . , data=tmp, as.percent=TRUE, rightAxisLabels=tmpcc$rowheights,
    main="The total width of each bar is 100
The width of each sub-bar is its proportion of its row. All bars have the same thickness.")
BBB

CCC <- likertWeighted47(~ . | rowheights, tmpcc,
    as.percent=TRUE, xlim=c(-100,100),
    rightAxisLabels=tmpcc$rowheights,
    main="The total width of each bar is 100
The width of each sub-bar is its proportion of its row. The thickness of each bar is the unique sum of its magnitudes.")
CCC

tmpdd <- cbind(tmp, rownames=row.names(tmp))
likertWeighted47(~ . | rownames, tmpdd,
    as.percent=TRUE, xlim=c(-100,100),
    rightAxisLabels=tmpcc$rowheights,
    main="Equal row thicknesses")

likertWeighted47(~ . | rownames, tmpdd,
    as.percent=TRUE, xlim=c(-100,100),
    rightAxisLabels=tmpcc$rowheights,
    h.resizePanels=rowSums(tmp),
    main="Variable row thicknesses")

likertWeighted47(~ . | rownames, tmpdd,
lm.case

Description

Case statistics for regression analysis. `case.lm` calculates the statistics. `plot.case` plots the cases, one statistic per panel, and illustrates and flags all observations for which the standard thresholds are exceeded. `plot.case` returns an object with class `c("trellis.case", "trellis")` containing the plot and the row.names of the flagged observations. The object is printed by a method which displays the set of graphs and prints the list of flagged cases. `panel.case` is a panel function for `plot.case`. 

```R
DDD <- likertWeighted47(~ . | rownames, tmpdd,
  as.percent=TRUE, xlim=c(-100,100),
  rightAxisLabels=paste(tmpcc$rowheights, c(1,3,3), sep=" ", ),
  ylab.right="RowCount Total\nThicknesses Explicitly Specified",
  h.resizePanels=c(1,3,3),
  main="The total width of each bar is 100
The thickness of each sub-bar is its proportion of its row.
The thickness of each bar is the value of h.resizePanels.")

## comparisons
AAA
BBB
CCC
DDD

EEE <- likertWeighted47(~ . | rownames, tmpdd,
  as.percent=TRUE, xlim=c(-100,100),
  rightAxisLabels=rowSums(tmp),
  between=list(y=c(0,1)),
  ylab=c("C in its own group","A and B together"),
  main="Illustrating between: between=list(y=c(0,1))
Adjacent A and B with y.between = 0 are in the same bordered group.
Adjacent B and C with y.between != 0 are in different bordered groups.")

EEE
```
Usage

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

## S3 method for class 'case'
plot(x, fit,
    which=c("stu.res","si","h","cook","dffits",
        dimnames(x)[[2]][-(1:8)]),  ##DFBETAS
    between.in=list(y=4, x=9),
    cex.threshold=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,
    main.cex=NULL,
    ...
)

panel.case(x, y, subscripts, rownames, group.names,
    thresh, case.large,
    nn, pp, ss, cex.threshold,
    ...)

Arguments

fit "lm" object computed with x=TRUE
lms summary.lm(fit)
lmi lm.influence(fit)
x In plot.case, the matrix output from case.lm containing case diagnostics on 
each observation in the original dataset. In panel.case, the x variable to be 
plotted
which In plot.case, the names of the columns of x that are to be graphed.
between.in between trellis/lattice argument.
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.
main.cex cex for main title.
... other arguments to xyplot
y the y variable to be plotted.
thresh Named list of lists. Each list contains the components threshold (Sy$locations 
where a reference line will be drawn), thresh.label (the right-axis labels for the 
reference lines), thresh.id (the bounds defining "Noteworthy Observations").
```r

case.large   Named list of "Noteworthy Observations".
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.
```

**Details**

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

**Value**

`case.lm` returns a matrix, with one row for each observation in the original dataset. The columns contain the diagnostic statistics: `e` (residuals), `h*` (hat diagonals), `si*` (deleted standard deviation), `sta.res` (standardized residuals), `stu.res*` (Studentized deleted 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 `c("trellis.case", "trellis")` object containing the plot (including the starred columns by default) and also retains the row.names of the flagged observations in the `$panel.args.common$case.large` component. The print method for the `c("trellis.case", "trellis")` object prints the graph and the list of flagged observations.

`panel.case` is a panel function for `plot.case`.

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**References**


**See Also**

`lm.influence`.
Examples

```r
data(kidney)

kidney2.lm <- lm(clearance ~ concen + age + weight + concen*age,
                 data=kidney,
                 na.action=na.exclude)  ## recommended

kidney2.case <- case(kidney2.lm)

## this picture looks much better in portrait, specification is device dependent
plot(kidney2.case, kidney2.lm, par.strip.text=list(cex=.9),
     layout=c(2,3))
```

---

**lm.regsubsets**

Evaluate lm model with highest adjusted $R^2$.

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

"lm" object for the selected model.

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**See Also**

- `lm`, `regsubsets`
Description

lmatPairwise

Usage

lmatPairwise(x, ...)  
## S3 method for class 'matrix'  
lmatPairwise(x, levels, ...)  
## S3 method for class 'glht'  
lmatPairwise(x, ...)  
## S3 method for class 'mmc.multicomp'  
lmatPairwise(x, ...)  
## S3 method for class 'mmc'  
lmatPairwise(x, ...)

Arguments

x  
...  
levels  
levels

Details

details

Value

matrix

Author(s)

rmh

See Also

mmc, mcp

Examples

data(catalystm)  
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)  
catalystm.mmc <- mmc(catalystm1.aov)  
lmatPairwise(catalystm.mmc)
**lmatRows**

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

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

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

Richard M. Heiberger <rmh@temple.edu>

See Also

mmc,
glht.

Examples

```r
## catalystm example
## See ?MMC for more on this example
data(catalystm)
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
catalystm.mmc <-
  if.R(r=mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey")),
    s=multicomp.mmc(catalystm1.aov, plot=FALSE))
dimnames(catalystm.mmc$mca$lmat)[[1]]
lmatRows(catalystm1.aov, focus="catalyst")

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

---

**lmplot**

*Four types of residual plots for linear models.*

Description

Four types of residual plots for linear models. The first three are redesigns of plots that stats:::plot.lm presents. The first two show the positive residuals in `col[2]` and the negative residuals in color `col[1]`. The third and fourth use color `col[1]`. The fourth is based on an S-Plus panel that R\ does’t provide.

Usage

```r
lmplot(lm.object, ..., main=NULL,
       col=trellis.par.get("superpose.symbol")$col[1:2],
       ylim=NULL)
```
Arguments

- **lm.object**: Linear model object. See `lm` for details.
- **col**: Vector of color names. Only the first two are used. If not specified, then `trellis.par.get("superpose.symbol")$col[1:2]` is used.
- **main**: Standard main title for plots.
- **ylim**: Standard `lattice` argument. It is used as specified for the `residVSfitted`, `diagQQ`, and `diagplot5new` plots. For the `scaleLocation` plot, the `ylim` is modified to `c(0, max(abs(ylim)))`. The main reason for using the `ylim` argument is to allow visual comparison of the residuals for two different models on the same scale.
- **...**: Other arguments, currently ignored.

Details

The trellis plots from the four functions `residVSfitted`, `scaleLocation`, `diagQQ`, `diagplot5new` are displayed on the current device in a coordinated display.

Value

A list of three trellis objects is returned invisibly, the first contains the result of `residVSfitted` and `scaleLocation` together. The second `diagQQ`, and the third `diagplot5new`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

- `residVSfitted`, `scaleLocation`, `diagQQ`, `diagplot5new`.

Examples

```r
tmp <- data.frame(y=rnorm(100), x1=rnorm(100), x2=rnorm(100))
tmp.lm <- lm(y ~ x1 + x2, data=tmp)
lmplot(tmp.lm)
```

---

**logit**  

*Logistic and odds functions and their inverses.*

Description

Logistic and odds functions and their inverses.
Usage

logit(p)
antilogit(x)

odds(p)
antiodds(o)

Arguments

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

Value

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

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

odds(seq(0, 1, .1))
antiodds(odds(seq(0, 1, .1)))

matrix.trellis

Convert a one-dimensional trellis object to a two-dimensional trellis object. This permits combineLimits and useOuterStrips to work.

Description

matrix.trellis

Usage

matrix.trellis(x = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)

## S3 method for class 'trellis'
as.matrix(x, ..., row = FALSE, yname)
Arguments

\begin{itemize}
\item \textbf{x}
\item \textbf{y}
\item \text{ncol, nrow, byrow, dimnames}
\end{itemize}

See \texttt{matrix}.

\textbf{row} Logical. The default is FALSE to match the behavior of the generic \texttt{as.matrix}. I think TRUE usually looks better.

\textbf{yname} Character. Provides the name of the generated conditioning factor.

\textbf{...} Other arguments are ignored.

Details

\texttt{matrix.trellis} lets the user specify \texttt{nrow} and \texttt{ncol}. \texttt{as.matrix.trellis} produces either be a single column (by default) or a single row.

Value

trellis object with \texttt{length(dim(x)) == 2} and specified \texttt{nrow} and \texttt{ncol}.

Author(s)

Richard M. Heiberger \texttt{<rmh@temple.edu>}

Examples

\begin{verbatim}
tmp <- data.frame(a=1:3,
                  b=c(4,5,7),
                  c=5:7,
                  d=c(8, 9, 12),
                  e=9:11)

tmp
a1 <- xyplot(a + b ~ c + d + e, data=tmp, outer=TRUE, main="a1")
a1
dim(a1)
a2 <- xyplot(a + b ~ c + d + e, data=tmp, outer=TRUE,
             scales=list(relation="free"), main="a2")
a2
dim(a2)
try(combineLimits(a2))
combineLimits.trellisvector(a2)
combineLimits.trellisvector(update(a2, layout=c(3,2)))

a21 <- matrix.trellis(a2, ncol=3, nrow=2, byrow=TRUE)
a21 <- update(a21, main="a21")
a21
dim(a21)
a21$x.scales$at
combineLimits(a21)
a22 <- update(a21, main="a22")
\end{verbatim}
mcalinfct

MCA multiple comparisons analysis (pairwise)

Description

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

Usage

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

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

- **model**: aov object
- **focus**: name of one of the factors in the model, as a character object.
- **mmmm.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

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

mmc(model, ...) ## R

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

## Default S3 method:
mmc(model, lm object

linfct=NULL,
focus=
if (is.null(linfct))
{
  if (length(model$contrasts)==1) names(model$contrasts)
  else stop("focus or linfct must be specified.")
} else
{
  if (is.null(names(linfct)))
    stop("focus must be specified.")
  else names(linfct)
},
focus.lmat,
ylabel=deparse(terms(model)[[2]]),
lmat=if (missing(focus.lmat))
  t(linfct)
else {
  lmatContrast(t(none.glht$linfct), focus.lmat)
},
lmat.rows=lmatRows(model, focus),
lmat.scale.abs2=TRUE,
estimate.sign=1,
order.contrasts=TRUE,
level=.95,
calpha=NULL,
alternative = c("two.sided", "less", "greater"),
...
)

multicomp.mmc(x, ## S-Plus

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

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

Arguments

model "aov" object in "lm" method.
ylabel name of the response variable.
lmat contrast matrix as in the S-Plus multicomp. The convention for lmat in R is to use the transpose of the linfct component produced by glht. Required for user-specified contrasts.
lmat.rows rows in lmat for the focus factor.
focus define the factor to compute contrasts of. In R this argument often can be used to simplify the call. The statement mmc(my.aov, focus="factorA") is interpreted as mmc(my.aov, factorA="Tukey", `interaction_average`=TRUE, `covariate_average`=TRUE) With TRUE, TRUE, multcomp::glht always gives the same result as the S-Plus multcomp function. Without the TRUE, TRUE, multcomp::glht gives a different answer when there are interactions or covariates in the model. See glht.
focus.lmat R only. Contrast matrix used in the user-specified comparisons of the focus factor. This is the matrix the user constructs. Row names must include all levels of the factor. Column names are the names the user assigns to the contrasts. Each column must sum to zero. See catalystm.lmat in the Examples section for an example. The focus.lmat matrix is multiplied by the lmat from the none component to create the lmat for the user-specified contrasts. Display the hibrido.lmat and maiz2.lmat in the maiz example below to see what is happening.
linfct In R, see glht.
... other arguments. alternative and base are frequently used with glht.
comparisons argument to multicomp
lmat.scale.abs2 logical, scale the contrasts in the columns of lmat to make the sum of the absolute values of each column equal 2.
estimate.sign numeric. If 0, leave contrasts in the default lexicographic direction. If positive, force all contrasts to positive, reversing their names if needed (if contrast A-B is negative, reverse it to B-A). If negative, the force all contrasts to positive.
order.contrasts

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

alternative

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

level

Confidence level. Defaults to 0.95.

crit.point, calpha

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

plot

logical, display the plot if TRUE.

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

arguments to plot.mmc.multicomp.

x, drop

See "[".

Details

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

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

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

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

Value

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

mca

Object containing the pairwise comparisons.

none

Object comparing each mean to 0.

lmat

Object for the contrasts specified in the lmat argument.

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

The multiple comparisons calculations in R and S-Plus use completely different functions. MMC plots in R are constructed by `mmc` based on `glht`. MMC plots in S-Plus are constructed by `multicomp.mmc` based on the S-Plus `multicomp`. The MMC plot is the same in both systems. The details of getting the plot differ.

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

There are two plotting functions for MMC plots. `mmcplot`, the newer `lattice`-based function, is recommended. `mmcplot`, chooses better default values for its arguments and is better coordinated with the tiebreaker plot.

The older `plot.mmc.multicomp`, built on `base` graphics, chooses sensible defaults for its many arguments, but they still 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**

`mmcplot`, `plot.mmc.multicomp`, `as.multicomp`

**Examples**

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

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

bwplot(concent ~ catalyst, data=catalystm,
scales=list(cex=1.5),
```

ylab=list("concentration", cex=1.5),
xlab=list("catalyst",cex=1.5))

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

catalystm.mca <-
glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey"))
confint(catalystm.mca)
plot(catalystm.mca)  ## multcomp plot
mmcplot(catalystm.mca, focus="catalyst")  ## HH plot

## pairwise comparisons

catalystm.mmc <-
  mmc(catalystm1.aov, focus="catalyst")
catalystm.mmc

## Not run:
## these three statements are identical for a one-way aov

  mmc(catalystm1.aov)  ## simplest
  mmc(catalystm1.aov, focus="catalyst")  ## generalizes to higher-order designs
  mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey"))  ## glht arguments

## End(Not run)

mmcplot(catalystm.mmc, style="both")

## User-Specified Contrasts
## Row names must include all levels of the factor.
## Column names are the names the user assigns to the contrasts.
## Each column must sum to zero.
catalystm.lmat <- cbind("AB-D" =c( 1, 1, 0,-2),

  "A-B" =c( 1,-1, 0, 0),

  "ABD-C"=c( 1, 1,-3, 1))
dimnames(catalystm.lmat)[[1]] <- levels(catalystm$catalyst)
catalystm.lmat
catalystm.mmc <-
  mmc(catalystm1.aov,
  linfct = mcp(catalyst = "Tukey"),
  focus.lmat=catalystm.lmat)
catalystm.mmc

mmcplot(catalystm.mmc, style="both", type="lmat")

## Not run:
## Dunnett's test
## weightloss example
data(weightloss)
 bwplot(loss ~ group, data=weightloss,
weightloss.aov <- aov(loss ~ group, data=weightloss)
summary(weightloss.aov)

group.count <- table(weightloss$group)

tmp.dunnett <-
  glht(weightloss.aov,
  linfct=mcp(group=contrMat(group.count, base=4)),
  alternative="greater")
mmcplot(tmp.dunnett, main="contrasts in alphabetical order", focus="group")

tmp.dunnett.mmc <-
  mmc(weightloss.aov,
  linfct=mcp(group=contrMat(group.count, base=4)),
  alternative="greater")
mmcplot(tmp.dunnett.mmc,
  main="contrasts ordered by average value of the means\nof the two levels in the contrasts")

tmp.dunnett.mmc

## End(Not run)

## Not run:
## two-way ANOVA
## display example

data(display)

interaction2wt(time ~ emergenc * panel.ordered, data=display)

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

## multiple comparisons
## MMC plot
displayf.mmc <- mmc(displayf.aov, focus="panel")
displayf.mmc

## same thing using glht argument list
displayf.mmc <-
  mmc(displayf.aov,
  linfct=mcp(panel="Tukey", 'interaction_average'=TRUE, 'covariate_average'=TRUE))
mmcplot(displayf.mmc)

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

displayf.mmc <-
  mmc(displayf.aov, focus="panel", focus.lmat=panel.lmat)

## same thing using glht argument list
displayf.mmc <-
  mmc(displayf.aov,
    linfct=mcp(panel="Tukey", `interaction_average`=TRUE, `covariate_average`=TRUE),
    focus.lmat=panel.lmat)

mmcplot(displayf.mmc, type="lmat")

## End(Not run)

## Not run:
## split plot design with tiebreaker plot
##
## This example is based on the query by Tomas Goicoa to R-news
## http://article.gmane.org/gmane.comp.lang.r.general/76275/match=goicoa
## It is a split plot similar to the one in HH Section 14.2 based on
## Yates 1937 example. I am using the Goicoa example here because its
## MMC plot requires a tiebreaker plot.

data(maiz)

interaction2wt(yield ~ hibrido+nitrogeno+bloque, data=maiz,
  par.strip.text=list(cex=.7))
interaction2wt(yield ~ hibrido+nitrogeno, data=maiz)

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

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

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

## R glht() requires aov, not aovlist
maiz2.aov <- aov(terms(yield ~ bloque+nitrogeno + hibrido/nitrogeno,
    keep.order=TRUE),
    data=maiz)

summary(maiz2.aov)

## There are many ties in the group means.
## These are easily seen in the MMC plot, where the two clusters
## c("P3747", "P3732", "LH74") and c("Mol17", "A632")
## are evident from the top three contrasts including zero and the
## bottom contrast including zero. The significant contrasts are the
## ones comparing hybrids in the top group of three to ones in the
## bottom group of two.
We have two graphical responses to the ties.
1. We constructed the tiebreaker plot.
2. We construct a set of orthogonal contrasts to illustrate
   the clusters.

pairwise contrasts with tiebreakers.

```r
maiz2.mmc <- mmc(maiz2.aov,
  linfct=mcp(hibrido="Tukey", interaction_average=TRUE))
mmcplot(maiz2.mmc, style="both") # MMC and Tiebreaker
```

orthogonal contrasts

```r
hibrido.lmat <- cbind("PPL-MA" =c(2, 2,-3,-3, 2),
    "PP-L" =c(1, 1, 0, 0,-2),
    "P47-P32"=c(1,-1, 0, 0, 0),
    "M-A" =c(0, 0, 1,-1, 0))
dimnames(hibrido.lmat)[[1]] <- levels(maiz$hibrido)
hibrido.lmat

maiz2.mmc <-
  mmc(maiz2.aov, focus="hibrido", focus.lmat=hibrido.lmat)
maiz2.mmc
```

same thing using glht argument list

```r
maiz2.mmc <-
  mmc(maiz2.aov, linfct=mcp(hibrido="Tukey",
    interaction_average=TRUE), focus.lmat=hibrido.lmat)

mmcplot(maiz2.mmc, style="both", type="lmat")
```

End(Not run)

---

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

**Usage**

```r
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 argument for multicomp in S-Plus and to type in R glht
- **df**: scalar, residual degrees of freedom
- **sigmahat**: sqrt(MSE) from the ANOVA table
- **comparisons**: argument to S-Plus multicomp only.
- **estimate.sign**, **order.contrasts**, **lmat.rows**: See lmat.rows in mmc.
- **ry**: See argument ry.mmc in plot.mmc.multicomp.
- **crit.point**: See argument crit.point in S-Plus multicomp. The equivalent is not in glht.
- **iso.name**, **x.offset**: See plot.mmc.multicomp.
Value

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

Note

The multiple comparisons calculations in R and S-Plus use completely different functions. MMC plots in R are constructed by mmc based on glht. MMC plots in S-Plus are constructed by multicomp.mmc based on the S-Plus multicomp. The MMC plot is the same in both systems. The details of getting the plot differ.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

mmc

Examples

## This example is from Hsu and Peruggia
## This is the S-Plus version
## See ?aovSufficient for R

if.R(r={}
  s={
    data(pulmonary)
    pulmonary.aov <- aovSufficient(FVC ~ smoker,
        data=pulmonary)
    summary(pulmonary.aov)

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

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

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

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

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

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

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

## orthogonal contrasts
plot(pulmonary.mmc)

## pairwise and orthogonal contrasts on the same plot
plot(pulmonary.mmc, print.mca=TRUE, print.lmat=TRUE)
par(old.oms)
}
Control aspect ratio in MMC plots to maintain isomeans grid as a square.

Control aspect ratio in MMC plots to maintain isomeans grid as a square.

Usage

mmcAspect(trellis)

Arguments

trellis A trellis object. If there is more than one panel, the first panel will be used.

Value

New numeric aspect ratio that will force the isomeans grid to be a square rotated to have vertical and horizontal diagonals.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

mmcplot

Functions used by mmcplot.

Description

Functions used by mmcplot.

Usage

mmcisomeans(mmc, col=c("black","red"), lwd=c(1,1), lty=c(2,1),
  type = "mca", xlim = NULL, ylim = NULL, ...,
  axis.right=2.2,
  ylab=paste(
    mmc$none$ylabel, "means",
    " | ",
    mmc$none$focus, "level"),
  ylab.right=NULL,
  ...,
  ...)
`mmcisomeans`

```r
xlab="contrast value",
contrast.label=TRUE,
means.height=TRUE)

mmcmatch(mmc, col=c("black","red"), lwd=c(1,1), lty=c(2,1),
type = "mca", xlim = NULL, ylim = NULL, ...,
axis.right=2.2,
ylab=NULL,
ylab.right=NULL,
xlab="contrast value",
contrast.label=TRUE,
xlim.match=(type != "none"))

mmcboth(mmc, col=c("black","red"), lwd=c(1,1), lty=c(2,1),
type = "mca", h = c(0.7, 0.3), xlim = NULL, ylim = NULL, ...,
ylab.right=NULL, MMCname="MMC", Tiebreakername="Tiebreaker")
```

**Arguments**

- **mmc**: mmc object or other object as indicated by method.
- **type**: One of c("mca", "lmat", "linfct", "none"). For the default "mca", an MMC plot is drawn of the pairwise contrasts. For "lmat" or "linfct", an MMC plot is drawn of the contrasts specified to mmc in the lmat or linfct argument. For "none", a confidence interval plot for the group means is drawn.
- **h**: h argument for `resizePanels`.
- **xlim, ylim, xlab, ylab, ylab.right**: Standard `xyplot` arguments.
- **col, lwd, lty**: Standard `xyplot` arguments applied to the line segments representing the contrasts.
- **...**: Other arguments, to be forwarded to methods.
- **axis.right**: Value used internally for `par.settings=list(layout.widths=list(axis.right=axis.right))`. The user may need to set this in two circumstances. First, if the contrast names overflow the right edge of the plotting window, then use a larger value than the default. Second, if there is a ylab.right and it is too far away from the figure, then use a smaller value than the default.
- **contrast.label**: Logical. The default TRUE means place the word contrasts at the bottom of the right axis under the tick labels. FALSE means don’t place anything there.
- **MMCname, Tiebreakername**: Panel names when mmcplot is used with style="both".
- **xlim.match**: Logical. If TRUE, use xlim based on the contrasts in the mca component. If FALSE, use xlim based on the values of the estimates in the current component.
- **means.height**: Logical, with default value TRUE. When TRUE, then display the values of the group means as the left axis tick labels.
**Value**

A "trellis" object.

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**References**

See `mmc` for the references.

**See Also**

`mmc` for the discussion of the MMC. `mmcplot` for the user calls that get executed by the functions documented here.

**Examples**

```
## Not run:
## these examples exercise all optional arguments
data(catalystm)
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)

catalystm.glht <-
  glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey"))
confint(catalystm.glht)

plot(catalystm.glht) ## this is the multcomp:::plot.glht

mmcplot(catalystm.glht) ## mmcplot.glht sends its argument to HH:::as.multicomp.glht with
## the default arguments (estimate.sign = 1, order.contrasts = TRUE) unless overridden:
##
## mmcplot(catalystm.glht, order.contrasts=FALSE, estimate.sign=0, main="B'")

catalystm.lmat <- cbind("AB-D" =c(1, 1, 0,-2),
  "A-B" =c(1,-1, 0, 0),
  "ABD-C"=c(1, 1,-3, 1))
dimnames(catalystm.lmat)[[1]] <- levels(catalystm$catalyst)
catalystm.mmc <-
  mmc(catalystm1.aov,
    linfct = mcp(catalyst = "Tukey"),
    focus.lmat=catalystm.lmat)

mmcplot(catalystm.mmc, type="mca", style="confint")
mmcplot(catalystm.mmc, type="lmat", style="confint")
mmcplot(catalystm.mmc, type="none", style="confint")
mmcplot(catalystm.mmc, type="none", style="confint", xlim.match=FALSE,
  main="xlim.match=FALSE is default for none confint")
mmcplot(catalystm.mmc, type="none", style="confint", xlim.match=TRUE, main="out of bounds")

mmcplot(catalystm.mmc$mca, style="confint")
mmcplot(catalystm.mmc$lmat, style="confint")
```
MMC (Mean–mean Multiple Comparisons) plots in lattice

Description

MMC (Mean–mean Multiple Comparisons) plots in lattice
Usage

mmcplot(mmc, ...)  
## S3 method for class 'mmc'
mmcplot(mmc, col=c("black","red"), lwd=c(1,1), lty=c(2,1), ...,  
        style=c("isomeans", "confint", "both"),  
        type=c("mca", "lmat", "linfct", "none"))  
## S3 method for class 'glht'
mmcplot(mmc, col=c("black","red"), lwd=c(1,1), lty=c(2,1),  
        focus=mmc$focus, ...)  
## S3 method for class 'mmc.multicomp'
mmcplot(mmc, col=c("black","red"), lwd=c(1,1), lty=c(2,1), ...)  
## S3 method for class 'multicomp'
mmcplot(mmc, col=c("black","red"), lwd=c(1,1), lty=c(2,1), ...)  
## Default S3 method:
mmcplot(mmc, ...)

Arguments

mmc  
mmc object or other object as indicated by method.

col, lwd, lty  
Standard xyplot arguments applied to the line segments representing the contrasts.

focus  
Name of the factor for which the glht object was constructed.

...  
Other arguments to be passed on to the functions called by the methods.

style  
Style of graph: The default isomeans is the standard MMC plot with the isomeans grid. confint is a confidence interval plot, similar to the plot produced by multcomp::plot.glht. both prints both the isomeans and the confint plot as two panels of a trellis structure. When the underlying sets of means are close to each other, there will of necessity be overprinting in the isomeans panel and the confint panel will be needed as a tiebreaker. By default the xlim for the confint style will match the xlim of the corresponding isomeans plot.

type  
mca for the default paired-comparisons plot. lmat or linfct for a user-specified set of contrasts. none for confidence intervals on the set of group means (that is, no comparisons).

Value

A trellis object containing the graphs.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References

See mmc for the references.
See Also

`mmc` for the discussion of the MMC and for many examples. The functions `mmcisomeans`, `mmcmatch`, `mmcboth` are the internal functions that do the actual work of plotting.

Examples

```r
data(catalystm)
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
catalystm.mmc <-
  mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey"))
mmcpplot(catalystm.mmc)
mmcpplot(catalystm.mmc, style="both", MMCname="catalyst")
```

---

**mmcPruneIsomeans**

**MMC plots in lattice—suppress isomeans grid lines for specified levels of the factor.**

**Description**

MMC plots in lattice—suppress isomeans grid lines for specified levels of the factor.

**Usage**

```r
mmcPruneIsomeans(mmc, keep=NULL)
```

**Arguments**

- `mmc` An "mmc.multicomp" object.
- `keep` Index vector of rows of `mmc$none$table` that will be kept in the display.

**Value**

A modified "mmc.multicomp" object.

**See Also**

`mmc`

**Examples**

```r
## needed
## Not run:
## See file hh/scripts/hh2/tway.R for the complete example.
## A better example is needed for the .Rd documentation.
## possibly based on filmcoat temperature | pressure example.

data(rhiz.clover)
c(1,2,5,10,11,12)
rhiz.clover$cs <- with(rhiz.clover, interaction(comb, strain))
```
rhiz.clover.cs.aov <- aov(Npg ~ cs, data=rhiz.clover)
rhiz.clover.cs.aov

cs.mmc <- mmc(rhiz.clover.cs.aov, linfct=mcp(cs="Tukey"),
              calpha=qtukey(.95, 6, 48)/sqrt(2))
dimnames(cs.mmc$mca$lmat)[[2]]
cl.index <- grep("clover\.[[:print:]]*clover\.", dimnames, value=TRUE)

c.index
clover.lmat <- cs.mmc$mca$lmat[, cl.index] ## suppress "clover+alfalfa" contrasts
dimnames(clover.lmat)[[1]]
dimnames(clover.lmat)[[1]] <- levels(rhiz.clover$cs)
clover.lmat[1,] <- -colSums(clover.lmat[-1, ])
clover.lmat

csc.mmc <- mmc(rhiz.clover.cs.aov, linfct=mcp(cs="Tukey"),
              focus.lmat=clover.lmat,
              calpha=qtukey(.95, 6, 48)/sqrt(2))

## this example needs a window 11 inches high and 14 inches wide
mmcplot(csc.mmc, type="lmat", style="both")

## suppress "clover+alfalfa" means
csc.mmc.clover <- mmcPruneIsomeans(csc.mmc, keep = c(1,2,5,10,11,12))
csc.mmc.clover

## this example needs a window 11 inches high and 14 inches wide
mmcplot(csc.mmc.clover, type="lmat", style="both")

## End(Not run)

---

**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 mmcplot or the older 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 mmc in R, or multicomp.mmc in S-Plus
data(catalystm)

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

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

catalystm.mmc <-
    multicomp(catalystm1.aov, method="Tukey")
print(catalystm.mmc)

## the contrasts have been ordered by height (see ?MMC),
## which in this example corresponds to sort.order=c(1,2,4,3,5,6),
## and reversed, to make the contrast Estimates positive.
print(as.glht(catalystm.mmc$mca))

## ## For consistency with the S-Plus example,
## ## we change all factor level "A" to "control".
## as.glht(multicomp.label.change(catalystm.mmc$mca, "A", "control"))
},s=
{catalystm.mca <-
multicomp(catalystm1.aov, method="Tukey")
print(catalystm.mca)

catalystm.mmc <-
multicomp.mmc(catalystm1.aov, method="Tukey", plot=FALSE)
## the contrasts have been ordered by height (see ?MMC),
## which in this example corresponds to sort.order=c(1,2,4,3,5,6),
## and reversed, to make the contrast Estimates positive.
print(catalystm.mmc$mca)

## S-Plus multicomp already uses simple names. This function is
## therefore used in more complex two-way ANOVA examples. We illustrate
## here by changing all factor level "A" to "control".
print(multicomp.label.change(catalystm.mmc$mca, "A", "control"))
})
```

**multicomp.reverse**

Force all comparisons in a "multicomp" object to have the same sign.

Description

Force all comparisons in a "multicomp" object to have the same sign. If the contrast "A-B" has a negative estimate, reverse it show the contrast "B-A" with a positive estimate. If a contrast name does not include a minus sign "-" and the contrast is reversed, then an informative message is printed.
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

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

Plot a normal curve or a t-curve with both x (with mean and se as specified) and z or t (mean=0, se=1) axes. Shade a region for rejection region, acceptance region, confidence interval. The density axis is marked in units appropriate for the z or t axis. The existence of any of the arguments se, sd, n forces dual x and (z or t) scales. When none of these arguments are used, the main title defaults to "Standard Normal Density N(0,1)" and only the z scale is printed. A second density curve, appropriate for an alternative hypothesis is displayed when the argument axis.name="z1" is specified. The shaded area is printed on the plot.

When the optional argument df.t is specified, then a t-distribution with df.t degrees of freedom is plotted.

norm.observed plots a vertical line with arrowhead markers at the location of the observed xbar.

normal.and.t.dist is a driver function that uses all the others. It's primary function is drawing a plot. It returns an invisible list containing the values it calculated and displayed on the graph.

norm.curve draws the curves and filled areas as requested by the normal.and.t.dist function. Any out of bounds errors (for example, with normal.and.t.dist(deg.free=1)) are suppressed with par(err=-1) by this function and restored to the previous value when the norm.curve function completes.

**Usage**

```r
normal.and.t.dist(mu.H0 = 0,
mu.H1 = NA,
obs.mean = 0,
std.dev = 1,
n = NA,
deg.freedom = NA,
alpha.left = alpha.right,
alpha.right = .05,
Use.mu.H1 = FALSE,
Use.obs.mean = FALSE,
Use.alpha.left = FALSE,
Use.alpha.right= TRUE,
hypoth.or.conf = 'Hypothesis',
xmin = NA,
xmax = NA,
gxbar.min = NA,
gxbar.max = NA,
cex.crit = 1.2,
polygon.density= -1,
polygon.lwd = 4,
col.mean = 'limegreen',
```
col.mean.label = 'limegreen',
col.alpha = 'blue',
col.alpha.label= 'blue',
col.beta = 'red',
col.beta.label = 'red',
col.conf = 'palegreen',
col.conf.arrow = 'darkgreen',
col.conf.label = 'darkgreen'
)

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

norm.curve(mean=0, se=sd/sqrt(n),
critical.values=mean + se*c(-1, 1)*z.975,
z=if(se==0) 0 else
 do.call("seq", as.list(c(((par()$usr[1:2]-mean)/se, length=109)))))
 shade, col="blue",
 axis.name=ifelse(is.null(df.t) || df.t==Inf, "z", "t"),
 second.axis.label.line=3,
 sd=1, n=1,
 df.t=NULL,
 axis.name.expr=axis.name,
 Use.obs.mean=TRUE,
 col.label=col,
 hypoth.or.conf="Hypoth",
 col.conf.arrow=par("col"),
 col.conf.label=par("col"),
 col.crit=ifelse(hypoth.or.conf="Hypoth",
 col.conf.arrow="blue", col.conf.arrow),
 cex.crit=1.2,
 polygon.density=-1,
 polygon.lwd=4,
 col.border=ifelse(is.na(polygon.density), FALSE, col),
 ...)

norm.observed(xbar, t.xbar, t.xbar.H1=NULL,
 col="green",
 p.val=NULL, p.val.x=par()$usr[2]+ left.margin,
 t.or.z=ifelse(is.null(deg.free) || deg.free==Inf, "z", "t"),
 t.or.z.position=par()$usr[1]-left.margin,
 cex.small=par()$cex*.7, col.label=col,
 xbar.negt=NULL, cex.large=par()$cex,
left.margin=.15*diff(par()$usr[1:2]),
sided="", deg.free=NULL)
	norm.outline(dfunction, left, right, mu.H0, se, deg.free=NULL,
col.mean="green")

Arguments

xlim, ylim, xmin, xmax, gxbar.min, gxbar.max

xlim, ylim. Defaults to correct values for standard Normal(0,1). User must set
values for other mean and standard error.

mean

Mean of the normal distribution in xbar-scale, used in calls to dnorm.

se

standard error of the normal distribution in xbar-scale, used in calls to dnorm.

sd, std.dev, n

standard deviation and sample size of the normal distribution in x-scale. These
may be used as an alternate way of specifying the standard error se.

df.t, deg.freedom

Degrees of freedom for the t distribution. When df.t is NULL, the normal distri-
bution is used.

critical.values

Critical values in xbar-scale. A scalar value implies a one-sided test. A vector
of two values implies a two-sided test.

main

Main title.

z

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

shade

Valid values for shade are "right", "left", "inside", "outside", "none". Default is
"right" for one-sided critical.values and "outside" for two-sided critical values.

col

color of the shaded region.

col.label, col.alpha, col.alpha.label

color of the area of the shaded rejection region and its label.

col.beta, col.beta.label

color of the area of the shaded region for Type II error and its label.

hypoth.or.conf

"Hypoth" or "Conf"

col.conf

Color of plot within confidence limits.

col.conf.arrow

Color of arrow denoting confidence limits.

col.conf.label

Color of label giving confidence level.

col.mean.label

Color of label for observed mean.

col.crit, cex.crit

Color and cex of critical values.

axis.name, axis.name.expr

defaults to "z" for the standard normal scale centered on the null hypothes-

is 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. The axis.name.expr allows R
users to say expression(z[1]) to get real subscripts.
second.axis.label.line

Defaults to 3. Normally not needed. When two curves are drawn, one normal and one t, then the second curve needs a different label for the y-axis. Set this value to 4 to avoid overprinting.

xbar, obs.mean

xbar-value of the observed data.

t.val

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

... Other arguments which are ignored.

Use.obs.mean Logical. If TRUE, then include "mean" on the plot.

alpha.right, alpha.left

Area in tail of curve.

Use.alpha.right, Use.alpha.left Logical. If TRUE, then include the specified α on the plot.

t.val.H1 t-value under alternate hypothesis.

p.val under specified hypothesis

p.val.x.t.or.z.position location on x-axis to put label

t.or.z label for axis.

cex.small cex for left margin labels of axis.

xbar.negt location in data scale of negative t- or z-value corresponding to observed x-value.

Used for two-sided p-values.

cex.large cex for labels in top margin.

left.margin distance to the left of par()$usr[1].

sided type of test.

deg.free degrees of freedom or NULL.

dfunction "dnorm" or "dt"

left left end of interval

right right end of interval

mu.H0, mu.H1 mean under the null hypothesis and alternative hypothesis.

Use.mu.H1 Logical. If TRUE, then include mu.H1 on the plot.

col.mean Color of outline.

polygon.density, polygon.lwd, col.border density, lwd, border arguments to polygon. polygon.density is -1 by default to give a solid color filled region. Setting polygon.density to a positive value (we recommend 10) gives a diagonally-hatched area appropriate for printing the graph on a black and white printer.

Value

An invisible list containing the calculated values of probabilities and critical values in the data scale, the null hypothesis z- or t-scale, and the alternative hypothesis z- or t-scale, as specified. The components are: beta.left, beta.middle, beta.right, crit.val, crit.val.H1, crit.val.H1.left, crit.val.left, crit.val.left.z, crit.val.z, obs.mean.H0.p.val, obs.mean.H0.side, obs.mean.H0.z, obs.mean.H1.z, obs.mean.x.neg, obs.mean.x.pos, obs.mean.z.pos, standard, standard.error, standard.normal
norm.curve

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

Examples

```r
normal.and.t.dist()
normal.and.t.dist(xmin=-4)
normal.and.t.dist(std.dev=2)
normal.and.t.dist(std.dev=2, Use.alpha.left=TRUE, deg.free=6)
normal.and.t.dist(std.dev=2, Use.alpha.left=TRUE, deg.free=6, gxbar.max=.20)
normal.and.t.dist(std.dev=2, Use.alpha.left=TRUE, deg.free=6, 
gxbar.max=.20, polygon.density=10)
normal.and.t.dist(std.dev=2, Use.alpha.left=FALSE, deg.free=6, 
gxbar.max=.20, polygon.density=10, 
mu.H1=2, Use.mu.H1=TRUE, 
obs.mean=2.5, Use.obs.mean=TRUE, xmin=-7)
normal.and.t.dist(std.dev=2, hypoth.or.conf="Conf")
normal.and.t.dist(std.dev=2, hypoth.or.conf="Conf", deg.free=8)

old.par <- par(oma=c(4,0,2,5), mar=c(7,7,4,2)+.1)
norm.setup()
norm.curve()
norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(100, 5, 100+5*(1.645))
norm.observed(112, (112-100)/5)
norm.outline("dnorm", 112, par()$usr[2], 100, 5)
norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(100, 5, 100+5*(-1.645), shade="left")
norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(mean=100, se=5, col="red")
norm.setup(xlim=c(75,125), mean=100, se=5)
norm.curve(100, 5, 100+5*c(-1.96, 1.96))
norm.setup(xlim=c(-6, 12), se=2)
norm.curve(critical.values=2*1.645, se=2, mean=2*(1.645+1.281552), 
col='green', shade="left", axis.name="z1")
norm.curve(critical.values=2*1.645, col='red')
norm.setup(xlim=c(-6, 12), se=2)
norm.curve(critical.values=2*1.645, se=2, mean=2*(1.645+1.281552), 
col='green', shade="left", axis.name="z1")
norm.curve(critical.values=2*1.645, se=2, mean=0, 
col='red', shade="right")

par(mfrow=c(2,1))
norm.setup()
```
norm.curve()
mtext("norm.setup(); norm.curve()", side=1, line=5)
norm.setup(n=1)
norm.curve(n=1)
mtext("norm.setup(n=1); norm.curve(n=1)", side=1, line=5)
par(mfrow=c(1,1))

par(mfrow=c(2,2))

## naively scaled,
## areas under the curve are numerically the same but visually different
norm.setup(n=1)
norm.curve(n=1)
norm.observed(1.2, 1.2/(1/sqrt(1)))
norm.setup(n=2)
norm.curve(n=2)
norm.observed(1.2, 1.2/(1/sqrt(2)))
norm.setup(n=4)
norm.curve(n=4)
norm.observed(1.2, 1.2/(1/sqrt(4)))
norm.setup(n=10)
norm.curve(n=10)
norm.observed(1.2, 1.2/(1/sqrt(10)))
mtext("areas under the curve are numerically the same but visually different",
    side=3, outer=TRUE)

## scaled so all areas under the curve are numerically and visually the same
norm.setup(n=1, ylim=c(0,1.3))
norm.curve(n=1)
norm.observed(1.2, 1.2/(1/sqrt(1))), ylim=c(0,1.3))
norm.curve(n=2)
norm.observed(1.2, 1.2/(1/sqrt(2))), ylim=c(0,1.3))
norm.curve(n=4)
norm.observed(1.2, 1.2/(1/sqrt(4))), ylim=c(0,1.3))
norm.curve(n=10)
norm.observed(1.2, 1.2/(1/sqrt(10))), ylim=c(0,1.3))
mtext("all areas under the curve are numerically and visually the same",
    side=3, outer=TRUE)

par(mfrow=c(1,1))

## t distribution
mu.H0 <- 16
se.val <- .4
df.val <- 10
crit.val <- mu.H0 - qt(.95, df.val) * se.val
mu.alt <- 15
obs.mean <- 14.8
alt.t <- (mu.alt - crit.val) / se.val
norm.setup(xlim=c(12, 19), se=se.val, df.t=df.val)
norm.curve(critical.values=crit.val, se=se.val, df.t=df.val, mean=mu.alt,
    col='green', shade="left", axis.name="t1")
norm.curve(critical.values=crit.val, se=se.val, df.t=df.val, mean=mu.H0,
    col='gray', shade="right")
norm.observed(obs.mean, (obs.mean-mu.H0)/se.val)

## normal
norm.setup(xlim=c(12, 19), se=se.val)
norm.curve(critical.values=crit.val, se=se.val, mean=mu.alt,
    col='green', shade="left", axis.name="z1")
norm.curve(critical.values=crit.val, se=se.val, mean=mu.H0,
    col='gray', shade="right")
norm.observed(obs.mean, (obs.mean-mu.H0)/se.val)

## normal and t
norm.setup(xlim=c(12, 19), se=se.val, main="t(6) and normal")
norm.curve(critical.values=15.5, se=se.val, mean=16.3,
    col='gray', shade="right")
norm.curve(critical.values=15.5, se.val, df.t=6, mean=14.7,
    col='green', shade="left", axis.name="t1", second.axis.label.line=4)
norm.curve(critical.values=15.5, se=se.val, mean=16.3,
    col='gray', shade="none")
norm.setup(xlim=c(12, 19), se=se.val, main="t(6) and normal")
norm.curve(critical.values=15.5, se=se.val, mean=15.5,
    col='gray', shade="right")
norm.curve(critical.values=15.5, se=se.val, df.t=6, mean=15.5,
    col='green', shade="left", axis.name="t1", second.axis.label.line=4)
norm.curve(critical.values=15.5, se=se.val, mean=15.5,
    col='gray', shade="none")

par(old.par)

---

**NormalAndTplot**

*Specify plots to illustrate Normal and t Hypothesis Tests or Confidence Intervals.*

**Description**

Specify plots to illustrate Normal and t Hypothesis Tests or Confidence Intervals.
Usage

NormalAndTplot(mean0, ...)  # Default S3 method:
NormalAndTplot(mean0=0,  
    mean1=NA,  
    xbar=NA,  
    df=Inf, n=1,  
    sd=1,  
    xlim=c(-3, 3)*sd/sqrt(n) + range(c(mean0, mean1, xbar), na.rm=TRUE),  
    ylim,  
    alpha.right=.05, alpha.left=0,  
    float=TRUE, ntcolors="original",  
    digits=4, digits.axis=digits, digits.float=digits,  
    distribution.name=c("normal","z","t","binomial"),  
    type=c("hypothesis", "confidence"),  
    zaxis=FALSE, z1axis=FALSE,  
    cex.z=.5, cex.xbar=.5, cex.y=.5, cex.prob=.6, cex.top.axis=1,  
    cex.left.axis=1, cex.pb.axis=1,  
    cex.xlab=1, cex.ylab=1.5, cex.strip=1,  
    main=NA, xlab, ylab,  
    prob.labels=(type="hypothesis"),  
    xhalf.multiplier=1,  
    yhalf.multiplier=1,  
    cex.main=1,  
    key.axis.padding=4.5,  
    number.vars=1,  
    sub=NULL,  
    NTmethod="default",  
    power=FALSE,  
    beta=FALSE,  
    ...)  

## S3 method for class 'htest'
NormalAndTplot(mean0, type="hypothesis", xlim=NULL, mean1=NA, ...,  
    xbar, sd, df, n, alpha.left, alpha.right, ## ignored  
    distribution.name, sub ## these input arguments will be ignored  
)

Arguments

mean0 Null hypothesis $\mu_0$. When graphing a confidence interval, mean0 will be used for xbar should xbar itself have the value NA. For the htest method, mean0 is an "htest" object. See NTplot for more information.

mean1 Alternative hypothesis $\mu_1$.

xbar Observed $\bar{x}$.

ds Standard deviation in the data scale $\sigma$ for normal-, or $s$ for t-distribution.

df Degrees of freedom for t-distribution.

n Number of observations per group.
main, xlab, ylab, xlim, ylim, sub

Standard \texttt{xyplot} arguments. Default values are constructed if these arguments are missing. The input value \texttt{main=NA} forces a new constructed \texttt{main} instead of using the main coming in through the \texttt{htest} methods.

... Additional \texttt{xyplot} arguments.

\begin{description}
\item[number.vars] Number of variables. 1 for a one-sample test, 2 for two-sample tests and paired tests.
\item[alpha.left, alpha.right] For \texttt{type="hypothesis"}, the sum of these two numbers is the probability of the Type I Error $\alpha$. When both of these numbers are positive, there is a two-sided test. Note that it is not required that they be equal. If one of the numbers is 0, then it is a one-sided test. For \texttt{type="confidence"}, 1 minus the sum of these two numbers is the confidence level.
\item[float] Logical. If \texttt{TRUE}, then the probabilities $\alpha$, $\beta$, power, and $p$-values or the confidence value are displayed on the graph. If \texttt{FALSE}, these values are not displayed.
\item[ntcolors] Vector of colors used in the graph. The default value is "original" and two named alternatives are "stoplight" and "BW". The sets of colors associated with these three named sets are shown in a \texttt{dontrun} section of the examples. The user can enter any other color scheme by specifying a vector of ten named colors. The names are: \texttt{col.alpha, col.notalpha, col.beta, col.power, col.pvalue, col.pvaluetranslucent, col.critical, col.border, col.text, col.conf}.
\item[digits.axis, digits.float, digits] 
\texttt{digits.axis} is the number of significant digits for the top axis. \texttt{digits.float} is the number of significant digits for the floating probability values on the graph. \texttt{digits} is a convenience argument to set both \texttt{digits.axis} and \texttt{digits.float} at the same time. These number is passed to the \texttt{format} function.
\item[distribution.name] Name of distribution.
\item[type] "hypothesis" for a Hypothesis Test graph, or "confidence" for a Confidence Interval graph.
\item[zaxis, z1axis] Logical or list. Should the $z$-axis centered on $\mu_0$, or the $z_1$-axis centered on $\mu_1$, be displayed? The list version of the argument must have two components at and labels as specified in \texttt{panel.axis}.
\item[cex.z, cex.xbar, cex.y, cex.prob, cex.top.axis, cex.left.axis, cex.pb.axis, cex.xlab, cex.ylab, cex.strip] 
\texttt{cex.z} is the \texttt{cex} value for the $z$ and $z_1$ axes on the plot. \texttt{cex.prob} is the \texttt{cex} value for the floating probabilities on the graph. \texttt{cex.top.axis} is the \texttt{cex} value for the top axis values. \texttt{cex.main} is the \texttt{cex} value for the main title. \texttt{cex.xbar} and \texttt{cex.y} are the \texttt{cex} values for the horizontal and vertical axes of the plot. \texttt{cex.left.axis} and \texttt{cex.pb.axis} are the \texttt{cex} values for the power or beta (Type II error) values and the $\mu_1$ value in the power and beta plots. \texttt{cex.xlab}, \texttt{cex.ylab}, and \texttt{cex.strip} are the \texttt{cex} values for \texttt{xlab}, \texttt{ylab}, and \texttt{strip} labels.
\item[key.axis.padding] tuning constant to create additional room above the graph for a larger \texttt{cex.main} to fit.
NormalAndTplot

prob.labels logical. If TRUE label the floating probability values with their name, such as \( \alpha \). If FALSE, then don’t label them. The default is TRUE for type="hypothesis" and FALSE for type="confidence".

xhalf.multiplier, yhalf.multiplier
Numerical tuning constants to control the width and height of the floating probability values. Empirically, we need a smaller value for the shiny app then we need for direct writing onto a graphic device.

NTmethod Character string used when shiny=TRUE. It is normally calculated by the methods.
NTmethod tells shiny how to use or ignore the df and n sliders.
"htest" objects by default are interpreted as a single observation (n=1) of a \( t \)-statistic with df degrees of freedom. The slider will let the user change the df, but not the n.
"power.htest" objects are interpreted as a set of \( n \) observations per group and df is calculated as \( (n-1) \) for single-sample tests and as \( 2(n-1) \) for two-sample tests. The slider will let the user change \( n \) and will calculate the revised df.
For the normal approximation to the binomial (distribution.name="binomial"), only \( n \) is meaningful. The df is always ignored.
For the default situation of \( t \), determined by the initially specified sample size \( n > 1 \), the degrees of freedom is calculated as \( (n-1) \) for single-sample tests and as \( 2(n-1) \) for two-sample tests. The default \( z \), is initially specified by a sample size \( n = 1 \).
In all cases except the "binomial", the user can change the interpretation of the \( n \) and df sliders. The interpretation when both \( n \) and df are under user control is not always obvious.

power, beta Logical. If TRUE, then display that graph, else don’t display it. Passed forward to powerplot.

Details
The graphs produced by this single function cover most of the first semester introductory Statistics course. The htest method plots the results of the stats::t.test function.
NormalAndTplot is built on xyplot. Most of the arguments detailed in xyplot documentation work to control the appearance of the plot.

Value
"trellis" object.

Note
This function is built on lattice and latticeExtra. It supersedes the similar function normal.and.t.dist built on base graphics that is used in many displays in the book by Erich Neuwirth and me: \textit{R through Excel}, Springer (2009). \url{https://link.springer.com/book/10.1007/978-1-4419-0852-4}.
Many details, particularly the alternate color scheme and the concept of floating probability labels, grew out of discussions that Erich and I have had since the book was published. The method for "htest" objects incorporates ideas that Jay Kerns and I developed at the 2011 UseR! conference. This version incorporates some ideas suggested by Moritz Heene.
Author(s)
Richard M. Heiberger (rmh@temple.edu)

See Also
NTplot

Examples

```
NTplot(mean0=0, mean1=2, xbar=1.8, xlim=c(-3, 5))
NTplot(mean0=0, mean1=2, xbar=1.8, xlim=c(-3, 5), distribution.name="t", df=4)
NTplot(mean0=100, sd=12, mean1=113, xbar=105, xlim=c(92, 120), n=20, zaxis=TRUE, z1axis=TRUE)
NTplot(mean0=100, sd=12, xbar=105, xlim=c(92, 108), n=20, ntcors=\"stoplight\")
NTplot(xbar=95, sd=10, xlim=c(65, 125), type=\"confidence\", alpha.left=.025, alpha.right=.025)
```

```
x <- rnorm(12, mean=.78)
x.t <- t.test(x)
NTplot(x.t)
NTplot(x.t, type=\"confidence\")
x.tg <- t.test(x, alternative=\"greater\")
NTplot(x.tg)
```

```
y <- rnorm(12, mean=-.05)
xy.t <- t.test(x, y)
NTplot(xy.t)
NTplot(xy.t, type=\"confidence\")
```

```R
## Not run:
if (interactive())
NTplot(shiny=TRUE)  ## with any other arguments for initialization of the shiny app.
## End(Not run)

## Not run:
## The partially transparent colors are:
black127="#0000007F"  ## HH:::ColorWithAlpha("black")
green127="#00FF007F"  ## HH:::ColorWithAlpha("green")
blue127 ="#0000FF7F"  ## HH:::ColorWithAlpha("blue")
```

```
## this is the default set of colors that are assigned when
## ntcors=\"original\" or when ntcors is not specified

  c(col.alpha = "blue",
    col.notalpha = "lightblue",
    col.beta = "red",
    col.power = "pink",
    col.pvalue = "green",
    col.pvaluetranslucent = green127,
```
col.critical = "gray50",
col.border = black127,
col.text = "black",
col.conf = "lightgreen")

NTplot(  
NTplot(mean1 = 2,  
NTplot(  xbar=1)  
NTplot(mean1 = 2, xbar=1)  
NTplot(type="confidence")

## this is the set of colors that are assigned when ntcolors="stoplight"
c(col.alpha = "red",  
c.col.notalpha = "honeydew2",  
c.col.beta = "orange",  
c.col.power = "pink",  
c.col.pvalue = "blue",  
c.col.pvaluetranslucent = blue127,  
c.col.critical = "gray50",  
c.col.border = black127,  
c.col.text = "black",  
c.col.conf = "lightgreen")

NTplot(   ntcolors="stoplight")  
NTplot(mean1 = 2,   ntcolors="stoplight")  
NTplot(    xbar=1, ntcolors="stoplight")  
NTplot(mean1 = 2, xbar=1, ntcolors="stoplight")  
NTplot(type="confidence", ntcolors="stoplight")

## this is the set of colors that are assigned when ntcolors="BW"
c(col.alpha = "gray35",  
c.col.notalpha = "gray85",  
c.col.beta = "gray15",  
c.col.power = "gray40",  
c.col.pvalue = "gray50",  
c.col.pvaluetranslucent = HH:::ColorWithAlpha("gray65"),  
c.col.critical = "gray15",  
c.col.border = "gray75",  
c.col.text = "black",  
c.col.conf = "gray45")

NTplot(   ntcolors="BW")  
NTplot(mean1 = 2,   ntcolors="BW")  
NTplot(    xbar=1, ntcolors="BW")  
NTplot(mean1 = 2, xbar=1, ntcolors="BW")  
NTplot(type="confidence", ntcolors="BW")

## End(Not run)

## Not run:
## mean1 and xbar
NTplot(mean0=0, mean1=2, xbar=1.8, xlim=c(-3, 5))
NTplot(mean0=0, mean1=-2, xbar=-1.8, xlim=c(-5, 3),
       alpha.left=.05, alpha.right=0)
NTplot(mean0=0, mean1=2, xbar=2.1, xlim=c(-3, 5),
       alpha.left=.025, alpha.right=.025)
NTplot(mean0=0, mean1=-2, xbar=-2.1, xlim=c(-5, 3),
       alpha.left=.025, alpha.right=.025)

## mean1
NTplot(mean0=0, mean1=2, xbar=NA, xlim=c(-3, 5))
NTplot(mean0=0, mean1=-2, xbar=NA, xlim=c(-5, 3),
       alpha.left=.05, alpha.right=0)
NTplot(mean0=0, mean1=2, xbar=NA, xlim=c(-3, 5),
       alpha.left=.025, alpha.right=.025)
NTplot(mean0=0, mean1=-2, xbar=NA, xlim=c(-5, 3),
       alpha.left=.025, alpha.right=.025)

## xbar
NTplot(mean0=0, mean1=NA, xbar=1.8, xlim=c(-3, 5))
NTplot(mean0=0, mean1=NA, xbar=-1.8, xlim=c(-5, 3),
       alpha.left=.05, alpha.right=0)
NTplot(mean0=0, mean1=NA, xbar=2.1, xlim=c(-3, 5),
       alpha.left=.025, alpha.right=.025)
NTplot(mean0=0, mean1=NA, xbar=-2.1, xlim=c(-5, 3),
       alpha.left=.025, alpha.right=.025)

## t distribution
## mean1 and xbar
NTplot(mean0=0, mean1=2, xbar=1.8, xlim=c(-3, 5),
       distribution.name="t", df=4)
NTplot(mean0=0, mean1=-2, xbar=-1.8, xlim=c(-5, 3),
       alpha.left=.05, alpha.right=0, distribution.name="t", df=4)
NTplot(mean0=0, mean1=2, xbar=2.1, xlim=c(-3, 5),
       alpha.left=.025, alpha.right=.025, distribution.name="t", df=4)
NTplot(mean0=0, mean1=-2, xbar=-2.1, xlim=c(-5, 3),
       alpha.left=.025, alpha.right=.025, distribution.name="t", df=4)

## mean1
NTplot(mean0=0, mean1=2, xbar=NA, xlim=c(-3, 5),
       distribution.name="t", df=4)
NTplot(mean0=0, mean1=-2, xbar=NA, xlim=c(-5, 3),
       alpha.left=.05, alpha.right=0, distribution.name="t", df=4)
NTplot(mean0=0, mean1=2, xbar=NA, xlim=c(-3, 5),
       alpha.left=.025, alpha.right=.025, distribution.name="t", df=4)
NTplot(mean0=0, mean1=-2, xbar=NA, xlim=c(-5, 3),
       alpha.left=.025, alpha.right=.025, distribution.name="t", df=4)

## xbar
NTplot(mean0=0, mean1=NA, xbar=1.8, xlim=c(-3, 5),
       distribution.name="t", df=4)
NTplot(mean0=0, mean1=NA, xbar=-1.8, xlim=c(-5, 3),
       alpha.left=.05, alpha.right=0, distribution.name="t", df=4)
NTplot(mean0=0, mean1=NA, xbar=2.1, xlim=c(-3, 5),
alpha.left=.025, alpha.right=.025, distribution.name="t", df=4)
NTplot(mean0=0, mean1=NA, xbar=-2.1, xlim=c(-5, 3),
alpha.left=.025, alpha.right=.025, distribution.name="t", df=4)

## confidence intervals
NTplot(mean0=0, xlim=c(-3, 4), type="confidence")
NTplot(xbar=01, xlim=c(-3, 4), type="confidence")
NTplot(mean0=0, xlim=c(-4, 3), type="confidence",
alpha.left=.05, alpha.right=0)
NTplot(mean0=0, xlim=c(-3, 3), type="confidence",
alpha.left=.025, alpha.right=.025)
NTplot(mean0=95, sd=10, xlim=c(65, 125), type="confidence",
alpha.left=.025, alpha.right=.025)
NTplot(mean0=95, sd=10, xlim=c(65, 125), type="confidence",
alpha.left=.025, alpha.right=.025,
distribution="t", df=10)

## End(Not run)

---

**NormalAndTPower**

*Construct a power graph based on the NTplot.*

**Description**

Construct a power graph based on the NTplot. The exported function `powerplot` calls `NormalAndTPower` to construct a power curve or beta curve (operating characteristic curve) (or both) from its argument and catenates it to the original graph. The unexported function `NormalAndTPower` does the construction.

**Usage**

```r
powerplot(nt, ...)
```

## S3 method for class 'NormalAndTplot'
```
powerplot(nt, power=TRUE, beta=FALSE, ...,
        hh=if (power && beta) c(6,2,2) else c(6,2))
```

NormalAndTPower(nt,
which=c("power","beta"),
digits=4,
digits.top.axis=digits, digits.left=digits,
col.power=attr(nt, "color")["col.power"],
col.beta=attr(nt, "color")["col.beta"],
cex.pb.axis=1, cex.left.axis=1, cex.xbar=1,
lwd.reference=4, lwd.line=2,
main=which, ...)

---
Arguments

nt
power, beta
which

Arguments
For the generic powerplot, an object. For the NormalAndTplot method, a "NormalAndTplot" object from NTplot.

Logical. If TRUE, then display that graph, else don’t display it. Used by powerplot.

Which graph is to be displayed? "power" for the power curve, or "beta" for the operating characteristic curve. Used by NormalAndTPower.

Additional arguments passed on to methods.

The h argument for resizePanels.

digits.top.axis, digits.left, digits, cex.pb.axis, cex.left.axis, cex.xbar

digits.top.axis is the number of significant digits for the top axis. digits.left is the number of significant digits for the observed power or beta on the left axis. digits is a convenience argument to set both digits.axis and digits.left at the same time. These number is passed to the format function. cex.top.axis is the cex value for the top axis values. cex.left.axis is the cex value for the observed power or beta on the left axis. cex.xbar is the cex value for the horizontal axis.

col.power, col.beta

Colors used for the crosshairs on the power and beta panels. The default values are the colors used for the power and beta regions of the NTplot panel.

lwd.reference, lwd.line

lwd values for the power or beta panel. lwd.line is used for the power curve or beta curve. lwd.reference is used for the crosshairs.

main

Main title for graph.

Value

"trellis" object.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

Examples

nt <- NTplot(mean0=2, mean1=4, sd=3, n=20, xlim=c(-.1, 6.1), xbar=3.5)
powerplot(nt)

## Not run:
tt <- NTplot(mean0=2, mean1=4, sd=3, n=20, xlim=c(-.1, 6.1), xbar=3.5, df=4, distribution.name="t")
powerplot(tt)

ntc <- NTplot(xbar=2, sd=3, n=20, xlim=c(-.1, 4.1), type="confidence",
        alpha.left=.025, alpha.right=.025)
ntc
try(powerplot(ntc))

## End(Not run)
normalApproxBinomial  
Plots to illustrate Normal Approximation to the Binomial—hypothesis tests or confidence intervals.

Description

Plots to illustrate Normal Approximation to the Binomial—hypothesis tests or confidence intervals.

Usage

normalApproxBinomial(p0= if (number.vars==1) .5 else 0, p1=NA, p2=NA, p.hat=if (number.vars==1) .75 else 0, n=1, xlim=if (number.vars==1) c(0,1) else c(-1,1), ylim=c(0, 5), type=c("hypothesis","confidence"), alpha.left=if (type=="hypothesis") 0 else .025, alpha.right=if (type=="hypothesis") .05 else .025, xlab=if (number.vars==1)
  "w = p = population proportion"
  else
  "w = p[1] - p[2] :: population proportions", ..., number.vars=if (!is.na(p1) && !is.na(p2)) 2 else 1)

Arguments

p0  Null hypothesis value of p.
p1  Alternate hypothesis value of p for one-sample cases. Second sample value of p for two-sample cases.
p2  Second sample value of p.
p.hat  Observed value of p.
n  Number of observations (for example, number of coins tossed).
xlim, ylim, xlab  Standard xyplot arguments...
type  "hypothesis" for a Hypothesis Test graph, or "confidence" for a Confidence Interval graph.
..., alpha.left, alpha.right  Additional arguments forwarded to NTplot.
number.vars  Number of variables. 1 for a one-sample test, 2 for two-sample tests and paired tests.

Details

This is a wrapper function for the plots in NTplot.
**npar.arma**

Count the number of parameters in an ARIMA model specification.

**Description**

Count the number of parameters in an ARIMA model specification. When `arima==FALSE`, just the AR and MA parameters are counted. When `arima==TRUE`, then the number of difference parameters are also included.

**Usage**

```r
npar.arma(x, arima=FALSE)
npar.sarma(model, arima=FALSE)
npar.rarma(arma, arima=FALSE)
```

**Arguments**

- `x` : An "arima" object in S-Plus or a "Arima" object in R.
- `model` : A model specification in the S-Plus style.
- `arma` : A arma specification in the R style.
- `arima` : Logical. TRUE is number of differencings is to be counted.

**Value**

"trellis" object.

**Author(s)**

Richard M. Heiberger (rmh@temple.edu)

**Examples**

```r
# These are interactive and won't work in R CMD check
if (interactive())
  NTplot(distribution.name="binomial", n=20, ylim=c(0, 4.2), p1=.8, shiny=TRUE)
if (interactive())
  NTplot(p0=.4, p.hat=.65, p1=.7, distribution.name="binomial", n=15, shiny=TRUE)
if (interactive())
  NTplot(p.hat=.65, distribution.name="binomial", n=15, type="confidence", shiny=TRUE)
```

## End(Not run)
Value

A scalar number giving the count.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

Examples

```r
co2.arima <-
  if.R(s=
    arima.mle(co2, list(list(order=c(0,1,1)),
                list(order=c(0,1,1), period=12)))
    ,r=
    arima(co2,
          order=c(0,1,1),
          seasonal=list(order=c(0,1,1), period=12))
  )

npar.arma(co2.arima)
npar.arma(co2.arima, arima=TRUE)
npar.sarma(list(list(order=c(0,1,1)),
               list(order=c(0,1,1), period=12)))
npar.sarma(list(list(order=c(0,1,1)),
               list(order=c(0,1,1), period=12),
               arima=TRUE)

if.R(s=(),
    r=npar.rarma(co2.arima$arma)
  )
if.R(s=(),
    r=npar.rarma(co2.arima$arma, arima=TRUE)
  )
```

Description

Specify plots to illustrate Normal and t Hypothesis Tests or Confidence Intervals, including normal approximation to the binomial.
NTplot

Usage

NTplot(mean0, ...)  
## Default S3 method:  
NTplot(mean0=0, ..., shiny=FALSE,  
distribution.name = c("normal","z","t","binomial"))  
## S3 method for class 'htest'  
NTplot(mean0, ..., shiny=FALSE, NTmethod="htest")  
## S3 method for class 'power.htest'  
NTplot(mean0, ..., shiny=FALSE, xbar=NA, ## these input values are used mean1, n, df, sd, distribution.name, sub, ## these input values ignored alpha.left, alpha.right, number.vars) ## these input values ignored  
## NTplot(NTplot(htest.object), n=20) ## allows override of arguments  
## S3 method for class 'NormalAndTplot'  
NTplot(mean0, ..., shiny=FALSE)

Arguments

mean0 For the default method, mean0 is either missing or a numeric argument for the mean under the null hypothesis. For the htest method, mean0 is an htest object from the t.test or the z.test function. For the NormalAndTplot method mean0 is a "NormalAndTplot" object from a previous use of the NTplot function. For the power.htest method, mean0 is a power.htest object from the power.t.test function.

xbar See NormalAndTplot.

... Other arguments, selected from the options for the default method NormalAndTplot. Three named color schemes are available: the default ntcards="original", ntcards="stoplight", and ntcards="BW". Their definitions, along with information on specifying other color schemes, are shown in NormalAndTplot.

shiny Logical. If TRUE, a shiny app is started to provide an interactive graphics device in a web-browser. If FALSE, a plot is drawn on the current graphics device. For short browser windows (height < 800 pixels), you may adjust the pixel height of the plot in the last user input field on the Fonts tab.

htest logical. TRUE for "htest" objects.

mean1, n, df, sd, sub, alpha.left, alpha.right, number.vars These variables are ignored here. They are captured so they won’t interfere with similarly named variables that are generated in the power.htest method.

distribution.name Ignored by htest and power.htest methods. Otherwise passed on to the next method.

NTmethod Character string used when shiny=TRUE. It is normally calculated by the methods. NTmethod tells shiny how to use or ignore the df and n sliders. See the extended discussion in NormalAndTplot.

Details

The graphs produced by this single function cover most of the first semester introductory Statistics course. All options of the t.test, power.t.test, and z.test are accepted and displayed.
NTplot is built on *xyplot*. Most of the arguments detailed in *xyplot* documentation work to control the appearance of the plot.

The shiny app (called when the argument shiny=TRUE) provides animated sliders for the means, standard deviation, xlims, significance levels, df, and n. The df and n are rounded to integers for the sliders (relevant for *htest* and *power.htest* objects). Checkboxes and Radio buttons are available for various display options.

When you have a graph on the shiny window that you wish to keep, click on the "Display Options" tab, and then on the "Display Call" radio button. The main shiny window will show an R command which will reproduce the current plot. Pick it up with the mouse and drop it into an R console window.

To get out of the shiny window and return to an interactive R console, move the cursor back to the console window and interrupt the shiny call, usually by entering Ctrl-C or ESC.

**Value**

"trellis" object. The object can be plotted or fed back into the NTplot function with argument shiny=TRUE to allow interactive graphical investigation of the hypothesis test or confidence interval. The attributes of the object\ NTobj <- NTplot()\ attr(NTobj, "scales") and attr(NTobj, "prob") make the data values and probability values accessible for further R computations. The "call" attribute cat(attr(NT.object, "call"), "\n") displays a statement that can be copied back into R to reproduce the graph. The cat() is needed to unescape embedded quotes. The "call.list" attribute attr(NT.object, "call.list") is a list that can be used with do.call to reproduce the graph. do.call(NTplot, attr(NT.object, "call.list")). This is usually not needed by the user because the simpler statement NTplot(NT.object) does it for you.

**Note**

This function is built on lattice and latticeExtra. It supersedes the similar function normal.and.t.dist built on base graphics that is used in many displays in the book by Erich Neuwirth and me: *R through Excel*, Springer (2009). [https://link.springer.com/book/10.1007/978-1-4419-0052-4](https://link.springer.com/book/10.1007/978-1-4419-0052-4). Many details, particularly the alternate color scheme and the concept of floating probability labels, grew out of discussions that Erich and I have had since the book was published. It incorporates ideas that Jay Kerns and I developed at the 2011 UseR! conference. This version incorporates some ideas suggested by Moritz Heene.

**Author(s)**

Richard M. Heiberger (rmh@temple.edu)

**See Also**

NormalAndTplot, print.NormalAndTplot.

**Examples**

```r
x1 <- rnorm(12)
x2 <- rnorm(12, mean=.5)

NT.object <- NTplot(mean0=0, mean1=1)
```
objip

Loop objects() through all attached directories (items in the search() list) looking for a regular expression pattern.

Description

Loop objects() through all attached directories looking for a regular expression pattern.

Usage

objip(pattern, where = search(), all.names=FALSE, mode="any", class,
    ls.function=if (mode != "any" || !missing(class)) "ls.str" else "ls")

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.

all.names In R, a logical that is passed to the ls function. If ‘TRUE’, all object names are returned. If ‘FALSE’, names which begin with a ’.’ are omitted.

mode, class See ls.str and mode for storage mode of an object. See class for object classes.

ls.function Either ls or ls.str. If either mode or class is specified then the default is ls.str. If neither is specified then the default is ls.

Value
A list of 0 or more character vectors. Each character vector has the name of one of the items in the search() list. Each character vector contains the names of the objects in the specified environment which match the pattern. If there are no matching names in an environment, then the corresponding character vector is removed from the result.

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

See Also
ls.

Examples
objip("qq")
objip("^qq")
objip("qq$")
## Not run:
## R only
objip("rowSums", all.names=TRUE)

## list all objects in the second and third attached packages
search()
objip()[2:3]

## End(Not run)

OddsRatio Calculate or plot the odds ratio for a 2x2 table of counts.

Description
Calculate or plot the odds ratio for a 2x2 table of counts. The plot shows the confidence intervals on the probability of row2 for fixed odds ratio and specified probability for row1.
OddsRatio

Usage

OddsRatio(x, alpha = 0.05)

plotOddsRatio(x,
    ylab="prob(col1 | row1)",
    xlab="prob(col1 | row2),
    alpha=c(.50, .05),
    col=trellis.par.get("superpose.line")$col,
    lwd=trellis.par.get("superpose.line")$lwd,
    lwd.reference=1,
    ...
)

plotOddsRatio.base(x,
    ylab = "prob(col1 | row1)", xlab = "prob(col1 | row2),
    alpha = c(0.05, 0.5),
    legend.x=1.05,
    oma=c(0,0,0,5), ...
)

Arguments

x 2 x 2 table of counts
alpha Significance levels of test. OddsRatio requires a single number in the range [0,1]. plotOddsRatio accepts more than one number on the range [0,1] and draws confidence lines at each value.
xlab, ylab x- and y-labels for the plot Sensible defaults are generated.
col, lwd Colors and linewidths to be used in the graph.
lwd.reference lineweight for reference line.
... other arguments, currently ignored.
legend.x x position of left-hand side of legend.
oma outer margin par()$oma, needed to make room for legend.

Value

plotOddsRatio returns a lattice object.
The older plotOddsRatio.base draws a plot with base graphics and invisibly returns the same list as OddsRatio returns for the first value of alpha.
OddsRatio returns the list:

p1, p2 proportion of each row total observed in the first column.
omega1, omega2 odds for each row, p/(1-p)
psihat odds ratio, omega2/omega1
s.ln.psihat standard deviation of ln(psihat)
ci.ln.psihat confidence interval for ln(psihat) using normal approximation
ci.psihat confidence interval for psihat calculated ase p(ci.ln.psihat)
prob1 = seq(0,1,.05), set of p1 values for plotting.

odds1 = p1/(1-p1)

odds2 = odds for the second row needed to retain \( \psi \) with the specified \( \text{odds1} \), calculated as \( \text{odds1} \* \psi \).

ci.odds2 = confidence interval for \( \text{odds2} \)

prob2 = \( \frac{\text{odds2}}{\text{odds2}+1} \)

ci.prob2 = \( \frac{\text{ci.odds2} \* (\text{ci.odds2}+1)}{\text{ci.odds2}+1} \)

**Author(s)**

Richard M. Heiberger <rmh@temple.edu>

**References**


**Examples**

```r
data(glasses)

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

## return the 95% CI information
OddsRatio(glasses)

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

**OneWayVarPlot**

*Displays a three-panel bwplot of the data by group, of the group means, and of the entire dataset. This is an approximate visualization of the Mean Square lines from the ANOVA table for a one-way ANOVA model.*

**Description**

Displays a three-panel bwplot of the data by group, of the group means, and of the entire dataset. This is an approximate visualization of the Mean Square lines from the ANOVA table for a one-way ANOVA model. The groups are centered using medians by default. Means, and anything else, is an option.
orthog.complete

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

Description

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

Usage

orthog.complete(x, normalize=TRUE, abs2.rows=1:nrow(x),
                          Int=TRUE, drop.Int=Int)

orthog.construct(y, x, x.rows, normalize=FALSE)
orthog.complete

Arguments

x  For orthog.complete, an n-row matrix of one or more columns.  For orthog.construct, a set of contrasts for a factor.
y  matrix of coefficients specifying the linear combinations estimated. This will usually be the lmat from an S-Plus "multicomp" object or the linfct component from an R "glht" object.
normalize, abs2.rows, x.rows
  The default normalizes the sum of squares of the rows in abs2.rows or x.rows to 1. The optional value normalize="abs2" scales the rows in abs2.rows or x.rows to make the sum of all positive value equal 1 and the sum of all negative values equal −1. Together, the sum of the absolute values of the specified rows in each column is 2.

Int
  logical. Default TRUE means make all columns orthogonal to the constant column (Intercept in regression terminology). The alternative is to use only the columns in the input matrix x.
drop.Int
  logical. The default is to drop the constant column and to keep all columns when the constant is not automatically generated.

Details

This function is based on qr.Q. The input matrix x has n rows and an arbitrary non-zero number of columns. The result is an n by n orthogonal matrix. By default the first column of the result is constant and is not returned. The second column of the result is orthogonal to the first result column. Together the first two result columns span the space of the constant column and the first input column. The third result column is orthogonal to the first two result columns and the the three result columns together span the space of the constant column and the first two inout columns. Similarly for the remaining result columns. Result columns beyond the number of input columns are constructed as an arbitrary orthogonal completion.

If the input columns are orthogonal to each other and to the constant column, then the result columns are rescaled versions of the input columns.

Optionally (drop.Int=FALSE), the constant column can be returned.

By default the columns are scaled to have sum of squares equal 1. If normalize="abs2", they are scaled to make the sum of all positive value equal 1 and the sum of all negative values equal −1. Together, the sum of the absolute values of each column is 2.

If the input is a set of columns from a contrast matrix for a design that has multiple terms, the abs2.rows argument is used to specify which rows are to be included in the normalization. These will normally be rows associated with one of the main effects.

Value

Matrix of orthogonal columns.

Author(s)

Richard M. Heiberger <rmh@temple.edu>
orthog.complete

References


See Also

MMC

Examples

```r
zapsmall(
  orthog.complete(cbind("4-12"=c(-1,-1, 0, 2),
                      "1-2" =c( 1,-1, 0, 0)))
)

zapsmall(
  orthog.complete(cbind("4-12"=c(-1,-1, 0, 2),
                      "1-2" =c( 1,-1, 0, 0)),
                 drop.Int=FALSE)
)

zapsmall(
  orthog.complete(cbind("4-12"=c(-1,-1, 0, 2),
                      "1-2" =c( 1,-1, 0, 0)),
                 normalize="abs2")
)

## used in MMC plots
tmp <- data.frame(y=rnorm(12),
                  a=factor(c("u","u","u","u",
                           "v","v","v","v",
                           "w","w","w","w")))
tmp.aov <- aov(y ~ a, data=tmp)
lmat <- if.R(  
    s=multicomp(tmp.aov, focus="a")$lmat,
    r={lmat.reduced <- t(glht(tmp.aov, linfct=mcp(a="Tukey"))$linfct)
        rbind(lmat.reduced, AU=-apply(lmat.reduced[-1,,], 2, sum))}
)  
zapsmall(lmat)

lmat.complete <- orthog.complete(lmat, abs2.rows=-1,
                                  normalize="abs2",
                                  drop.Int=FALSE)[,1:3]

zapsmall(lmat.complete)
if.R(r=zapsmall(lmat.complete[-4,]),
     s=())
```
panel.acf

*Panel functions for tsdiagplot.*

Description

Panel functions for `tsdiagplot`.

Usage

```r
panel.acf(..., n.used)
panel.std.resid(...)
panel.gof(...)
```

Arguments

- `...`: standard arguments to panel functions.
- `n.used`: number of lags. This number is needed to calculate the confidence interval which is \( \frac{2}{\sqrt{n\text{used}}} \).

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

`tsdiagplot`

panel.axis.right

*Right-justify right-axis tick labels.*

Description

`panel.axis.right` is almost identical to `panel.axis`. `axis.RightAdjustRight` is almost identical to `axis.default`. The only difference is that these functions right-justify right-axis tick labels.

Usage

```r
panel.axis.right(side = c("bottom", "left", "top", "right"),
                 at = pretty(scale.range),
                 labels = TRUE, draw.labels = TRUE,
                 check.overlap = FALSE, outside = FALSE, ticks = TRUE,
                 half = !outside,
                 which.half = switch(side, bottom = "lower",
                                     left = "upper", top = "upper",
                                     right = "lower"),
```
tck = as.numeric(ticks),
or = if (is.logical(labels)) 0 else c(90, 0),
text.col = axis.text$col, text.alpha = axis.text$alpha,
text.cex = axis.text$cex, text.font = axis.text$font,
text.fontfamily = axis.text$fontfamily,
text.fontface = axis.text$fontface,
text.lineheight = axis.text$lineheight,
line.col = axis.line$col, line.lty = axis.line$lty,
line.lwd = axis.line$lwd, line.alpha = axis.line$alpha)

axis.RightAdjustRight(side = c("top", "bottom", "left", "right"),
                      scales, components, as.table,
                      labels = c("default", "yes", "no"),
ticks = c("default", "yes", "no"),
...,
prefix = lattice.lattice.getStatus("current.prefix"))

Arguments

side, at, labels, draw.labels, check.overlap, outside, ticks, half, which.half
  See panel.axis and axis.default
tck, rot, text.col, text.alpha, text.cex, text.font, text.fontfamily
  See panel.axis and axis.default
text.fontface, text.lineheight, line.col, line.lty, line.lwd, line.alpha
  See panel.axis and axis.default
scales, components, as.table, ..., prefix
  See axis.default

Author(s)

Deepayan Sarkar Deepayan.Sarkar@R-project.org wrote panel.axis and axis.default. David
Winsemius wrote the modification panel.axis.right. Richard Heiberger rmh@temple.edu wrote
the modification axis.RightAdjustRight. Richard Heiberger is maintaining this distribution of
both functions.

See Also

panel.axis

Panel functions for bwplot.

Description

Panel function for bwplot that give the user control over the placement of the boxes. When used
with a positioned factor, the boxes are placed according to the position associated with the factor.
Usage

panel.bwplot.intermediate.hh(x, y, horizontal = TRUE, pch, col, lwd, ...) 

Arguments

x, y, pch, col, lwd, horizontal  
see xyplot and panel.bwplot.

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

panel.xyplot, xyplot, interaction2wt, position

Examples

```r
## see examples at
## Not run:
demo("bwplot.examples", package="HH")
## End(Not run)

panel.bwplot.superpose

Panel function for bwplot that displays an entire box in the colors coded by groups.

Description

Panel function for bwplot that displays an entire box (central dot, box, umbrella, outliers) in the same color, coded by the groups argument. The function is based on panel.superpose.
panel.bwplot.superpose

Usage

```r
panel.bwplot.superpose(x, y, ..., 
  groups=groups,
  col=rep(trellis.par.get("superpose.symbol")$col,
        length=length(groups)),
  pch=trellis.par.get("box.dot")$pch,
  panel.groups=panel.bwplot.groups)
```

```r
panel.bwplot.groups(..., col, pch, fill, fill.alpha=NULL, group.number)
```

Arguments

- **x, y**: Standard arguments to a `lattice` panel function. When `x` has class `positioned` (see `position`), the position will be forwarded by `panel.bwplot.superpose` to `panel.bwplot.groups`.
- **...**: Additional `lattice` arguments.
- **groups**: Factor to be used for color coding entire boxes: central dot, rectangle, umbrella, and outlier symbol.
- **col**: Colors to be assigned to the levels of the group. The default colors are taken from `trellis.par.get("superpose.symbol")$col`.
- **pch**: Standard `lattice` arguments. The `pch` describes the central dot. The outlier dots are specified in the `plot.symbol` component of `trellis.par.get`.
- **fill, fill.alpha**: These are related to the similarly named arguments in `panel.bwplot`. The `fill` argument is ignored. It is there to capture the automatically generated `fill` argument. The default `NULL` value of `fill.alpha` implies that there is no background color for the boxes. The user can set `fill.alpha` to a number between 0 and 1. The boxes will be shaded in a lighter version of their color as implied by the `groups` argument. The value 0 gives a transparent fill, and the value one makes the box the full opaque color.
- **panel.groups, group.number**: See `panel.superpose` for details.

Details

`panel.bwplot.superpose` is the user-level function. `panel.bwplot.groups` is the `panel.groups` function called by `panel.superpose`.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`position`, `panel.bwplot.intermediate.hh`, `panel.superpose`
Examples

tmp <- data.frame(Response=rnorm(20), Group=factor(rep(LETTERS[1:3], c(5,7,8))))

bwplot(Group ~ Response, data=tmp, 
main="Default panel.bwplot, groups ignored", groups=Group)

bwplot(Group ~ Response, data=tmp, 
main="panel.bwplot.superpose", groups=Group, panel=panel.bwplot.superpose)

bwplot(Group ~ Response, data=tmp, 
main="panel.bwplot.superpose with fill specified", fill.alpha=.33, 
groups=Group, panel=panel.bwplot.superpose)

bwplot(Group ~ Response, data=tmp, 
main="panel.bwplot.superpose, with color specified", 
col=c("forestgreen","blue", "brown"), 
groups=Group, panel=panel.bwplot.superpose)

Test <- data.frame(id=rep(letters, each=4), 
Week=rep(c(0,1,3,6), 26), 
Treatment=rep(factor(c("A","B"), levels=c("A","B")), each=52), 
y=rep(1:4, 52) + rep(4:5, each=52) + rnorm(104), 
stringsAsFactors=FALSE)

Test$WeekTrt <- with(Test, interaction(Week, Treatment))
position(Test$Week) <- unique(Test$Week)
position(Test$WeekTrt) <- as.vector(outer(position(Test$Week), c(-.2, .2), "+"))

tapply(Test$y, Test[c("Week", "Treatment")], median)

bwplot( y ~ WeekTrt, groups = Treatment, data = Test, 
main="default panel.bwplot, groups ignored")

bwplot( y ~ WeekTrt, groups = Treatment, data = Test, 
panel=panel.bwplot.superpose, 
scales=list(x=list(limits=c(-1, 7))), 
main="Minimal use of panel.bwplot.superpose")

bwplot( y ~ WeekTrt, groups = Treatment, data = Test, 
panel=panel.bwplot.superpose, 
scales=list(x=list(limits=c(-1, 7)), at=position(Test$Week))), 
box.width=.3, 
xlab="Week", 
pch=c(17, 16), 
key=list(col=trellis.par.get()$superpose.symbol$col[1:2], 
border=TRUE, title="Treatment", cex.title=1, columns=2, 
text=list(levels(Test$Treatment)), 
points=list(pch=c(17, 16)), 
par.settings=list(plot.symbol=list(pch=c(17, 16), cex=.5)), 

tests {param}for {param}
main="panel.bwplot.superpose with additional annotations"

bwplot(y ~ WeekTrt, groups = Treatment, data = Test,
       panel=panel.bwplot.superpose,
       scales=list(x=list(limits=c(-1, 7), at=position(Test$Week))),
       box.width=.3,
       xlab="Week",
       pch=c(17, 16),
       key=list(col=trellis.par.get()$superpose.symbol$col[1:2],
                border=TRUE, title="Treatment", cex.title=1, columns=2,
                text=list(levels(Test$Treatment)),
                points=list(pch=c(17, 16)),
                par.settings=list(plot.symbol=list(pch=c(17, 16), cex=.5)),
                main="panel.bwplot.superpose with fill and more complex panel.groups",
                panel.groups = function(...) {
                    panel.stripplot(...)
                    panel.bwplot.groups(...)
                },
                fill.alpha=.33,
                jitter.data = TRUE
)

panel.bwplott

Extension to S-Plus trellis to allow transposed plots.

Description

Extension to S-Plus trellis to allow transposed plots. All x - and y-components of the trellis object are interchanged. This function is not needed in R as lattice has a horizontal argument in its definitions.

Usage

panel.bwplott(x, y, box.ratio = 1,
               font = box.dot$font, pch = box.dot$pch, cex = box.dot$cex,
               col = box.dot$col, ..., transpose=FALSE)

Arguments

x, y, box.ratio, font, pch, cex, col, ...

See
panel.bwplot.

transpose logical. If FALSE, the plot is printed in the default orientation. If TRUE, the x- and y-components of the trellis object are interchanged. This has the effect, for example, of displaying vertical boxplots instead of the default horizontal boxplots.
Value

The function is used for its side effect of drawing boxplots in a trellis panel.

Note

This function is not needed in R. If it is used and

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

xyplot.

Description

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

Usage

panel.cartesian(x, y,
              x.label=unique(panel.labels[,"x"]),
y.label=unique(panel.labels[,"y"]),
group.label.side="",
axis3.line=1,
xg.label, yg.label, g.cex=.7,
rescale=list(x=TRUE,y=TRUE), ...,
browser.on=FALSE)

Arguments

x, y x and y as for any other panel function
x.label labels for the columns of the scatterplot matrix
y.label labels for the rows of the scatterplot matrix
axis3.line The x.label doesn’t always show up in the right place. This allows the user to adjust 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

data(rent)  ## Weisberg's file alr162

rent.lm <- lm(rnt.alf ~ rnt.till + cow.dens + lime, data=rent)
rent$resid.rent <- resid(rent.lm)

xysplom(resid.rent ~ rnt.till + cow.dens | lime, data=rent,
  layout=c(2,2))

xysplom(resid.rent ~ rnt.till + cow.dens | lime, data=rent,
  layout=c(2,2),
  xlab="", ylab="",
  x.label="", y.label="",
  group.label.side="",
  par.strip.text=list(cex=1.2),
  panel=panel.cartesian,
  axis3.line=2.4,
  scales=list(
    relation="same",
    alternating=FALSE, labels=FALSE, ticks=FALSE),
  between=list(x=1, y=3))

xysplom(resid.rent ~ rnt.till + cow.dens | lime, data=rent,
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

description

Confidence interval panel for MMC tiebreaker plots, or confidence interval plot.

Usage

panel.confintMMC(x, y, subscripts, ..., col, lwd, lty, lower, upper, contrast.name, right.text.cex = 0.8, contrast.height=FALSE)

Arguments

x
means

y
When called frommmcisomeans, the heights associated with the contrasts. When called frommmcmatch, integers from one to the number of means.

subscripts
Index into the contrast.names.

...
Additional arguments are ignored.

col, lty, lwd
Standard lattice arguments.
panel.dotplot.tb

lower
upper
contrast.name
right.text.cex
contrast.height

Vector of lower bounds for the intervals.
Vector of upper bounds for the intervals.
Names of the contrasts.
The right axis has non-standard controls.
Logical. The alternate TRUE means display the values of the contrast heights as the left axis tick labels.

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

See Also
See mmc for the references and examples.

---

panel.dotplot.tb  Dotplot with evenly spaced tiebreakers.

Description
Dotplot with evenly spaced tiebreakers. Multiple hits on a specific x value are stacked.

Usage
panel.dotplot.tb(x, y, factor=.1,
  jitter.data=TRUE, horizontal=TRUE,
  max.freq=max(sapply(subsets, length)),
  ...)

Arguments
x, y  See xyplot.
factor  jitter factor, see xyplot. Increment is factor/max.freq where max.freq is the maximum number of duplicates of any x value in any y group.
jitter.data, horizontal
  Always TRUE.
max.freq  maximum number of observation at any combination of response values, factor levels, and group levels. If the formula includes one or more conditioning factors, then the user is responsible for providing a value for max.freq.
...
  Other arguments for xyplot.

Details
Creates (possibly grouped) Dotplot of x against y. y is the ‘factor’.
Warning

If the formula includes one or more conditioning factors, then the user is responsible for providing a value for \texttt{max.freq}. The default behavior is a different \texttt{max.freq} for each panel in a multi-panel display.

Author(s)

Richard M. Heiberger

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

Examples

\begin{verbatim}
x <- c(1,1,2,2,2,5,4,2,1,5)
y <- factor(letters[rep(1:2, 5)])

dotplot(x, panel=panel.dotplot.tb)
dotplot(x, panel=panel.dotplot.tb, factor=.2)
dotplot(y ~ x, panel=panel.dotplot.tb)
dotplot(y ~ x, panel=panel.dotplot.tb, cex=1.5, factor=.15)

quiz <- data.frame(scores=sample(10, 360, replace=TRUE),
  date=rep(rep(c("0902", "0916", "0930"), c(40,40,40)), 3),
  section=rep(c("Stat1-3", "Stat1-5", "Stat1-8"), c(120,120,120)))

dotplot(date ~ scores | section, data=quiz,
  panel=panel.dotplot.tb, factor=.5)

dotplot(date ~ scores | section, data=quiz,
  panel=panel.dotplot.tb, factor=.5,
  layout=c(1,3), between=list(y=1),
  main="Three quizzes for three sections of Stat 1")
\end{verbatim}

## If the formula includes one or more conditioning factors, then the
## user is responsible for providing a value for the argument \texttt{max.freq}
##
a <- rep(1, 10)
z <- c(1,1,2,2,3,2,3,3,1,1)
g <- LETTERS[c(1,1,1,1,1,2,2,2,2,2)]

print(split=c(1,1,2,1), more=TRUE,
dotplot( a ~ z | g, panel=panel.dotplot.tb,
  factor=.6, cex=1.5, layout=c(2,1),
  main="different scaling in each panel")
)

print(split=c(2,1,2,1), more=FALSE,
dotplot( a ~ z | g, panel=panel.dotplot.tb, max.freq=3, 
factor=.6, cex=1.5, layout=c(2,1),
main="same scaling in each panel")
}

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

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

Usage
panel.interaction2wt(x, y, subscripts,
resposnelab, trace.values,
factor.levels, factor.position,
fun = mean,
se,
type="l",
...
box.ratio,
simple=FALSE,
simple.offset,
simple.scale,
simple.pch,
data.x,
col.by.row=TRUE,
key.in=NULL)

strip.interaction2wt(which.given, which.panel, var.name,
factor.levels, shingle.intervals,
strip.names = c(TRUE, TRUE), style = 1, ...)

Arguments
panel.interaction2wt arguments:
levels of x-factor

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
resposnelab  Character name of response variable, defaults to the name of the response variable.
trace.values  levels of trace-factor
fun    Summary function, defaults to mean
se    standard errors to be passed to panel.intxplot. Missing, logical, or a numeric
       vector. If se is missing or FALSE, or if simple is FALSE, then standard errors
       are not plotted. If TRUE, the standard errors are calculated from the sufficient
       statistics for each group as the group’s standard deviation divided by the square
       root of the group’s observation count. If a numeric vector, it is evaluated in the
       environment of the sufficient statistics.

type    See panel.xyplot,
box.ratio passed to panel.bwplot.intermediate.hh,
... extra arguments, primarily color, to be passed to panel.bwplot.intermediate.hh
key.in S-Plus only. Arguments to be passed through to the key for the trace-factor in
each row of the display. The most likely argument is x, which is needed if the
key is not correctly placed. Use, for example, key.in=list(x=-3.5) where
the units are the units of the left column of panels and the value is the location
where the left border of the key should be placed.

factor.position
        "position" attribute of factor.
simple logical. If TRUE, then simple effects are to be displayed.
simple.offset, simple.scale
        named list of offset and scale for the response and trace factors.
        See interaction.positioned for their use.
simple.pch Named list containing plotting characters for each level of one or more of the
        factors. simple.pch is used only when simple==TRUE. If the argument simple.pch
        is missing, then the integers for the levels of the factors are used. The characters
        are used for the median of the box plots in the diagonal panels. They match the
        trace factor of the interaction panel in the same column of the display.
data.x data.frame containing factors from the input data.frame
col.by.row logical. If TRUE (the default), simple effects plots color the simple effects on the
        main diagonals in the same color as the trace levels in their row. If FALSE, then
        simple effects are colored to match the x levels in their column.

strip.interaction2wt arguments
which.given, which.panel, var.name, factor.levels, shingle.intervals
        see documentation for strip.default.
strip.names Force strip.names=TRUE
style force style=1

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

References
Heiberger, Richard M. and Holland, Burt (2015). *Statistical Analysis and Data Display: An In-
See Also

interaction2wt, panel.bwplot.intermediate.hh

Examples

```r
## Not run:
tmp <- data.frame(y=rnorm(48),
  A=factor(rep(1:2, 24)),
  B=factor(rep(rep(1:3, each=2), 8)),
  C=factor(rep(rep(1:4, each=6), 2)))
interaction2wt(y ~ A+B+C, data=tmp,
  key.in=list(x=-3), ## key.in is ignored by R
  xlim=c(.4, 4.5))
interaction2wt(y ~ B+C, data=tmp, key.in=list(x=-2), xlim=c(.4, 4.5))
position(tmp$B) <- c(1, 2.4, 3.8)
interaction2wt(y ~ B+C, data=tmp, key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ B+C, data=tmp, simple=TRUE,
  simple.scale=list(B=.18, C=.27), box.ratio=.2,
  key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ C+B, data=tmp, simple=TRUE,
  simple.scale=list(B=.18, C=.27), box.ratio=.2,
  key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ B+C, data=tmp, simple=TRUE,
  simple.pch=list(C=c(16,17,18,19)),
  key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ C+B, data=tmp, simple=TRUE,
  simple.pch=list(A=c(1:2), B=c(3:5), C=c(16,17,18,19)),
  key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ C+B, data=tmp, simple=TRUE,
  simple.pch=list(A=c(1:2)),
  key.in=list(x=-2), xlim=c(.4, 4.5))
interaction2wt(y ~ B+C, data=tmp, simple=TRUE,
  simple.scale=list(B=.18, C=.27), box.ratio=.2,
  simple.pch=list(B=c(16,17,18)),
  key.in=list(x=-2), xlim=c(.4, 4.5),
  se=TRUE)
## End(Not run)
```

panel.isomeans

isomeans grid for MMC plots.
Description

isomeans grid for MMC plots.

Usage

panel.isomeans(ybar,
    lty.iso=7,
    col.iso='darkgray',
    lwd.iso=1,
    lty.contr0=2,
    col.contr0='darkgray',
    lwd.contr0=1,
    ...,
    col, lwd, lty ## capture potentially ambiguous name
)

Arguments

ybar Vector of means.

lty.iso, col.iso, lwd.iso
    color, line type, line width for the isomeans grid.

lty.contr0, col.contr0, lwd.contr0
    color, line type, line width for the vertical contrast=0 line.

... ignore any additional arguments

col, lwd, lty ignore these arguments. They are captured here to avoid ambiguity with col.iso
    and lty.iso.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

See mmc for the references and examples.
* the panel.barchart2 heights and widths when stack==TRUE are also based on the new stackWidth argument.

panel.likert calls panel.barchart2 scaling of stackWidth:

```
stackWidth <- stackWidth/mean(stackWidth) ## and maybe smaller with another /2
```

### Usage

```r
panel.barchart2(x, y, box.ratio = 1, box.width = box.ratio/(1 + box.ratio),
  horizontal = TRUE, origin = NULL, reference = TRUE, stack = FALSE,
  groups = NULL,
  col = if (is.null(groups)) plot.polygon$col else superpose.polygon$col,
  border = if (is.null(groups)) plot.polygon$border else superpose.polygon$border,
  lty = if (is.null(groups)) plot.polygon$lty else superpose.polygon$lty,
  lwd = if (is.null(groups)) plot.polygon$lwd else superpose.polygon$lwd,
  ..., identifier = "barchart",
  stackWidth=NULL)
```

```
panel.likert(..., horizontal=TRUE, reference.line.col="gray65")
```

### Arguments

- `x, y, box.ratio, box.width, horizontal, origin, reference, stack, groups, col`
  - See `panel.barchart`.
- `border, lty, lwd, identifier`
  - See `panel.barchart`.
- `...`
  - Extra arguments, if any, for `panel.barchart`.
- `stackWidth`
  - Heights in each horizontal stacked bar, when stack==TRUE, are constant and specified by this argument. We recommend starting with `stackWidth <- stackWidth/mean(stackWidth)` and adjusting as seems appropriate.
- `reference.line.col`
  - See `likert`.

### Author(s)

Richard M. Heiberger <rmh@temple.edu>

### See Also

- `likert`
panel.pairs.hh

Function based on S-Plus panel.pairs to add the subpanel.scales and panel.cex arguments.

Description

Function based on S-Plus panel.pairs to add the subpanel.scales and panel.cex arguments. In R, this is an alias for panel.pairs.

Usage

`panel.pairs.hh(x, y, z, subscripts, pscales, subpanel = panel.splom, varnames = dimnames(x)[[2]], ..., subpanel.scales, panel.cex=par()$cex)`

Arguments

- `x, y, z, subscripts, pscales, subpanel, varnames, ...`
  
  See `splom` in S-Plus.

- `subpanel.scales`
  Controls the size of the tick labels in the diagonal panel.

- `panel.cex`
  Controls the size of the variable names in the diagonal panel.

Value

"trellis" object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

`splom` in S-Plus.

Examples

```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.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 \( \text{lm}(y \sim x) \) for each panel in an additional strip label.

...

Remaining arguments to panel.xyplot.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

xysplom

---

partial.corr  

partial correlations

Description

The partial correlation of x and y conditioning on z is the ordinary correlation of the residuals from the regression of x on z and the regression of y on z.

Usage

partial.corr(vars, cond)
Arguments

vars  matrix of data.frame of all the variables to be correlated.
cond  matrix of data.frame of all the variables to be conditioned on.

Value

matrix of partial correlations of the numeric variables in the argument vars conditioning on the numeric variables in cond.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

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

Description

Construct a pdf file from a "latex" file. See Hmisc::latex for concepts.

Usage

```
pdf.latex(latex.object, ..., file, overwrite = TRUE, copy.mode = TRUE, copy.date = TRUE)
```

Arguments

latex.object  Result from a call to Hmisc::latex().
...  Optional arguments to Hmisc::dvi()
file  File name in getwd() to place resulting pdf file.
overwrite  If the file already exists, TRUE means replace it.
copy.mode, copy.date  If TRUE copy file mode and date from temporary directory to getwd().

Value

Filename of class "dvi"
Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

latex

Examples

## you will normally need these options. See ?Hmisc::latex for details.
options(latexcmd = 'pdflatex')
options(dviExtension = 'pdf')
options(xdvicmd = 'open')  ## Macintosh, Windows, SMP linux

## Not run:
## these examples place files in your current working directory

## matrix
tmp <- array(1:20, c(4,5), list(LETTERS[1:4], LETTERS[5:9]))
tmp

pdf.latex(latex(tmp))  ## for matrix, accept the default structure.tex and structure.pdf filenames.

pdf.latex(latex(tmp, title = "tmp"))  ## specify name of .tex and .pdf file.

## 3D array
tmp3 <- array(1:40, c(4,5,2), list(LETTERS[1:4], LETTERS[5:9], LETTERS[10:11]))
tmp3

pdf.latex(latex(tmp3))  ## for array, the default base filename is the
                       ## name of the argument, hence tmp3.tex and tmp3.pdf

pdf.latex(latex(tmp3, title = "somethingelse"))  ## or specify somethingelse

## End(Not run)

pdiscunif  

Discrete Uniform Distribution

Description

Discrete Uniform Distribution
Usage

pdiscunif(q, size)
qdiscunif(p, size)
ddiscunif(q, size)
rdiscunif(n, size)

Arguments

size parameter of distribution. Numbers from 1 to size are equally likely.
q Quantiles.
p Probability.
n number of items in the random sample.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

q <- seq(-.5, 7.5, .5)
pp <- pdiscunif(q, 6)

## xyplot(pp ~ q,
## scales=list(
## x=list(at=floor(min(q)):ceiling(max(q))),
## y=list(at=seq(0, 1, .1))))

qq <- qdiscunif(pp, 6)

dd <- ddiscunif(q, 6)

cbind(q, pp, qq, dd)

rdiscunif(12, 6)

Description

Helper functions for regr2.plot.
plot.hov

Usage

perspPlane(x, y, z, persp.out, ...)
perspFloor(x, y, z, persp.out, ...)
perspBack.wall.x(x, y, z, persp.out, ...)
perspBack.wall.y(x, y, z, persp.out, ...)

Arguments

x, y, z  Arguments to trans3d in R, or perspp in S-Plus.
persp.out  Result from previous call to persp.
...  Additional arguments to persp.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

regr2.plot

plot.hov  Homogeneity of Variance Plot

Description

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

Usage

hovPlot(x, data=NULL, method = "bf", ## x is a formula
transpose = TRUE, ...)

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

hovPlot.bf(x, group, ## x is the response variable
  y.name = deparse(substitute(x)),
  group.name = deparse(substitute(group)),
  transpose = TRUE, ...)

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

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

- **x**: Formula appropriate for oneway anova in `hovPlot`. Response variable in `hovPlot.bf`.
- **data**: `data.frame`
- **method**: Character string defining method. At this time the only recognized method is "bf" for the Brown-Forsyth method.
- **transpose**: Always TRUE in R. Normally TRUE in S-Plus to force vertical boxplots.
- **group**: factor.
- **y.name**: name of response variable, defaults to variable name in formula.
- **group.name**: name of factor, defaults to variable name in formula.
- **...**: additional arguments.

Value

"trellis" object with three panels containing boxplots for each group: The observed data "y", the data with the median subtracted "y-med(y)" , and the absolute deviations from the median "abs(y-med(y))". The Brown and Forsyth test statistic is the $F$ statistic resulting from an ordinary one-way analysis of variance on the data points in the third panel.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

- `aov`, `hov`

Examples

```r
data(turkey)
hov(wt.gain ~ diet, data=turkey)
hovPlot(wt.gain ~ diet, data=turkey)
```
MMC (Mean–mean Multiple Comparisons) plot. The plot method documented here is no longer recommended for R; use `mmcplot` instead. This method is still necessary for S-Plus.

Usage

```r
## S3 method for class 'mmc.multicomp'
plot(x,
xlab="contrast value",
ylab=none$ylabel,
focus=none$focus,
focus.main= main.method.phrase,
focus.main2=main2.method.phrase,
focus.main.method.phrase= paste("multiple comparisons of means of", ylab),
focus.main2.method.phrase=paste("simultaneous ",
   100*(1-none$alpha),"% confidence limits, ",
   method, " method", sep=" "),
focus.ry.mmc=TRUE,
key.x=par()$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.
plot.mmc.multicomp

ylab name of response variable
focus define the factor to compute contrasts of.
main, main2 main and second line of title of plot
main.method.phrase, main2.method.phrase default expressions for title of plot
ry.mmc range of values on the y-axis. It is similar to par("ylim"), but not the same as additional calculations are needed to maintain the isomeans grid as a square.
key.x, key.y location of the key displayed when iso.name=FALSE.
method method used to construct contrasts and confidence intervals. See the type argument to glht for the list.
print.lmat logical. If TRUE, then display the user-specified contrasts.
print.mca logical. If TRUE, then display the pair-wise contrasts.
iso.name logical. If TRUE, label the isomeans grid with the factor levels. If FALSE, label the isomeans grid with sequential numbers and display a key relating the numbers to the factor levels.
x.offset amount to move the vertical 0 line to the left or right to reduce overprinting of labels and plotted lines.
col.mca.signif, lty.mca.signif, lwd.mca.signif color, line type, line width for significant pairwise contrasts.
col.mca.not.signif, lty.mca.not.signif, lwd.mca.not.signif color, line type, line width for non-significant pairwise contrasts.
col.lmat.signif, lty.lmat.signif, lwd.lmat.signif color, line type, line width for significant user-specified contrasts.
col.lmat.not.signif, lty.lmat.not.signif, lwd.lmat.not.signif color, line type, line width for non-significant user-specified contrasts.
lty.iso, col.iso, lwd.iso color, line type, line width for the isomeans grid.
lty.contr0, col.contr0, lwd.contr0 color, line type, line width for the vertical contrast=0 line.
decdigits.ybar number of decimal digits in the left-axis labels.
... other arguments, currently ignored.

Note

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

When there is overprinting of labels (a consequence of level means being close together), a tiebreaker plot may be needed. See ?MMC for an example.

Author(s)

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


See Also

 mmc.plotMatchMMC, mmcplot.

Examples

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

## See \?MMC to see why these contrasts are chosen
catalystm.lmat <- cbind("AB-D" =c(1, 1, 0,-2),
 "A-B" =c( 1,-1, 0, 0),
 "ABD-C"=c( 1, 1,-3, 1))
dimnames(catalystm.lmat)[[1]] <- levels(catalystm$catalyst)

catalystm.mmc <-
if.R(r={mmc(catalystm1.aov, linfct = mcp(catalyst = "Tukey"),
 focus.lmat=catalystm.lmat)},
 s={multicomp.mmc(catalystm1.aov, focus.lmat=catalystm.lmat,
 plot=FALSE)}
)

## Not run:
## pairwise contrasts, default settings
plot(catalystm.mmc, print.lmat=FALSE)

## End(Not run)

## Centering, scaling, emphasize significant contrasts.
## Needed in R with 7in x 7in default plot window.
## Not needed in S-Plus with 4x3 aspect ratio of plot window.
plot(catalystm.mmc, x.offset=2.1, ry.mmc=c(50,58), print.lmat=FALSE)

## user-specified contrasts
plot(catalystm.mmc, x.offset=2.1, ry.mmc=c(50,58))

## reduce intensity of isomeans grid, number isomeans grid lines
plot(catalystm.mmc, x.offset=2.1, ry.mmc=c(50,58),
 lty.iso=2, col.iso=’darkgray’, iso.name=FALSE)
## both pairwise contrasts and user-specified contrasts
plot(catalystm.mmc, x.offset=2.1, ry.mmc=c(50,58), lty.iso=2, col.iso="darkgray", print.mca=TRUE)

## Not run:
## newer mmcplot
mmcplot(catalystm.mmc)
mmcplot(catalystm.mmc, type="lmat")

## End(Not run)

**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 R, `plot.multicomp.adjusted` replaces the bounds calculated by `multcomp:::confint.glht` with bounds based on a common standard error for a set of anova tables that are partitioned for the simple effects on an analysis conditioned on the levels of one of the factors. In S-Plus, `plot.multicomp.hh` augments the standard `plot.multicomp` to give additional user arguments to control the appearance of the plot.

`plotMatchMMC` uses the `plot.multicomp.hh` code. `plotMatchMMC` must immediately follow a plot of an `mmc.multicomp` object and is applied to either the `$mca` or `$lmat` component of the `mmc.multicomp` object. `plotMatchMMC` is used as a tiebreaker plot for the MMC plot. `plotMatchMMC` matches the horizontal scaling of the MMC plot and displays the individual contrasts in the same order as the MMC plot. See `mmc` for examples.

These functions are no longer recommended. Use `mmcplot` instead.

**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 = TRUE, plt.in = c(0.2, 0.9, 0.1, 0.9),
     x.label.adj=1, xrange.include=href, xlim,
     comparisons.per.page=21,
     col.signif=1, col.not.signif=1,
     lty.signif=4, lty.not.signif=4,
     lwd.signif=1, lwd.not.signif=1,
```
..., xlabel.print=TRUE, y.axis.side=2, ylabel.inside=FALSE)

plotMatchMMC(x, ..., xlabel.print=FALSE, cex.axis=par()$cex.axis, col.signif='red', main='', ylabel.inside=FALSE, y.axis.side=4, adjusted=FALSE)

Arguments

x A "multicomp" object. plotMatchMMC will also accept a mmc.multicomp object. It will use the lmat component if there is one, otherwise it will use the mca component.

ylabel Y label on graph.

y.axis.side Y labels are on the left by default when plotting a "multicomp" object. We move them to the right when matching the x-axis of an MMC plot.

... other arguments to plot.multicomp.

ylabel.inside Logical value, if FALSE (the default), the plotMatchMMC right-axis labels are in the margin. If TRUE, the right-axis labels are in the figure area. Setting the argument to TRUE makes sense when plotting the lmat component of an mmc.multicomp object.

href reference line for the intervals. The default is 0. S-Plus only.

xrange.include xlim will be extended to include these values. S-Plus only.

uniform S-Plus only. Logical value, if TRUE and the plots fill more than one page, the scale will be uniform across pages.

plt.in S-Plus only. Value for par("plt") to make better use of the space on the plotting page.

x.label.adj S-Plus only. This is the par("adj") applied to the x-location of the y.labels on the multicomp plot.

xlim x-range of the plot.

comparisons.per.page The default S-Plus plot.multicomp hardwires this to 21, which allows for all pairwise comparisons of 7 levels taken 2 at a time. The HH plot.multicomp makes it a variable. Use it together with plt.in to make better use of the space on the plot. S-Plus only.

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

col.signif Color for significant comparisons. S-Plus only for plot.multicomp. Both R and S-Plus for plotMatchMMC.

col.not.signif, lty.not.signif, lwd.not.signif Color, line type, and line width for non-significant comparisons. S-Plus only.
**Value**

`plot.multicomp` plots a "multicomp" object. In S-Plus, this masks the standard `plot.multicomp` in order to provide additional arguments for controlling the appearance. It defaults to the standard appearance. In R, it coerces its argument to a "glht" object and plots that with the appropriate plot method.

**Note**

The multiple comparisons calculations in R and S-Plus use completely different packages.

Multiple comparisons in R are based on `glht`. Multiple comparisons in S-Plus are based on `multicomp`. The MMC plot in the HH package 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`.
Examples

```r
## data and ANOVA
data(catalystm)
catalystm1.aov <- aov(concent ~ catalyst, data=catalystm)
summary(catalystm1.aov)
catalystm.mca <- if.R(r=glht(catalystm1.aov, linfct = mcp(catalyst = "Tukey")),
s=multicomp(catalystm1.aov, plot=FALSE))
if.R(s=plot(catalystm.mca),
  r=plot(confint(catalystm.mca, calpha=qtukey(.95, 4, 12)/sqrt(2))))
```

```
## calpha is strongly recommended in R with a large number of levels
## See ?MMC for details.
```

---

**position**

*Find or assign the implied position for graphing the levels of a factor. A new class "positioned", which inherits from "ordered" and "factor", is defined.*

Description

The default values for plotting a factor `x` are the integers `1:length(levels(x))`. These functions provide a way of specifying alternate plotting locations for the levels.

Usage

```
position(x)
```

```
position(x) <- value
```

```
## S3 method for class 'positioned'
is.numeric(x, ...)
## S3 method for class 'positioned'
as.numeric(x, ...)
## S3 method for class 'positioned'
x[... , drop=FALSE]
## S3 method for class 'positioned'
is.na(x)
## S3 method for class 'positioned'
as.positioned(x)
## S3 method for class 'positioned'
positioned(x, ..., value)
## S3 method for class 'positioned'
print(x, ...)
## S3 method for class 'positioned'
unique(x, incomparables = FALSE, ...)
```

unpositioned(x, ...)
position

Arguments

- `x` numeric vector or factor
- `value` numerical values to be associated with `levels(x)`. The length(`value`) must equal `length(levels(as.factor(x)))`.
- `...` other arguments.
- `drop` See `Extract`.
- `incomparables` See `unique`.

Value

position(x) <- value first forces its argument to be an ordered factor and then assigns the value to the "position" attribute of the ordered factor. The result is assigned class "positioned" and returned.

position(x) returns the position values associated with `levels(x)`. If `x` is a positioned factor, then the "position" attribute is returned. If `x` is a factor, then the integers 1:length(`levels(x)`) are returned. For anything else, as.numeric(`x`) is returned.

as.position(x) returns a numeric vector the length of the original vector. If `x` inherits from "factor", then the values in the vector are the values in `position(x)` subscripted by the levels of the factor. If `x` is numeric, then `x` itself is returned.

unpositioned(x) removes the "position" attribute and removes the "positioned" value from the the class of the object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

panel.interaction2wt, factor.

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.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## position is assigned to ordered in specified order
tmp <- ordered(c("cm","mm","m","m","mm","cm"),
```
```r
levels(c("mm","cm","m")) ## size order
levels(tmp)
position(tmp) <- c(-3, -2, 0) ## log10 assigned in size order
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## numeric stays numeric
tmp <- c(0.010, 0.001, 1.000, 1.000, 0.001, 0.010)
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## factor with numeric levels, position is integer position in size order
tmp <- factor(c(0.010, 0.001, 1.000, 1.000, 0.001, 0.010))
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## ordered with numeric levels, position is numeric value in size order
tmp <- ordered(c(0.010, 0.001, 1.000, 1.000, 0.001, 0.010))
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
as.positioned(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## factor with numeric levels
## position is assigned in size order
tmp <- factor(c(0.010, 0.001, 1.000, 1.000, 0.001, 0.010))
```
levels(tmp)
position(tmp) <- c(-3, -2, 0) ## log10 assigned in size order
tmp
as.numeric(tmp)
levels(tmp)
position(tmp)
as.position(tmp)
positioned(tmp)
unpositioned(tmp)
unique(tmp)

## boxplots coded by week
tmp <- data.frame(Y=rnorm(40, rep(c(20,25,15,22), 10), 5),
                  week=ordered(rep(1:4, 10)))
position(tmp$week) <- c(1, 2, 4, 8)
bwplot(Y ~ week, horizontal=FALSE,
       scales=list(x=list(limits=c(0,9),
                        at=position(tmp$week),
                        labels=position(tmp$week))),
data=tmp, panel=panel.bwplot.intermediate.hh)

#### 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","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","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.
print.latticeresids

Description

Print a latticeresids object.

Usage

## S3 method for class 'latticeresids'
print(x, ...,
   A321.left=0, A321.bottom=0.27,
   A4.left=0, A4.top=0.30,
   position=list(
      A321=c(A321.left, A321.bottom, 1, 1 ),
      A4  =c(A4.left, 0, 1, A4.top)),
   panel.width=NULL,
   which=1:4)

Arguments

x       A latticeresids object.
A321.left, A321.bottom, A4.left, A4.top, position
   The first three rows are on the same x scale (the scales of the independent variables). The arguments with "A321" in their name are used to construct the position argument to print.trellis for the first three rows. The fourth row is on a different x scale (the scales of each independent variable adjusted for all the other x variables). The arguments with "A4" in their name are used to construct the position argument to print.trellis for the fourth row. The two sets of rows \{1,2,3\} and \{4\} may have different widths for their left axis tick labels. The arguments A321.left and A4.left along with absolute dimensions for panel.width ("cm" or "in", not "npc") can be hand-tailored to make the columns line up precisely. See the example.

panel.width the panel.width argument of print.trellis.
which       Vector of row numbers which are to be printed. If not all four printed, consider adjusting the A321.bottom and A4.top values.

Details

The four trellis objects, one for each type of plot, are printed as a single four-row lattice object.

Author(s)

Richard M. Heiberger <rmh@temple.edu>
print.NormalAndTplot

Print method for Normal and t plots from NTplot.

Description
Print method for Normal and t plots from NTplot.

Usage

## S3 method for class 'NormalAndTplot'
print(x, tablesOnPlot=TRUE, plot=TRUE,
scales=FALSE, prob=FALSE, call=FALSE,
..., cex.table=.7, digits=attr(x, "call.list")$digits,
position.2=.17)

Arguments

x A "NormalAndTplot" object.
tablesOnPlot Logical. If TRUE, display the tables in the attr(x, "scales") and attr(x, "prob") on the plot.
plot Logical. If TRUE, display the graph on the plot.
scales, prob Logical. If TRUE, display the specified attribute on the R Console.
call Logical. If TRUE, display an R statement on the R console.
... Other arguments are ignored.
cex.table, digits
cex and digits for the tablesOnPlot display of the attr(x, "scales") and attr(x, "prob") tables.
position.2 When tablesOnPlot=TRUE, the graph occupies the top of the device beginning at position.2. This is the second value in the position argument of print.trellis.

Value
The argument is returned invisibly.

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

See Also
residual.plots.lattice
Print a "tsdiagplot" object.

Description

Print a "tsdiagplot" object.

Usage

## S3 method for class 'tsdiagplot'
print(x, ..., portrait=FALSE)
print1.tsdiagplot(x)
print2.tsdiagplot(x)

Arguments

x    a "tsdiagplot" object
...
Optional arguments to print. The only ... argument that is used is pages. If
pages is not used or pages==1, then use print1.tsdiagplot. If pages!=1,
then use print2.tsdiagplot.
portrait logical. If FALSE, arrange the panels for a landscape orientation (pdf with width=12
inches looks good). If TRUE, arrange the panels for a portrait orientation (pdf
with height=13 inches looks good).

Details

A "tsdiagplot" object is a collection of several "trellis" objects. We provide two options for
printing them.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

tsdiaclipse
print.TwoTrellisColumns

Print two conformable trellis plots in adjacent columns with user control of widths.

Description

Print two conformable trellis plots in adjacent columns with user control of widths. Left y tick-labels and left.strip are removed from the right-hand plot.

Usage

as.TwoTrellisColumns5(left, ## left is the left trellis object
right, ## right is the right trellis object
## Both left and right must have identical
## settings for number and size of vertical panels,
## left-axis labels, number of lines in main, sub, legend.
..., pw=c(.3, .30, .01, .30, .09),
px=list(
  LL=c(0, pwc[1]),
  LP=pwc[1:2],
  ML=pwc[2:3],
  RP=pwc[3:4],
  RL=pwc[4:5]),
pwc=cumsum(pw),
strip.left=TRUE,
y.tck=c(0,0)
)

## S3 method for class 'TwoTrellisColumns5'
print(x, px=attr(x, "px"), ...)

leftLabels.trellis(x)
rightLabels.trellis(x)
panelOnly.trellis(x, strip.left=FALSE, y.tck=0)
mainSubLegend.trellis(x)
emptyRightAxis(x)

Arguments

left, right Conformable "trellis" objects. Both must have the identical settings for number and size of vertical panels, left-axis labels, number of lines in main, sub, legend.

x "trellis" object.
px

These are used x-values used in the position argument of the print.trellis function. The default (constructed from the pw argument) makes the Left and Right panels the same width and the Middle containing the y-axis is given the remainder. Overlapping is permitted. The appearance depends on the width of the graphics device.

pw, pwc

pw vector of five positive numbers that sum to 1. These are the relative widths of the five sections of the result: LeftLabels, LeftPanel, MainSubLegend, RightPanel, RightLabels. pwc is the cumulative sum of pw. pwc is expanded in the px argument to the x values used in the position argument of the print.trellis function.

strip.left

See barchart.

y.tck

A vector of one or two numeric values. This will be used as the y.tck value for the right column of panels. See 'tck' in barchart for details.

...

Other arguments are ignored.

Details

as.TwoTrellisColumns5 constructs a "TwoTrellisColumns5" object, which is a list of five trellis objects named "LL", "LP", "ML", "RP", "RL". LL is the left labels from the left input object. LP is the panels from the left input object. ML is the middle labels from the left object; these are the main title, sub title, and legend. RP is the panels from the right input object. RL is the right labels from the right input object.

print.TwoTrellisColumns5 is a print method for a "TwoTrellisColumns5" object. It takes left-to-right positioning information from the "px" attribute of its argument x or from an input argument. The numbers are used as the "x" information for the position argument to the print.trellis method.

emptyLeftAxis, leftLabels.trellis, rightLabels.trellis, panelOnly.trellis, mainSubLegend.trellis, emptyLeftStrip, emptyRightAxis are functions which blank out the various components of the trellis argument and retains their vertical spacing.

Value

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert for the details on the motivating example.

Examples

## These are based on the Professional Challenges example in ?likert
data(ProfChal)
levels(ProfChal$Subtable)[6] <- "Prof Recog" ## reduce length of label

## initial ordering of Question factor
PCC <- likert(Question ~ . | Subtable, ProfChal, ylab=NULL,
rightAxis=TRUE,
layout=c(1,6),
strip=FALSE,
strip.left=strip.custom(bg="gray97"),
par.strip.text=list(cex=.7),
scales=list(y=list(relation="free"),
main="Is your job professionally challenging?")

## initial ordering of Question factor
PCP <- likert(Question ~ . | Subtable, ProfChal, ylab=NULL,
as.percent=TRUE,
layout=c(1,6),
strip=FALSE,
strip.left=strip.custom(bg="gray97"),
par.strip.text=list(cex=.7),
scales=list(y=list(relation="free"),
main="Is your job professionally challenging?")

## Not run:
## default equal widths of the two panels
as.TwoTrellisColumns5(PCP, PCC) ## 11in x 7in

## make left panel twice as wide as right panel
as.TwoTrellisColumns5(PCP, PCC, pw=c(.3, .4, .01, .2, .09)) ## 11in x 7in

## make left panel twice as wide as right panel, and control position of main and legend
as.TwoTrellisColumns5(PCP, PCC, ## 11in x 7in
px=list(
    LL=c(.00, .30),
    LP=c(.30, .70),
    ML=c(.60, .61), ## arbitrary,
    RP=c(.71, .91),
    RL=c(.91, 1.00)))

## End(Not run)

## Size that works in default 7x7 window. 7x7 is not recommended for
## this example because most of the space is used for labeling and not
## much for the panels containing the data. Use the px values for the
## 11x7 illustrated above in the dontrun section.

as.TwoTrellisColumns5(PCP, PCC, ## 7in x 7in
px=list(
    LL=c(.00, .50),
    LP=c(.50, .70),
    ML=c(.50, .51), ## arbitrary,
## visually center the labels and legend
RP=c(.71, .87),
RL=c(.87, 1.00))

## Ordering the rows by the lengths of the positive bars and also
## put percents and counts on the same plot.
## The easiest way is to use the LikertPercentCountColumns function:

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

## Not run:
## Ordering the rows by the lengths of the positive bars and also
## putting percents and counts on the same plot requires coordination.
## The easiest way is to order the original tables of counts by the
## order of the percent plot.

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

## percentPlot
pct.order <- percentPlot$y.limits[[1]]

ProfChal2 <- ProfChal
ProfChal2$Question <- factor(ProfChal2$Question, levels=rev(pct.order))

countPlot <- likert(Question ~ . | Subtable, ProfChal2,
layout=c(1,6),
rightAxis=TRUE,
scales=list(y=list(relation="free"),
          x=list(at=c(0, 250, 500))),
ylab=NULL, between=list(y=0),
strip.left=strip.custom(bg="gray97"), strip=FALSE,
par.strip.text=list(cex=.7),
main="Is your job professionally challenging?")

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

## Size that works in default 7x7 window. 7x7 is not recommended for
## this example because most of the space is used for labeling and not
## much for the panels containing the data. Use the px values for the
# 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`
Description

Prints a likert plot in the traditional format for a population pyramid, with the Left and Right sides in separate panels, with the x-tick marks on the left side made positive, and with the y-axis in the Middle.

Usage

```r
## S3 method for class 'pyramidLikert'
print(x, ...,
  panel.width=.48,
  px=list(
    L=c(0, panel.width),
    R=c(1-panel.width, 1),
    M=c(panel.width, 1-panel.width)),
  keepLegend=(length(x$legend$bottom$args$text) > 2),
  xlab.top=list(
    L=list(x$legend$bottom$args$text[1]),
    R=list(x$legend$bottom$args$text[2]),
    M=list(x$ylab, just=1)))
```

```r
as.pyramidLikert(x, ...,
  panel.width=.48,
  px=list(
    L=c(0, panel.width),
    R=c(1-panel.width, 1),
    M=c(panel.width, 1-panel.width)),
  keepLegend=(length(x$legend$bottom$args$text) > 2),
  xlab.top=list(
    L=list(x$legend$bottom$args$text[1]),
    R=list(x$legend$bottom$args$text[2]),
    M=list(x$ylab, just=1)))
```

Arguments

- `x` a single-panel 'trellis' object.
- `...` Other arguments are ignored.
- `panel.width` Numeric scalar between 0 and 0.5. Common width of left and right panels. The default value .48 value works well for the USAge. table example. This number is expanded in the px argument to the x values used in the position argument of the `print.trellis` function.
- `px` x values used in the position argument of the `print.trellis` function. The default makes the Left and Right panels the same width and the Middle containing the y-axis is given the remainder.
keepLegend

If TRUE and x contains a bottom legend, then it is printed along with the Middle section containing the y-axis. If FALSE or there is no bottom legend, then the bottom legend is not printed.

xlab.top

A vector of three labels. The default is designed for a population triangle with two levels (usually, Male on one side and Female on the other). The Left and Right labels are taken from the first two labels in the legend. The Middle value is the variable name for the y-axis.

Details

This is a print method for population triangles. It is designed for a likert plot with one left-side level and one right-side level. It works for any single-panel "trellis" object, in the sense that it produces a plot.

Value

The input argument x.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert

Examples

data(USAge.table)  ## from latticeExtra
USA79 <- USAge.table[75:1, 2:1, "1979"]/1000000
PL <- plot(as.likert(USA79),
    main="Population of United States 1979 (ages 0-74)",
    xlab="Count in Millions",
    ylab="Age",
    scales=list(
        y=list(
            limits=c(0,77),
            at=seq(1,76,5),
            labels=seq(0,75,5),
            tck=.5))
)

PL
as.pyramidLikert(PL)

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

## Not run:
## run the shiny app
if (interactive()) shiny::runApp(system.file("shiny/PopulationPyramid", package="HH"))

## End(Not run)

## For additional examples, see demo(PoorChildren, package="HH")

---

**rbind.trellis**

*Extend matrix reshaping functions to trellis objects.*

**Description**

Extend matrix reshaping functions to trellis objects. See the details section for comparisons with similar functions in the *lattice* package.

**Usage**

```r
transpose(x)
## S3 method for class 'trellis'
transpose(x)
## Default S3 method:
transpose(x)
## S3 method for class 'trellis'
aperm(a, perm, ...)
## S3 method for class 'trellis'
rbind(..., deparse.level=1,
  combineLimits=TRUE, useOuterStrips=TRUE)
## S3 method for class 'trellis'
cbind(..., deparse.level=1,
  combineLimits=TRUE, useOuterStrips=TRUE)
```

**Arguments**

- `...`, `x`, `a` A set of trellis objects.
- `perm` Permutation vector, see `aperm` for details.
- `combineLimits`, `useOuterStrips` logical. If `TRUE` (the default), use the similarly named *latticeExtra* functions before returning the result.
deparse.level See \texttt{cbind} for details. These functions ignore this argument and always use the names(list(...)), if non-NULL, for the labels. If NULL, then the first length(list(...)) uppercase letters are used.

Details

t\texttt{transpose.trellis} tries to capture and modify all potentially relevant trellis components. \texttt{transpose.trellis} is more comprehensive than the similar \texttt{t.trellis} which adjusts only the \texttt{perm.cond} component.

\texttt{aperm.trellis} does not attempt to check all potentially relevant trellis components. It does not adjust \texttt{layout.heights}, \texttt{layout.widths}, or \texttt{between}. It may show strange axis positions or strip positions for any non-standard arrangement, for example, for any trellis object that has already been through \texttt{latticeExtra::combineLimits}.

Value

trellis object constructed from arguments with new dim and layout.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

F <- xyplot((1:15) ~ (1:15) | rep(factor(letters[3:5]), each=5))
G <- xyplot((1:18) ~ (1:18) | rep(factor(letters[3:5]), each=6))
\texttt{rbind(AAA=F, BBB=G)}
\texttt{cbind(AAA=F, BBB=G)}

tmp <- data.frame(y=1:24,
x=1:24,
a=rep(letters[1:2], each=12),
b=rep(letters[3:5], each=4, times=2),
c=rep(letters[6:9], times=6))

t3 <- xyplot(y ~ x | c*b*a, data=tmp,
panel=function(x, y, ...) panel.text(x, y, y),
scales=list(alternating=FALSE))

## t3
\texttt{t3u <- update(t3, layout=c(4*3, 2), between=list(x=c(0,0,0,1)), main="t3")}
\texttt{useOuterStripsT2L1(t3u)}

## Not run:
## update(t3, layout=c(24, 1))

\texttt{t3.321 <- aperm(t3, c(3,2,1))}
\texttt{update(t3.321, main="t3.321", layout=c(6,4), between=list(x=c(0,1)))}  ## 2*3,4
\texttt{try(transpose(t3))}  ## requires a one- or two-dimensional trellis object.

\texttt{t3.123 <- aperm(t3, c(1,2,3))}  ## identity operation
\texttt{t3.132 <- aperm(t3, c(1,3,2))}
t3.213 <- aperm(t3, c(2,1,3))
t3.231 <- aperm(t3, c(2,3,1))
t3.312 <- aperm(t3, c(3,1,2))
t3.321 <- aperm(t3, c(3,2,1))

u3.123 <- update(t3.123, main="t3.123", layout=c(12,2),
                between=list(x=c(0,0,0,1))) ## 4*3,2
u3.132 <- update(t3.132, main="t3.132", layout=c(8,3),
                between=list(x=c(0,0,0,1))) ## 4*2,3
u3.213 <- update(t3.213, main="t3.213", layout=c(3,8),
                between=list(y=c(0,0,0,1)), par.strip.text=list(cex=.8)) ## 3,4*2
u3.231 <- update(t3.231, main="t3.231", layout=c(6,4),
                between=list(y=c(0,0,1))) ## 2*3,4
u3.312 <- update(t3.312, main="t3.312", layout=c(2,12),
                between=list(y=c(0,0,0,1)), par.strip.text=list(cex=.6)) ## 2,3*4
u3.321 <- update(t3.321, main="t3.321", layout=c(6,4),
                between=list(y=c(0,1))) ## 2,3*4

u5 <- tempfile("u5", fileext = ".pdf")
pdf(u5, width=17, height=22)
print(u3.123, split=c(1,1,2,3), more=TRUE)
print(u3.132, split=c(2,1,2,3), more=TRUE)
print(u3.213, split=c(1,2,2,3), more=TRUE)
print(u3.231, split=c(1,2,2,3), more=TRUE)
print(u3.312, split=c(1,3,2,3), more=TRUE)
print(u3.321, split=c(1,3,2,3), more=FALSE)
dev.off()

try(transpose(t3.123)) ## layout is a matrix, but dim is not.

## Not run:
t2 <- xplot(y ~ x | b*c, data=tmp,
            panel=function(x, y, ...) panel.text(x, y, y),
            scales=list(alternating=FALSE))
t2

## aperm(t2, 1:2) ## identity

transpose(t2)
aperm(t2, 2:1)

t1a <- xplot(y ~ x | b, data=tmp[tmp$a=='a',])
t1b <- xplot(y ~ x | b, data=tmp[tmp$a=='b',])
t1a

t1b

rbind(t1a, t1b)
rbind(AAA=t1a, BBB=t1b)

cbind(t1a, t1b)
cbind(AAA=t1a, BBB=t1b)
## End(Not run)

### 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**

```r
regr1.plot(x, y, model=lm(y~x), coef.model, alt.function, main="put a useful title here", xlab=deparse(substitute(x)), ylab=deparse(substitute(y)), jitter.x=FALSE, resid.plot=FALSE, points.yhat=TRUE, pch=16, ..., length.x.set=51, x.name, pch.yhat=16, cex.yhat=par()$cex*.7, err=-1)
```

**Arguments**

- **x**: x variable
- **y**: y variable
- **model**: Defaults to the simple linear model `lm(y ~ x)`. Any model object with one x variable, such as the quadratic `lm(y ~ x + I(x^2))` can be used.
- **coef.model**: Defaults to the coefficients of the model argument. Other intercept and slope coefficients for a straight line (for example, `c(3,5)`) can be entered to illustrate the sense in which they are not "least squares".
- **alt.function**: Any function of a single argument can be placed here. For example, `alt.function=function(x) {3 + 2*x + 3*x^2}`. All coefficients must be specified.
main, xlab, ylab

arguments to plot.

jitter.x logical. If TRUE, the x is jittered before plotting. Jittering is often helpful when there are multiple y-values at the same level of x.

resid.plot If FALSE, then do not plot the residuals. If "square", then call resid.squares to plot the squared residuals. If TRUE (or anything else), then call resid.squares to plot straight lines for the residuals.

points.yhat logical. If TRUE, the predicted values are plotted.

... other arguments.

length.x.set number of points used to plot the predicted values.

x.name If the model argument used a different name for the independent variable, you might need to specify it.

pch Plotting character for the observed points.

pch.yhat Plotting character for the fitted points.

cex.yhat cex for the fitted points.

err The default -1 suppresses warnings about out of bound points.

Note

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

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

resid.squares

Examples

data(hardness)

## linear and quadratic regressions
hardness.lin.lm <- lm(hardness ~ density, data=hardness)
hardness.quad.lm <- lm(hardness ~ density + I(density^2), data=hardness)

anova(hardness.quad.lm) ## quadratic term has very low p-value
par(mfrow=c(1,2))

regr1.plot(hardness$density, hardness$hardness,
    resid.plot="square",
    main="squared residuals for linear fit",
    xlab="density", ylab="hardness",
    points.yhat=FALSE,
    xlim=c(20,95), ylim=c(0,3400))

regr1.plot(hardness$density, hardness$hardness,
    model=hardness.quad.lm,
    resid.plot="square",
    main="squared residuals for quadratic fit",
    xlab="density", ylab="hardness",
    points.yhat=FALSE,
    xlim=c(20,95), ylim=c(0,3400))

par(mfrow=c(1,1))

regr2.plot(x, y, z,
    main.in="put a useful title here",
    resid.plot=FALSE,
    plot.base.plane=TRUE,
    plot.back.planes=TRUE,
    plot.base.points=FALSE,
    eye=NULL, ## S-Plus
    theta=0, phi=15, r=sqrt(3), ticktype="detailed", ## R
    ...)
eye
S-Plus only. See
persp.
theta, phi, r, ticktype
R only. See
persp.
...
Other arguments to persp.

Value
"Viewing Transformation" for projecting 3D coordinates (x,y,z) into the 2D plane. See persp for
details.

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". The demo called in the examples section shows the geometry of regression
coefficients, the change in predicted y when x1 is changed one unit holding all other x variables
constant.

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

References
Heiberger, Richard M. and Holland, Burt (2015). Statistical Analysis and Data Display: An Inter-

See Also
resid.squares, regr1.plot, persp

Examples
data(fat)
regr2.plot(fat[,"abdomin"], xlab="abdomin",
  fat[,"biceps"], ylab="biceps",
  fat[,"bodyfat"], zlab="bodyfat",
  resid.plot="square",
  eye=c(335.5, 115.65, 171.9), ## used only in S-Plus
  theta=140, phi=35, r=sqrt(15), ## used only in R
  box=is.R(),
  plot.back.planes=FALSE,
  main="Least-squares with two X-variables")

## Not run:
demo("regr2", package="HH", ask=FALSE)
## run the file manually to see the individual steps.

## End(Not run)

### regrresidplot

**Draw a plot of y vs x from a linear model object, with residuals indicated by lines or squares.**

**Description**

Draw a plot of response vector y vs predictor variable x from a linear model object all of whose predictors are a function of x, with residuals indicated by lines or squares.

**Usage**

```r
regrresidplot(x, y, resid.plot = FALSE, fit.line=TRUE, 
    lm.object = lm(y ~ x), x.name = names(lm.object$model)[2], 
    col = trellis.par.get()$plot.symbol$col, 
    col.yhat = NULL, col.fit = "gray80", col.resid = "gray40", ...) 
```

```r
panel.residSquare(x, y, yhat, resid.plot = FALSE, col = "black", ...) 
```

**Arguments**

- **x**
  - Predictor variable. Must be a vector or a single column.
- **y**
  - Response variable. Must be a vector or a single column.
- **yhat**
  - Predicted value of y based on the model in `lm.object` over the `xlim` range of the plot.
- **resid.plot**
  - Logical or character. Should the residuals from `lm.object` be plotted, and how? Default is `FALSE`. Alternatives are `TRUE` for lines and "square" for squares.
- **fit.line**
  - Logical. Should the fitted regression line from `lm.object` be plotted? Default `TRUE`.
- **lm.object**
  - Linear model object of y against some function of x. The default value is the simple linear regression of `lm(y ~ x)`.
- **x.name**
  - Name of x-variable to be used in the construction of the fitted values.
- **col**
  - Color of observed points.
- **col.yhat**
  - Color of fitted points. Default is `NULL`.
- **col.fit**
  - Color of fitted line.
- **col.resid**
  - Color of residuals, either lines or squares depending on the value of `resid.plot`.
- **...**
  - Additional arguments to the panel functions.

**Value**

`regrresidplot` returns a “trellis” object. `panel.residSquare` is a panel function with no useful returned value.
**resid.squares**

plot squared residuals in inches to match the y-dimension

**Description**

plot squared residuals in inches to match the y-dimension

**Usage**

```r
resid.squares(x, y, y.hat, resid.plot = "square", ...)
```

**Arguments**

- `x`: x values
- `y`: observed y values
- `y.hat`: predicted y values
- `resid.plot`: If "square", then plot the squared residuals. If TRUE (or anything else), then plot straight lines for the residuals.
- `...`: Other graphics arguments.

**Details**

The goal is to get real squares on the screen or paper. The trick is to play games with the aspect ratio. We find the number of inches that each vertical residual occupies. We then find the number of x-units that corresponds to, and plot a rectangle with height=height in the y-data units and with width=the number of x-units that we just calculated.
residual.plots

Description

Residual plots for a linear model. Four sets of plots are produced: (1) response against each of the predictor variables, (2) residuals against each of the predictor variables, (3) partial residuals for each predictor against that predictor ("partial residuals plots"), and (4) partial residuals against the residuals of each predictor regressed on the other predictors ("added variable plots").

Usage

residual.plots(lm.object, X=dft$x,
    layout=c(dim(X)[2],1),
    par.strip.text=list(cex=.8),
    scales.cex=.6,
    na.action=na.pass,
    y.relation="free",
    ...)
Arguments

lm.object  An object inheriting from "lm". It may be necessary for the lm.object to be constructed with arguments x=TRUE, y=TRUE.
X  The x matrix of predictor variables used in the linear model lm.object.
layout, par.strip.text  trellis or lattice arguments.
scales.cex  cex argument forwarded to the scales argument of xyplot.
na.action  A function to filter missing data. See lm.
y.relation  See relation in the discussion of the scales argument in xyplot.
...  Other arguments for xysplom or xyplot.

Value

A list of four trellis objects, one for each of the four sets of plots. The objects are named "y.X", "res.X", "pres.X", "pres.Xj". The default "printing" of the result will produce four pages of plots, one set per page. They are often easier to read when all four sets appear as separate rows on one page (this usually requires an oversize device), or two rows are printed on each of two pages.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

References


See Also

residual.plots.lattice

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)
```
Construct four sets of regression plots: $Y$ against $X$, residuals against $X$, partial residuals against $X$, partial residuals against each $X$ adjusted for all the other $X$ columns.

### Description

Construct four sets of regression plots. Response variable $Y$ against each $X_j$, residuals $e$ against each $X_j$, partial residuals plots of $e^j$ against each $X_j$, added variable plots of $e^j$ against the residuals of each $X_j$ adjusted for the other $X$ columns. The slopes shown in the panels of both bottom rows are equal to the regression coefficients.

### Usage

```r
residual.plots.lattice(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="same",
...)
```

### Arguments

- `lm.object`  
- `X`  
- `layout`  
- `par.strip.text`  
- `scales.cex`  
- `na.action`  
- `y.relation`  

#### lattice arguments. See `xyplot`.

- `scales.cex`  
  cex for the scales argument in `xyplot`.
- `na.action`  
  See `na.action`.
- `y.relation`  
  relation for the y argument to scales argument in `xyplot`.

### Value

"trellis" object.
residVSfitted

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

See Also
residual.plots, print.lattice.resids

Examples

data(longley)
longley.lm <- lm(Employed ~ ., data=longley, x=TRUE, y=TRUE)
residual.plots.lattice(longley.lm)
## Not run:
longleyResid <- tempfile("longleyResid", fileext = ".pdf")
pdf(longleyResid, height=9, width=14)
print(residual.plots.lattice(longley.lm, pch=19),
     A4.left=.0125, panel.width=list(5,"cm"))
dev.off()
## End(Not run)

draw plots of resid ~ y.hat and sqrt(abs(resid)) ~ y.hat

Description
Draw plots of resid ~ y.hat and sqrt(abs(resid)) ~ y.hat. This is a pair of lattice functions that duplicate the first and third panels of stats::plot.lm.

Usage
residVSfitted(linearmodel, groups = (e >= 0), ...)
scaleLocation(linearmodel, groups = (e >= 0), ...)

Arguments

linearmodel "lm" object.
groups This is the standard groups argument for xyplot. The default value is one symbol and color for positive residuals and a different symbol and color for negative residuals.
... Additional arguments to xyplot.

Value
"trellis" object.
Author(s)
Richard M. Heiberger <rmh@temple.edu>

Examples

```r
data(fat)
fat.lm <- lm(bodyfat ~ abdomin, data=fat)

A <- residVSfitted(fat.lm, pch=c(25,24),
                   fill=trellis.par.get("superpose.symbol")$col[1:2])
B <- scaleLocation(fat.lm, pch=c(25,24),
                   fill=trellis.par.get("superpose.symbol")$col[1:2])
BA <- c("Scale-Location"=B,
        "Residuals vs Fitted"=update(A, scales=list(y=list(at=-100, alternating=3))),
        layout=c(1,2))
BA

BAu <-
  update(BA,
         ylab=c(B$ylab, A$ylab),
         ylab.right=c(B$ylab.right, A$ylab.right),
         xlab.top=NULL,
         between=list(y=1),
         par.settings=list(layout.widths=list(ylab.right=6))
    )

C <- diagQQ(fat.lm)
D <- diagplot5new(fat.lm)

print(BAu, split=c(1,1,2,1), more=TRUE)
print(update(c("Normal Q-Q"=C), xlab.top=NULL, strip=TRUE),
      ## split=c(2,1,2,2),
      # position=c(.5, .54, 1, 1),  ## .54 is function of device and size
      # more=TRUE)
print(update(D, xlab.top=NULL,
            strip=strip.custom(factor.levels=D$xlab.top),
            par.strip.text=list(lines=1.3)),
      ## split=c(2,2,2,2),
      # position=c(.5, 0, 1, .57),  ## .57 is function of device and size
      # more=FALSE)
## the .54 and .57 work nicely with the default quartz window on Mac OS X.
```

---

**ResizeEtc**

Display multiple independent trellis objects on the same coordinated scale.
Description

This function is a wrapper for several of the functions in the latticeExtra package.

Usage

```r
ResizeEtc(c.list, 
condlevelsName, 
x.same, y.same, 
layout, 
strip=TRUE, 
strip.left=TRUE, 
strip.values, strip.left.values, 
strip.par, strip.left.par,  ## only the second is effective
        ## when both are specified
        resize.height, resize.width, 
main, 
...)
```

Arguments

c.list combination of two or more trellis objects from `c.trellis`. If c.list has names, the names will appear in the strips.

condlevelsName Name of the dimname of the items in the c.list.
x.same, y.same If TRUE, force all panels to have the same x.limits or y.limits.
layout Standard lattice layout argument.
strip, strip.left standard lattice arguments described in `barchart`.
strip.values, strip.left.values strip names for the panels. Only the second is effective when both are specified.
strip.par, strip.left.par par.strip.text. Only the second is effective when both are specified.
resize.height, resize.width h and w arguments to `resizePanels`.
main Main title for resulting combined plot.
... Other arguments to `barchart`.

Value

"trellis" object combining each of the individual plots in the c.list argument according to the specifications in the other arguments.

Author(s)

Richard M. Heiberger <rmh@temple.edu>
See Also

c.trellis, plot.likert

Examples

## see the examples in ?HH::plot.likert

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

## This is the same example as in ?HH::plot.likert
## Here, it is done with explicit use of ResizeEtc.
data(ProfChal)
tmp <- data.matrix(ProfChal[,1:5])
rownames(tmp) <- ProfChal$Question
AA <- likert(tmp[1,], box.width=unit(.4,"cm"), positive.order=TRUE)
BB <- likert(tmp[2:6,], box.width=unit(.4,"cm"), positive.order=TRUE)
CC <- likert(tmp[7:10,], box.width=unit(.4,"cm"), positive.order=TRUE)
DD <- likert(tmp[11:12,], box.width=unit(.4,"cm"), positive.order=TRUE)
EE <- likert(tmp[13:14,], box.width=unit(.4,"cm"), positive.order=TRUE)
FF <- likert(tmp[15:16,], box.width=unit(.4,"cm"), positive.order=TRUE)

BB

## print(AA, more=TRUE, split=c(1,1,3,2))
## print(BB, more=TRUE, split=c(2,1,3,2))
## print(CC, more=TRUE, split=c(3,1,3,2))
## print(DD, more=TRUE, split=c(1,2,3,2))
## print(EE, more=TRUE, split=c(2,2,3,2))
## print(FF, more=FALSE, split=c(3,2,3,2))

ResizeEtc(c.list=c(AA,BB,CC,DD,EE,FF),
          layout=c(1,6), main="Not yet good enough")

Group <- levels(ProfChal$Subtable)

ResizeEtc(c.list=c(AA,BB,CC,DD,EE,FF),
          condlevelsName='Group',
          x.same=TRUE,
          layout=c(1,6),
          strip.left.values=Group,
          strip.left.par=list(cex=.7, lines=5),
          resize.height=c(1,5,4,2,2,2)+.5,
          main=list("Is your job professionally challenging?", x=unit(.65, "npc")))
Display multiple independent trellis objects, representing likert plots, on the same coordinated scale.

Description

This is a method for ResizeEtc intended for use with "likert" plots that allows positive values on the negative side of the axis.

Usage

```r
## S3 method for class 'likertPlot'
ResizeEtc(c.list,
  x,
  x.pl.nonames,
  horizontal,
  ...
)
```

Arguments

- `c.list` combination of two or more trellis objects from `c.trellis`. If `c.list` has names, the names will appear in the strips.
- `x` List of two-dimensional objects with the same columns. See `plot.likert.list` for details.
- `x.pl.nonames` List of "likert" objects corresponding to the items in argument `x`. The items in `x.pl.nonames` are unnamed.
- `horizontal` Standard argument for `barchart`.
- `...` Other arguments to `ResizeEtc`.

Value

The result is a "trellis" object. It is essentially the same object returned by `ResizeEtc` with possibly adjusted x tick-labels to put positive labels on the negative axis. If `horizontal==FALSE`, then the possible adjusted labels are the y tick-labels.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

- `ResizeEtc`, `likert`
Description

Row and columns percents.

Usage

rowPcts(x, ...)  
colPcts(x, ...)

Arguments

x    numerical matrix
...

Value

Calculate percents by row or column. The rowSums or colSums are stored in the Sums attribute of the result.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

rowSums

Examples

tmp <- matrix(1:12, 3, 4,  
dimnames=list(c("A","B","C"),  
c(letters[4:7])))
tmp
rowPcts(tmp)
colPcts(tmp)
### Description

Time series plot.

### Usage

```r
seqplot(xts, ...)  
```

#### Default S3 method:

```r
seqplot(xts,  
  pch.seq=letters,  
  groups=as.numeric(cycle(xts)),  
  a=NULL, b=NULL, h=NULL, v=NULL,  
  ylab=deparse(substitute(xts)),  
  xlab="Time",  
  lwd=1, lty=c(1,3),  
  type="b",  
  col=trellis.par.get("superpose.symbol")$col,  
  col.line="gray60",  
  ...)  
```

#### S3 method for class 'ts'

```r
seqplot(xts, pch.seq=letters, groups=as.numeric(cycle(xts)),  
  x.at=pretty(time(xts)[groups==min(groups)]),  
  x.labels,  
  ylab=deparse(substitute(xts)),  
  ...)  
```
Arguments

- `xts` Time series
- `pch.seq` sequence of `pch` characters for use with the time series. The characters repeat over the cycle of the series.
- `groups` Numeric vector used to choose the plotting characters over cycles.
- `a, b, h, v` Arguments to `panel.abline`.
- `x.at, x.labels` shortcut for `scales=list(x=list(at=x.at, labels=x.labels))`
- `col` Color of dots in sequence plot. The default is to make the choose a number of colors to match the frequency of the time series `xts`.
- `col.line` Color of connecting lines. The default is "gray60".
- `...` Additional arguments to `xyplot`.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

tsacfplots

Examples

```r
seqplot(co2)
```

Description

`seqplot.forecast` is `seqplot` with confidence bands for the forecast region.
Usage

```r
seqplotForecast(xts, forecast, multiplier = 1.96,
series = deparse(substitute(observed)), ylim,
CI.percent=round((1-2*(1-pnorm(multiplier)))*100,2),
main = paste(
    series, " with forecast + ",
CI.percent, ", % CI", sep=""),
xlab=NULL, ylab=NULL,
...) ## x.at, xlim
```

Arguments

- **xts**: This is the observed series
- **forecast**: forecast values based on the model
- **multiplier**: Half-width of confidence interval in standard normal units. Defaults to 1.96.
- **CI.percent**: Width of confidence band. Defaults to the standard normal, two-sided value associated with the multiplier (95 percent for the default multiplier=1.96).
- **series**: Name of time series will be used to construct the main title for the plot.
- **ylim, xlab, ylab, main**
  - standard trellis parameters
  - additional arguments to `xyplot`.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

- `seqplot`

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>
Functions based on \texttt{strip.default} for use with the \texttt{useOuterScales} function.

\section*{Description}
Functions based on \texttt{strip.default} for use with the \texttt{useOuterScales} function. See \texttt{useOuterScales} for more information.

\section*{Usage}
\begin{verbatim}
strip.useOuterStrips.first(which.given, which.panel, var.name, ...)
strip.useOuterStrips.last(which.given, which.panel, var.name, ...)
strip.left.useOuterStrips(which.given, which.panel, var.name, ...)
strip.top2(which.given, which.panel, var.name, ...)
strip.top1(which.given, which.panel, var.name, ...)
strip.left2(which.given, which.panel, var.name, ...)
strip.left1(which.given, which.panel, var.name, ...)
\end{verbatim}

\section*{Arguments}
\begin{verbatim}
which.given, which.panel, var.name, ...
\end{verbatim}
See \texttt{strip.default}.

\section*{Details}
The appropriate function is chosen by specifying arguments to \texttt{useOuterScales}.
\begin{itemize}
\item \texttt{strip.useOuterStrips.first} places strip labels at the top of the first row of lattice panels. Used when \texttt{as.table==TRUE}.
\item \texttt{strip.useOuterStrips.last} places strip labels at the top of the first row of lattice panels. Used when \texttt{as.table==FALSE}.
\item \texttt{strip.left.useOuterStrips} places strip labels at the left of the first column of lattice panels.
\item \texttt{strip.top2} places row strip labels at the top of each panel.
\item \texttt{strip.top1} places column strip labels at the top of each panel.
\item \texttt{strip.left2} places row strip labels at the left of each panel.
\item \texttt{strip.left1} places column strip labels at the left of each panel.
\end{itemize}

\section*{Value}
See \texttt{strip.default}.

\section*{Author(s)}
Richard M. Heiberger \texttt{<rmh@temple.edu>}
strip.xysplom

See Also

useOuterScales

Examples

## See examples in ?useOuterScales

---

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

sufficient(x, yname = dimnames(x)[[2]][[1]], factor.names.keep = dimnames(x)[[2]][-c(1, 2)])

Arguments

x data.frame containing a continuous variable and two factors.

yname Character name of response variable.

factor.names.keep Character vector containing the names of two factors in the x data.frame.

Value

Data.frame containing five columns and as many rows as are implied by the crossing of the two factors. Each row contains the mean in a column with the name yname and its factor values in columns named with the name in factor.names.keep. The standard deviation of the observations in the group are in the column "sd" and the number of observations in the group is in the column "nobs".

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

intxplot
summary.arma.loop

summary.arma.loop

summary and print and subscript methods for tsdiagplot and related objects.

Description

summary and print and subscript methods for tsdiagplot and related objects.

Usage

## S3 method for class 'arma.loop'
summary(object, ...)
## S3 method for class 'arma.loop.list'
summary(object, ...)
## S3 method for class 'arma.loop'
print(x, ...)
## S3 method for class 'arma.loop.list'
print(x, ...)
## S3 method for class 'tsacfplots'
print(x,
    ts.pos=c(.00, .00, .70, 1.00),
    acf.pos=c(.65, .10, 1.00, .90),
    ...,
    portrait=FALSE,
    ts.pos.portrait=c(0, .3, 1, 1),
    acf.pos.portrait=c(.1, 0, .9, .35))
## S3 method for class 'arma.loop'
x[...], drop = TRUE]
## S3 method for class 'diag.arma.loop'
x[...], drop = TRUE]

Arguments

x, object object to be summarized or printed or subscripted.
ts.pos, acf.pos, ts.pos.portrait, acf.pos.portrait
    Default positions for print.trellis
portrait logical. If FALSE, arrange the panels for a landscape orientation. If TRUE, arrange
    the panels for a portrait orientation.
... additional arguments
drop See Extract.

Author(s)

Richard M. Heiberger (rmh@temple.edu)
ToBW.likert

Change colors in a likert plot to shades of Black and White.

Description

Change colors in a likert plot to shades of Black and White. This function is tailored for a likert plot, an example of a "trellis" object. likert is based on panel.bwplot. There are other places in the structure of a more general "trellis" object where colors are stored. The specifics for this plot is (1) that the colors for negative values in the plot are in reverse order and (2) the color for a neutral-position panel appears on both the positive and negative side. The default values are for three items on the negative side, two on the positive side, and no neutral. See the examples for an example with a neutral.

Usage

ToBW.likert(x,
   colLegendOrder=c("gray85", "gray20", "gray60", "gray75", "gray45"),
   ## ^Ask    Refu    ^Imp | Impt    Essn
   ##
   colBarchartOrder=colLegendOrder[c(3,2,1,4,5)],
   ## ^Imp    Refu    ^Ask | Impt    Essn
   columns=5)
   ## negative colors are in reverse order in the BarchartOrder

Arguments

x "trellis" object, specifically one constructed by the likert function.

colLegendOrder Revised value of x$legend$bottom$args$rect$col

colBarchartOrder Revised value of both x$panel.args.common$col x$panel.args.common$border.

columns Revised value of x$legend$bottom$args$columns

Value

"trellis" object, identical to the input object except for the colors.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likert
Examples

tmp <- array(1:20, c(4, 5),
             list(letters[1:4],
                  c("NotAsked","VeryNegative","Negative","Positive","VeryPositive")))
tmp

Ltmp <- likert(tmp, ReferenceZero=3.5, col=c("gray85", likertColor(4)), as.percent=TRUE)
Ltmp

ToBW.likert(Ltmp)

## with neutral

tmp2 <- array(1:20, c(4, 5),
              list(letters[1:4],
                   c("VeryNegative","Negative","Neutral","Positive","VeryPositive")))
tmp2

Ltmp2 <- likert(tmp2, ReferenceZero=3, col=likertColor(5),
               as.percent=TRUE, main="Neutral")
Ltmp2

ToBW.likert(Ltmp2,
            colLegendOrder=c("gray20", "gray60", "gray85", "gray75", "gray45"),
            colBarchartOrder=c("gray85", "gray60", "gray20", "gray85", "gray75", "gray45")
            )
            update(main="Wrong way to handle neutral",
                   ToBW.likert(Ltmp2,
                   colLegendOrder=c("gray20", "gray60", "gray85", "gray75", "gray45")))


toCQxR

Reshape a 3-way array to a 2-way data.frame that can be used with a trellis conditioning formula to get the three-way behavior. Used with likertWeighted().

Description

Reshape a 3-way array to a 2-way data.frame that can be used with a trellis conditioning formula to get the three-way behavior. Used with likertWeighted().

Usage

unCQxR(x, C = 1, R = 2, Q = 3)
tsacfplots

Arguments

x
Three-way array, with dimensions "Classification", "Responses", "Questions" in some order.

C, R, Q
Integers, one each of 1,2,3; positions of the three dimensions.

Value

Data.frame with CQ rows and Q + N columns, where N is either 1 or 2 for the number of condition variables in the formula for likertWeighted.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

likertWeighted

Examples

tmp3 <- array(1:40, c(4,5,2), list(LETTERS[1:4], LETTERS[5:9], LETTERS[10:11]))
tmp3
toCQxR(tmp3)

tzacfplots

Coordinated time series and ACF and PCF plots.

Description

Coordinated time series and ACF and PCF plots.

Usage

tzacfplots(x,
    ylab=deparse(substitute(x)),
    x.name=ylab[[1]],
    main=paste("Series:", x.name),
    lag.at=NULL,
    lag.max=NULL,
    lag.units=NULL,
    lag.0=TRUE,
    ...
)

tacf.plot(x,
    ylab=NULL,
    series=deparse(substitute(x)),
    ...)
main=paste("ACF and PACF:", series),
lag.max,
lag.units=frequency(x),
lag.at=pretty(apacf$lag),
lag.labels=lag.at*lag.units,
    lag.0=TRUE,
    strip=TRUE, strip.left=FALSE,
...

Arguments

x         time series
ylab, main standard trellis arguments.
x.name, series Character string, name for the time series.
lag.at    Location of ticks for the acf and pacf plots.
lag.labels Labels for ticks for the acf and pacf plots.
lag.max   Maximum lag used in the acf and pacf plots.
lag.units Units for time series, defaults to frequency(x)
lag.0     Logical. If TRUE, then plot the correlation (identically 1) at lag=0. If FALSE, do not plot the correlation at lag=0.
strip, strip.left Standard lattice arguments described in xyplot.
...    Additional arguments to seqplot for tsacfplots. Additional arguments to strip.default for acf.pacf.plot.

Details

The acf and pacf plots are scaled identically.

Value

"tsacfplots" object containing two "trellis" objects.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

See Also

seqplot

Examples

tsacfplots(co2)
acf.pacf.plot(co2)
**tsdiagplot**  
*Times series diagnostic plots for a structured set of ARIMA models.*

**Description**

Times series diagnostic plots for a structured set of ARIMA models.

**Usage**

```r
  tsdiagplot(x,
    p.max=2, q.max=p.max,
    model=c(p.max, 0, q.max), ## S-Plus
    order=c(p.max, 0, q.max), ## R
    lag.max=36, gof.lag=lag.max,
    armas=arma.loop(x, order=order,
        series=deparse(substitute(x)), ...),
    diags=diag.arma.loop(armas, x,
        lag.max=lag.max,
        gof.lag=gof.lag),
    ts.diag=rearrange.diag.arma.loop(diags),
    lag.units=ts.diag$tspar["frequency"],
    lag.lim=range(pretty(ts.diag$acf$lag))*lag.units,
    lag.x.at=pretty(ts.diag$acf$lag)*lag.units,
    lag.x.labels={tmp <- lag.x.at
        tmp[as.integer(tmp)!=tmp] <- ""
        tmp},
    lag.0=TRUE,
    main, lwd=0,
    ...)
```

```r
  acfplot(rdal, type="acf",
    main=paste("ACF of std.resid:", rdal$series,
        " model:", rdal$model),
    lag.units=rdal$tspar["frequency"],
    lag.lim=range(pretty(rdal[[type]]$lag)*lag.units),
    lag.x.at=pretty(rdal[[type]]$lag)*lag.units,
    lag.x.labels={tmp <- lag.x.at
        tmp[as.integer(tmp)!=tmp] <- ""
        tmp},
    lag.0=TRUE,
    xlim=xlim.function(lag.lim/lag.units),
    ...)
```

```r
  aicsigplot(z, z.name=deparse(substitute(z)), series.name="ts",
    model=NULL,
    xlab="", ylab=z.name,
    main=paste(z.name, series.name, model),
    ...)
```
tsdiagplot

layout=c(1,2), between=list(x=1,y=1), ...

residplot(rdal,
  main=paste("std.resid: " , rdal$series,
     " model: " , rdal$model),
     ...
)

gofplot(rdal,
  main=paste("P-value for gof: " , rdal$series,
     " model: " , rdal$model),
  lag.units=rdal$tspar["frequency"],
  lag.lim=range(pretty(rdal$gof$lag)*lag.units),
  lag.x.at=pretty(rdal$gof$lag)*lag.units,
  lag.x.labels={tmp <- lag.x.at
      tmp[as.integer(tmp)!=tmp] <- 
      xlim=xlim.function(lag.lim/lag.units),
  pch=16, ...)

Arguments

x Time series vector.
p.max, q.max Maximum number of AR and MA arguments to use in the series of ARIMA models.
model A valid S-Plus model for arima.mle.
order A valid R order for arima. The additional argument seasonal may also be used.
lag.max Maximum lag for the acf and pacf plots.
gof.lag Maximum lag for the gof plots.
armas An arma.loop object.
diags An diag.arma.loop object.
ts.diag, rdal A list constructed as a rearranged diag.arma.loop object.
lag.units Units for time series, defaults to frequency(x)
lag.lim scaling for xlim in acf and pacf plots.
lag.x.at, lag.x.labels Location of ticks and labels for the acf and pacf plots.
lag.0 Logical. If TRUE, then plot the correlation (identically 1) at lag=0. If FALSE, do not plot the correlation at lag=0.
type "acf" or "pacf"
z A matrix constructed as the aic or sigma2 component of the summary of a arma.loop object.
z.name "aic" or "sigma2"
series.name Character string describing the time series.
tsdiagplot

xlab, ylab, layout, between, pch, xlim, main, lwd

Standard trellis arguments.

... Additional arguments. tsdiagplot sends them to arima or arima.mle. acfplot, aicsigplot residplot, and gofplot send them to xyplot.

Value

tsdiagplot returns a "tsdiagplot" object which is a list of "trellis" objects. It is printed with its own print method.
The other functions return "trellis" objects.

Author(s)

Richard M. Heiberger (rmh@temple.edu)

References


See Also

tsacfplots, arma.loop

Examples

data(tser.mystery.X)
X <- tser.mystery.X

X.dataplot <- tsacfplots(X, lwd=1, pch.seq=16, cex=.7)
X.dataplot

X.loop <- if.R(
  s=
    arma.loop(X, model=list(order=c(2,0,2)))
  ,r=
    arma.loop(X, order=c(2,0,2))
)

X.dal <- diag.arma.loop(X.loop, x=X)
X.diag <- rearrange.diag.arma.loop(X.dal)
X.diagplot <- tsdiagplot(armas=X.loop, ts.diag=X.diag, lwd=1)
X.diagplot

X.loop
X.loop[["1","1"]]


**Description**

Update a multi-panel "trellis" object so that scales for axes are displayed only on the bottom and left boundaries when printed, instead of in every panel as is usual. This function succeeds even when xlim across columns and ylim across rows are not identical. Multiple options are available for strip labels. The default for strip labels is similar to `useOuterStrips`. Additional options include outerStrips for each panel and interchanged row and column strip locations. This is only meaningful when there are exactly two conditioning variables.

**Usage**

```r
useOuterScales(x,
    axis.xlab.padding=4,
    ylab.axis.padding=3,
    strip,
    strip.left,
    layout.widths.strip.left=.5,
    layout.heights.strip=.5,
    x.ticks=is.numeric(x$x.limits),
    y.ticks= is.numeric(x$y.limits) +
    if (!missing(strip.left) && !strip.left) 0
      else 2.5,
    inner=FALSE,
    interchangeRC=FALSE)
```

**Arguments**

- `x` An object of class "trellis"
  - `ylab.axis.padding`, `axis.xlab.padding`, `layout.heights.strip`, `layout.widths.strip.left` These values are passed to the `par.settings`:
    - `layout.widths=list(ylab.axis.padding=ylab.axis.padding,
                              strip.left=layout.widths.strip.left),`
    - `layout.heights=list(axis.xlab.padding=axis.xlab.padding,
                             strip=layout.heights.strip)`
  - `x.ticks=is.numeric(x$x.limits),` `y.ticks= is.numeric(x$y.limits) +` `if (!missing(strip.left) && !strip.left) 0
    else 2.5,
  - `inner=FALSE,` `interchangeRC=FALSE)`
- `x$as.table`
- `strip`, `strip.left`

`useOuterScales` controls the strip labels by assigning appropriate functions for these two arguments. The functions used by `useOuterScales` are described in `strip.useOuterStrips.first`. `useOuterScales` uses the values of its `strip`, `strip.left`, `inner`, `interchangeRC`, and `x$as.table` arguments to determine
useOuterScales

which functions to assign. The default values place the columns strip labels at
the top of the top row of panels and the row strip labels at the left of the left
column of panels. See the Examples section for the full set of possibilities that
are provided.

x.ticks, y.ticks

x.ticks is used as the ticks argument to panel.axis for the "bottom" axis.
y.ticks is used as the ticks argument to panel.axis for the "left" axis.
y.ticks needs to be larger when the left strip is present because the tick and
label are partially overwritten by the left strip. When the left.strip=FALSE,
then we need to make the y.ticks smaller.

inner

Logical with default FALSE, meaning that the strip labels are displayed only on
the top row and left column of the array of panels. When TRUE, the strip labels
are displayed on the top and left of every panel.

interchangeRC

Logical with default FALSE. When TRUE, the column labels appear on the left
strip of the panels, and the row labels appear on the top of the panels. TRUE is
only meaningful when inner=TRUE.

Details

useOuterScales modifies a "trellis" object with length(dim(x)) == 2 so that when plotted,
scales appear on only the top and left panels of the array of panels. Strips appear as specified, by
default on the top and left boundaries of the panel layout.

If the original "trellis" object x includes non-default strip and strip.left arguments, they
will be ignored. To provide customized strip behaviour, specify the custom strip functions directly
as arguments to useOuterStrips.

Value

An object of class "trellis"; essentially the same as x, but with certain properties modified.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

See Also

useOuterStrips, strip.default

Examples

OuterScalesData <- data.frame(y=1:16,
    AA=rep(factor(letters[1:8]), 2),
    BB=rep(factor(LETTERS[12:13]), each=8),
    CC=rep(factor(rep(LETTERS[9:11], times=c(3,1,4))), 2))

OuterScalesData

BC0 <- barchart(AA ~ y | BB * CC, data=OuterScalesData,
    origin=0,
    scales=list(x=list(limits=c(0,16.5)),


y=list(relation="free"),
    between=list(x=1, y=1),
    main="0. barchart")

## Not run:
BC0

## End(Not run)
BC1 <- update(
    resizePanels(BC0, h=c(3,1,4)),
    main="1. resizePanels")
BC1

BC2 <- update(
    useOuterStrips(BC1),
    main="2. useOuterStrips") ## package:latticeExtra
BC2

BC3 <- update(
    useOuterScales(BC1),
    main="3. useOuterScales")
BC3

## Not run:
BC4 <- update(
    useOuterScales(BC1),
    ylab="ABC",
    main="4. useOuterScales, ylab")
BC4

BC5 <- update(
    useOuterScales(update(BC1, as.table=TRUE)),
    main="5. useOuterScales, as.table")
BC5

try(useOuterScales(BC1, interchangeRC=TRUE)) ## incompatible options

## Not run:
BC6 <- update(
    useOuterScales(BC1, inner=TRUE),
    main="6. useOuterScales, inner")
BC6

## Not run:
BC7 <- update(
    useOuterScales(BC1, inner=TRUE, interchangeRC=TRUE),
    main="7. useOuterScales, inner, interchangeRC")
BC7

BC8 <- update(
    useOuterScales(BC1, strip=FALSE),
    xlab.top=c("L","M"),
    main="8. useOuterScales, strip=FALSE, xlab.top")
BC8

BC9 <- update(
  useOuterScales(BC1, strip=strip.default),
  main="9. useOuterScales, strip=strip.default")
BC9

try(print(useOuterScales(BC1, strip=date))) ## date is not a valid strip function

BC10 <- update(
  useOuterScales(BC1, strip.left=FALSE),
  ylab=c("I","J","K"),
  main="10. useOuterScales, strip.left=FALSE, ylab")
BC10

BC11 <- update(
  useOuterScales(BC1, strip.left=strip.default),
  main="11. useOuterScales, strip.left=strip.default")
BC11

try(print(useOuterScales(BC1, strip.left=date))) ## date is not a valid strip function

BC12 <- update(
  useOuterScales(BC1,
    inner=TRUE, interchangeRC=TRUE, strip.left=FALSE),
  xlab.top=c("L","M"),
  main="12. useOuterScales, inner, \n interchangeRC=TRUE, strip.left=FALSE, \n xlab.top, strip.background",
  par.settings=list(strip.background=list(col="gray98")))
BC12

BC13 <- update(
  useOuterScales(update(BC1, as.table=TRUE),
    inner=TRUE, interchangeRC=TRUE, strip.left=FALSE),
  xlab.top=c("L","M"),
  main="13. useOuterScales, inner, \n interchangeRC=TRUE, strip.left=FALSE, \n xlab.top, as.table")
BC13

BC14 <- update(
  useOuterScales(BC1,
    inner=TRUE, strip=FALSE, interchangeRC=TRUE),
  ylab=list(c("I","J","K"), rot=0),
  main="14. useOuterScales, inner, \n strip=FALSE, interchangeRC, \n ylab")
BC14

BC15 <- update(
  useOuterScales(BC1,
    strip=FALSE, strip.left=FALSE),
  xlab.top=c("L","M"), ylab=list(c("I","J","K"), rot=0),
  main="15. useOuterScales, strip=FALSE, strip.left=FALSE, \n xlab, ylab")
BC15

## End(Not run)
## Not run: ## display 16 options for strip labels with outerScales

```
useOuterScales16 <- tempfile("useOuterScales16", fileext = "pdf")
pdf(useOuterScales16, height=16, width=21)
print(BC0, split=c(1,1,4,4), more=TRUE)
print(BC1, split=c(2,1,4,4), more=TRUE)
print(BC2, split=c(3,1,4,4), more=TRUE)
print(BC3, split=c(4,1,4,4), more=TRUE)
print(BC4, split=c(1,2,4,4), more=TRUE)
print(BC5, split=c(2,2,4,4), more=TRUE)
print(BC6, split=c(3,2,4,4), more=TRUE)
print(BC7, split=c(4,2,4,4), more=TRUE)
print(BC8, split=c(1,3,4,4), more=TRUE)
print(BC9, split=c(2,3,4,4), more=TRUE)
print(BC10, split=c(3,3,4,4), more=TRUE)
print(BC11, split=c(4,3,4,4), more=TRUE)
print(BC12, split=c(1,4,4,4), more=TRUE)
print(BC13, split=c(2,4,4,4), more=TRUE)
print(BC14, split=c(3,4,4,4), more=TRUE)
print(BC15, split=c(4,4,4,4), more=FALSE)
dev.off()
```

## End(Not run)

## Not run:
## Verify y.ticks default value depends on
## is.numeric(x$y.limits).
## and on whether strip.left=FALSE

```
CB0 <- barchart(y ~ AA | CC * BB, data=OuterScalesData,
origin=0,
scales=list(y=list(limits=c(0,16.5)),
        x=list(relation="free"),
        between=list(x=1, y=1),
        main="CB0. barchart")
CB0

CB1 <- update(
    resizePanels(CB0, w=c(3,1,4)),
    main="CB1. resizePanels")
CB1

CB2 <- update(
    useOuterStrips(CB1),
    main="CB2. useOuterStrips") ## package:latticeExtra
CB2

CB3 <- update(
    useOuterScales(CB1),
    main="CB3. useOuterScales, y.limits is numeric")
CB3
```
useOuterStripsT2L1

Three-factor generalization of latticeExtra::useOuterStrips

Description

Three-factor generalization of latticeExtra::useOuterStrips

Usage

useOuterStripsT2L1(x, ..., strip.height=.4, strip.names=c(TRUE, TRUE))

Arguments

x

A lattice object with dim(x)==3.

...  

Additional arguments to be forwarded to the strip.default function.

strip.height

Height of each the strip for each factor. The number of factors in the top and left strips may not be the same. This argument is multiplied by the number of factors in each location and sent on to the lattice par.settings argument for the layout.widths$strip.left and layout.heights$strip components.

strip.names

See strip.default.
Value

A trellis object with two factors in the top strip and 1 factor in the strip.left.

Author(s)

Richard M. Heiberger <rmh@temple.edu>

Examples

tmp <- data.frame(A=rep(factor(letters[1:2]), each=12),
                  B=rep(factor(letters[3:5]), each=4, times=2),
                  C=rep(factor(letters[6:9]), times=6),
                  x=1:24,
                  y=1:24)

F <- xyplot(y ~ x | B*A*C, data=tmp,
            panel=function(x, y, labels, ...) {
              panel.text(x, y, matrix(1:24, 6, 4, byrow=TRUE)[panel.number()], ...)
            },
            layout=c(6, 4), between=list(x=c(.5, .5, 1.5), y=1))

useOuterStripsT2L1(F)

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

vif(xx, ...)

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

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

## S3 method for class 'lm'
vif(xx, na.action = na.exclude, ...) ## xx is a "lm" object computed with x=TRUE
Arguments

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

Details

A simple diagnostic of collinearity is the *variance inflation factor*, VIF one for each regression coefficient (other than the intercept). Since the condition of collinearity involves the predictors but not the response, this measure is a function of the $X$’s but not of $Y$. The VIF for predictor $i$ is $1/(1-R^2_i)$, 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

data(usair)

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

usair.lm <- lm(lnSO2 ~ temp + lnmfg + wind + precip, data=usair, x=TRUE)
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")

---

**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`.
- `...` 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`, `case.lm`.

Examples

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

usair.lm <- lm(lnSO2 ~ temp + lnmfg + wind + precip, data=usair)
X.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

`xysplom(x, ...)`

## S3 method for class 'formula'
`xysplom(x, data=NULL, 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,`
# Arguments

**x**
- In the "formula" method, a formula. In the "default" method, a data.frame. Any variables that are used in a formula with + should be numeric. Factors are not rejected, but their levels will be combined strangely.

**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`.

---

```r
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)
```
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

```r
## xysplom syntax options

tmp <- data.frame(y=rnorm(12), x=rnorm(12), z=rnorm(12), w=rnorm(12),
g=factor(rep(1:2,c(6,6))))
tmp2 <- tmp[,1:4]

xysplom(y + w ~ x , data=tmp, corr=TRUE, beta=TRUE, cartesian=FALSE, layout=c(1,2))

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

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

xysplom(w + y ~ x + z, data=tmp)
xysplom(w + y ~ x + z | g, data=tmp, layout=c(2,4))
xysplom(w + y ~ x + z | g, data=tmp, cartesian=FALSE)
```
## Not run:

xyplot in R has many similar capabilities with xysplom

if.R(r=
    xyplot(w + z ~ x + y, data=tmp, outer=TRUE)
    ,s=
        ()
)

## End(Not run)
Index

* NA
  diag.maybe.null, 63
* algebra
  orthog.complete, 193
* aplot
  F.curve, 72
  norm.curve, 169
  panel.interaction2wt, 208
  perspPlane, 217
  strip.useOuterStrips.first, 262
  useOuterScales, 273
* classes
  ancova-class, 28
  positioned-class, 230
* color
  col.hh, 54
* confidence
  CIplot, 51
* datasets
  col3x2, 55
  datasets, 60
  Discrete4, 66
* data
  HH-package, 5
* design
  glhtWithMCP, 993, 75
  HH-package, 5
  interaction2wt, 88
  lmatPairwise, 140
  mmc, 147
  panel.interaction2wt, 208
* device
  export.eps, 70
  GSremove, 78
* distribution
  F.curve, 72
  norm.curve, 169
  pdiscunif, 216
* dpot
  ancova, 25
  ancovaplot, 29
  as.multicomp, 44
  as.vector.trellis, 47
  axis.i2wt, 48
  combineLimits.trellisvector, 56
  cplx, 60
  grid.yaxis.hh, 77
  hovBF, 84
  interaction.positioned, 87
  intxplot, 93
  ladder, 96
  latticeresids, 103
  legendGrob2wt, 104
  lmatRows, 141
  lmplot, 142
  matrix.trellis, 144
  multicomp.order, 165
  multicomp.reverse, 167
  OneWayVarPlot, 192
  panel.acf, 196
  panel.bwplot.intermediate.hh, 197
  panel.bwplot.superpose, 198
  panel.bwplot, 201
  panel.ci.plot, 204
  panel.dotplot.tb, 206
  panel.likert, 211
  panel.pairs.hh, 213
  panel.xysplom, 214
  plot.multicomp, 223
  position, 226
  print.latticeresids, 231
  push.vp.hh, 238
  rbind.trellis, 241
  residual.plots.trellis, 252
  strip.background0, 261
  strip.xysplom, 263
  sufficient, 264
  useOuterStripsT2L1, 278
INDEX

* dynamic
  * CIplot, 51
* hplot
  * ae.dotplot, 12
    AEdotplot, 17
    AEdotplot.data.frame, 20
  * ancova, 25
  * ancovaplot, 29
  * arma.loop, 37
  * as.likert, 39
  * bivariateNormal, 48
  * ci.plot, 49
  * CIplot, 51
  * diagplot5new, 64
  * diagQQ, 65
  * emptyMainLeftAxisLeftStripBottomLegend, 69
  * extra, 71
  * F.curve, 72
  * HH-package, 5
  * interaction2wt, 88
  * ladder, 96
  * likert, 104
  * likertColor, 120
  * likertMosaic, 123
  * LikertPercentCountColumns, 128
  * lm.case, 136
  * mmc, 147
  * mmc.mean, 155
  * mmcAspect, 159
  * mmcisomeans, 159
  * mmcplot, 162
  * mmcPruneIsomeans, 164
  * norm.curve, 169
  * NormalAndTPlot, 175
  * NormalAndTPower, 182
  * normalApproxBinomial, 184
  * NTplot, 186
  * OddsRatio, 190
  * panel.axis.right, 196
  * panel.cartesian, 202
  * panel.confintMMC, 205
  * panel.isomeans, 210
  * partial.corr, 214
  * plot.hov, 218
  * plot.mmc.multicomp, 220
  * print.tsdiagplot, 233
  * print.TwoTrellisColumns, 234
  * pyramidLikert, 239
  * regresidplot, 248
  * residual.plots, 250
  * residVSfitted, 253
  * ResizeEtc, 254
  * ResizeEtc.likertPlot, 257
  * seqplot, 259
  * seqplot.forecastPlot, 260
  * tsacfplots, 268
  * tsdiagplot, 270
  * xysplom, 282
* htest
  * ae.dotplot, 12
    AEdotplot, 17
    AEdotplot.data.frame, 20
  * aovSufficient, 34
  * glhtWithMCP, 99, 3
  * HH-package, 5
  * lmatPairwise, 140
  * mcalinfct, 146
  * mmc, 147
  * mmc.mean, 155
  * OddsRatio, 190
* manip
  * ladder, 96
* math
  * logit, 143
* misc
  * dchisq.intermediate, 61
  * defunct, 62
  * hhpdf, 80
  * HHscriptnames, 81
* models
  * ancova, 25
  * ancovaplot, 29
  * anova.ancovaplot, 32
  * anovaMean, 33
  * do.formula.trellis.xysplom, 67
  * hov, 82
  * plot.hov, 218
  * regr1.plot, 244
  * regr2.plot, 246
  * resid.squares, 249
* package
  * HH-package, 5
* print
  * as.matrix.listOfNamedMatrices, 41
  * summary.arma.loop, 265
**regression**
- anova, 25
- ancovaplot, 29
- ci.plot, 49
- cp.calc, 57
- HH.regsubsets, 78
- interaction.positioned, 87
- interval, 92
- lm.case, 136
- lm.regsubsets, 139
- regr1.plot, 244
- regr2.plot, 246
- resid.squares, 249
- residual.plots, 250
- vif, 279
- X.residuals, 281

**shiny**
- AE.dotplot, 12
- bivariateNormal, 48
- CIplot, 51
- HH-package, 5
- likert, 104
- NTplot, 186
- pyramidLikert, 239

**ts**
- arima.diag.hh, 36
- arma.loop, 37
- extra, 71
- gof.calculation, 76
- HH-package, 5
- npar arma, 185

**univar**
- sufficient, 264

**utilities**
- if.R, 85
- objip, 189
- arima.info.names not.ordered(extra), 71
- [.arma.loop(summary arma.loop), 265
- [.cp.object(cp.calc), 57
- [.diag arma.loop(summary arma.loop), 265
- [.listOfNamedMatrices (as.matrix.listOfNamedMatrices), 41
- [.mmc.multicomp(mmc), 147
- [.positioned(position), 226
- abbreviate, 42

abc (datasets), 60
abrasion (datasets), 60
acacia (datasets), 60
acf.pacf.plot (tsacfplots), 268
acfplot (tsdiagplot), 270
addNA, 42
AE.dotplot (ae.dotplot), 12
ae.dotplot, 12
aeanonym (datasets), 60
AEdataset (datasets), 60
AEdotplot, 12, 15, 17, 20, 21, 24
AEdotplot.AElogrelrisk
(AEdotplot.data.frame), 20
AEdotplot.AEtable
(AEdotplot.data.frame), 20
AEdotplot.data.frame, 17–19, 20
AElorlogrelrisk (AEdotplot.data.frame), 20
AEmatchSortorder
(AEdotplot.data.frame), 20
aeReshapeToLong (ae.dotplot), 12
aicsigplot (tsdiagplot), 270
analysis of covariance (ancova), 25
ancova, 25, 28, 29, 32
ancova-class, 28
ancovaplot, 12, 27, 29, 33
animal (datasets), 60
anneal (datasets), 60
anova.ancova (ancova), 25
anova.ancovaplot, 32
anova.lm, 34
anova.mean (defunct), 62
anovaMean, 33
antilogit (logit), 143
antiodds (logit), 143
aov, 27, 35, 83, 219
aov.ancovaplot (anova.ancovaplot), 32
aov.sufficient (defunct), 62
aovStatement (anova.ancovaplot), 32
aovStatementAndAnova
(anova.ancovaplot), 32
aovSufficient, 34
aperm, 241
aperm.trellis (rbind.trellis), 241
apple (datasets), 60
ara (datasets), 60
arima, 38, 271
arima.diag.hh, 36
arima.model (extra), 71
arma.loop, 37, 72, 266, 272
as.character.arima.model(extra), 71
as.data.frame.listOfNamedMatrices
  (as.matrix.listOfNamedMatrices), 41
as.glht, 166
as.glht(as.multicomp), 44
as.likert, 39, 113
as.likertDataFrame(as.matrix.listOfNamedMatrices), 41
as.listOfNamedMatrices(as.matrix.listOfNamedMatrices), 41
as.matrix, 145
as.matrix.listOfNamedMatrices, 41, 113
as.matrix.trellis(matrix.trellis), 144
as.MatrixList(as.matrix.listOfNamedMatrices), 41
as.multicomp, 44, 151
as.numeric.positioned(position), 226
as.position(position), 226
as.pyramidLikert(pyramidLikert), 239
as.rts(extra), 71
as.TwoTrellisColumns5, 128, 129
as.TwoTrellisColumns5
  (print.TwoTrellisColumns), 234
as.vector.trellis, 47
AudiencePercent(datasets), 60
axis.default, 48, 196, 197
axis.i2wt, 48
axis.RightAdjustRight
  (panel.axis.right), 196
balance(datasets), 60
barchart, 40, 109, 110, 113, 235, 255, 257
barleyp(datasets), 60
batch(datasets), 60
bean(datasets), 60
beta curve (NormalAndTPower), 182
birthweight(datasets), 60
bivariateNormal, 48
blood(datasets), 60
blyth(datasets), 60
breast(datasets), 60
brewer.pal.likert(likertColor), 120
budworm(datasets), 60
byss(datasets), 60
c.AEdotplot(AEdotplot.data.frame), 20
c.trellis, 69, 255–257
c3c4(datasets), 60
case(lm.case), 136
case.lm, 12, 282
catalystm(datasets), 60
cbind, 242
cbind.trellis(rrbind.trellis), 241
cc135(datasets), 60
cC176(datasets), 60
cement(datasets), 60
census4(datasets), 60
cereals(datasets), 60
chimp(datasets), 60
chisq.curve(F.curve), 72
chisq.observed(F.curve), 72
chisq.setup(F.curve), 72
ci.plot, 12, 49, 205
CIplot, 51
circuit(datasets), 60
class, 190
cO2(datasets), 60
cO2Covas(aoc), 25
cO2ArimaHH(extra), 62
cO2ArimaHHHH(extra), 71
col.hh, 54
col3x2, 55
colorRampPalette, 121
ColorSet(likertColor), 120
colPcts(rowPcts), 258
combineLimits, 56, 57
combineLimits.trellisvector, 56
Commander, 43
concord(datasets), 60
confint.glht, 45, 151
confint.interval.matrix(CIplot), 51
confint.interval.data(CIplot), 51
contrMat, 147
covariance(aoc), 25
cp.calc, 57
cplx, 60
crash(datasets), 60
crime(datasets), 60
darwin(datasets), 60
data, 43
datasets, 60
dchisq.intermediate, 61
ddiscunif (pdiscunif), 216
defunct, 62
deparse, 45
Design_2.8.2 (datasets), 60
Design_2.8.2_full (datasets), 60
dev2, 71
df.intermediate (dchisq.intermediate), 61
diag, 63
diag.arma.loop (arma.loop), 37
diag.maybe.null, 63
diagQQ, 65, 143
diamond (datasets), 60
differential (likert), 104
Discrete4, 66
display (datasets), 60
distress (datasets), 60
diverge_hcl, 110, 112, 121, 122, 125
do.formula.trellis.xysplom, 67
draft (datasets), 60
draft70mn (datasets), 60
drunk (datasets), 60
eggs (datasets), 60
elnino (datasets), 60
EmphasizeVerticalPanels, 68
employM16 (datasets), 60
emptyMainLeftAxisLeftStripBottomLegend, 69
emptyMainLeftAxisLeftStripBottomLegend, 69
emptyRightAxis
   (print.TwoTrellisColumns), 234
ergy (datasets), 60
esr (datasets), 60
export.eps, 70
extra, 71
Extract, 227, 265
F.curve, 72
F.observed (F.curve), 72
F.setup (F.curve), 72
fabricwear (datasets), 60
factor, 87, 227
fat (datasets), 60
feed (datasets), 60
filmcoat (datasets), 60
filter (datasets), 60
floating (likert), 104
format, 177, 183
formula, 67
fruitflies (datasets), 60
furnace (datasets), 60
girlht (datasets), 60
glasses (datasets), 60
glh.t, 46, 75, 142, 147, 149–151, 157, 221, 225
glh.mmc (defunct), 62
glhWithMCP, 993, 75
gof.calculation, 76
gofplot (tsdiagplot), 270
golf (datasets), 60
grid.text, 72
grid.xaxis.hh (grid.yaxis.hh), 77
grid.yaxis.hh, 77
GSremove, 78
gum (datasets), 60
gunload (datasets), 60
har1 (datasets), 60
har2 (datasets), 60
har3 (datasets), 60
hardness (datasets), 60
heartvalve (datasets), 60
HH (HH-package), 5
hh (defunct), 62
HH-defunct (defunct), 62
HH-package, 5
hh.file (defunct), 62
HH.regsubsets, 78
hhcapture (hhpdf), 80
hhcode (hhpdf), 80
hhdev.off (hhpdf), 80
hhlatex (hhpdf), 80
hhpdf, 80
hhpng (hhpdf), 80
HHscriptnames, 80, 81
hooppine (datasets), 60
hospital (datasets), 60
hotdog (datasets), 60
houseprice (datasets), 60
hov, 12, 82, 219
hovBF, 84
hovPlot, 83
hovPlot (plot.hov), 218
hovplotBF (hovBF), 84
hpErie (datasets), 60
INDEX

htwt (datasets), 60
iceskate (datasets), 60
icu (datasets), 60
if.R, 85
income (datasets), 60
inconsistent (datasets), 60
InsertVerticalPanels, 86
interaction.positioned, 87, 90, 209
interaction2wt, 12, 48, 88, 104, 198, 210
interval, 92
intubate (datasets), 60
intxpplot, 93, 264
ironpot (datasets), 60
is.likert (as.likert), 39
is.likertCapable (as.likert), 39
is.listOfNamedMatrices (as.matrix.listOfNamedMatrices), 41
is.na.positioned (position), 226
is.numeric.positioned (position), 226
is.positioned (position), 226
is.R, 85
jury (datasets), 60
kangaroo (datasets), 60
kidney (datasets), 60
kyphosis (datasets), 60

ladder, 12, 96, 203
ladder3 (ladder), 96
lake (datasets), 60
latex, 100, 101, 215, 216
latex.array, 99
latex.matrix (latex.array), 99
latex.table (latex.array), 99
latticeresids, 103
leftLabels.trellis
 (print.TwoTrellisColumns), 234
legendGrob2wt, 104
leukemia (datasets), 60
lft.asat (datasets), 60
lifeins (datasets), 60
likertColor, 110, 120, 125
likertColorBrewer, 112
likertColorBrewer (likertColor), 120
likertMosaic, 123
LikertPercentCountColumns, 128
likertplot (likert), 104
likertWeighted, 68, 130, 268
likertWeighted47, 133
lm, 35, 51, 139, 143, 205, 280, 282
lm.case, 136
lm.case (defunct), 62
lm.influence, 138
lm.regsubsets, 139
lm.matContrast (lm.matRows), 141
lm.matPairwise, 140
lm.matRows, 141
lm.plot, 142
logit, 143
logrelrisk (ae.dotplot), 12
longley (datasets), 60
ls, 190
ls.str, 190
lymph (datasets), 60

mainSubLegend.trellis
 (print.TwoTrellisColumns), 234
maiz (datasets), 60
make.xaxis.hh.labels (grid.yaxis.hh), 77
make.yaxis.hh.labels (grid.yaxis.hh), 77
manhours (datasets), 60
market (datasets), 60
matrix, 145
matrix (as.matrix.listOfNamedMatrices), 41
matrix.trellis, 47, 56, 144
mcalinfct, 146
mcp, 140
mcp2matrix.993 (glhtWithMCP.993), 75
median, 84
mice (datasets), 60
mileage (datasets), 60
MMC, 12, 35, 147, 166, 168, 195
MMC (mmc), 147
mmc, 45, 46, 75, 140–142, 147, 156, 157, 161, 163, 164, 206, 211, 222, 223, 225
mmc.mean, 155
mmcAspect, 159
mmcboth, 164
mmcboth (mmcisomeans), 159
mmcisomeans, 159, 164
mmcmatch, 164
mmcmatch (mmcisomeans), 159
<table>
<thead>
<tr>
<th>Function</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mmcplot</td>
<td>147, 151, 159, 161, 162, 220, 222, 223</td>
</tr>
<tr>
<td>mmcPruneIsomeans</td>
<td>164</td>
</tr>
<tr>
<td>model.frame.ancova</td>
<td>25</td>
</tr>
<tr>
<td>model.tables.ancovaplot</td>
<td>32</td>
</tr>
<tr>
<td>modelparm</td>
<td>45</td>
</tr>
<tr>
<td>mortality (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>mosaic</td>
<td>127</td>
</tr>
<tr>
<td>mpg (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>multicomponent (mmc)</td>
<td>147</td>
</tr>
<tr>
<td>multicomponent.label.change</td>
<td>(multicomponent.order), 165</td>
</tr>
<tr>
<td>multicomponent.mean (mmc.mean)</td>
<td>155</td>
</tr>
<tr>
<td>multicomponent.mmc.mean (mmc.mean)</td>
<td>155</td>
</tr>
<tr>
<td>multicomponent.order (165, 168)</td>
<td></td>
</tr>
<tr>
<td>multicomponent.reverse (166, 167)</td>
<td></td>
</tr>
<tr>
<td>muscle (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>na.action (67, 252, 280, 281, 283)</td>
<td></td>
</tr>
<tr>
<td>njgolf (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>norm.curve</td>
<td>169</td>
</tr>
<tr>
<td>norm.observed (norm.curve)</td>
<td>169</td>
</tr>
<tr>
<td>norm.outline (norm.curve)</td>
<td>169</td>
</tr>
<tr>
<td>norm.setup (norm.curve)</td>
<td>169</td>
</tr>
<tr>
<td>normal_and_t.dist (norm.curve)</td>
<td>169</td>
</tr>
<tr>
<td>NormalAndTPlot (175, 177, 187, 188, 232)</td>
<td></td>
</tr>
<tr>
<td>NormalAndTPower (182)</td>
<td></td>
</tr>
<tr>
<td>normalApproxBinomial (184)</td>
<td></td>
</tr>
<tr>
<td>normtemp (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>notch (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>npar.arma</td>
<td>185</td>
</tr>
<tr>
<td>npararma (npar.arma)</td>
<td>185</td>
</tr>
<tr>
<td>npar.sarma (npar.arma)</td>
<td>185</td>
</tr>
<tr>
<td>NTPlot (12, 176, 179, 183, 184, 186, 232)</td>
<td></td>
</tr>
<tr>
<td>NZScienceTeaching (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>oats (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>objip</td>
<td>189</td>
</tr>
<tr>
<td>odds (logit)</td>
<td>143</td>
</tr>
<tr>
<td>odds.ratio (defunct)</td>
<td>62</td>
</tr>
<tr>
<td>OddsRatio</td>
<td>190</td>
</tr>
<tr>
<td>odoffna (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>OneWayVarPlot</td>
<td>192</td>
</tr>
<tr>
<td>operating characteristic curve</td>
<td>(NormalAndTPower), 182</td>
</tr>
<tr>
<td>operator (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>oral (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>orthog.complete</td>
<td>193</td>
</tr>
<tr>
<td>orthog.construct (orthog.complete)</td>
<td>193</td>
</tr>
<tr>
<td>ozone (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>panel.acf</td>
<td>196</td>
</tr>
<tr>
<td>panel.ae.dotplot (ae.dotplot)</td>
<td>12</td>
</tr>
<tr>
<td>panel.ae.leftplot (ae.dotplot)</td>
<td>12</td>
</tr>
<tr>
<td>panel.ae.rightplot (ae.dotplot)</td>
<td>12</td>
</tr>
<tr>
<td>panel.ancova (ancova)</td>
<td>25</td>
</tr>
<tr>
<td>panel.ancova.superpose (ancovaplot)</td>
<td>29</td>
</tr>
<tr>
<td>panel.axis (177, 196, 197, 274)</td>
<td></td>
</tr>
<tr>
<td>panel.axis.right</td>
<td>196</td>
</tr>
<tr>
<td>panel.barchart</td>
<td>212</td>
</tr>
<tr>
<td>panel.barchart2 (panelLikert)</td>
<td>211</td>
</tr>
<tr>
<td>panel.bwplot (131, 198, 201, 266)</td>
<td></td>
</tr>
<tr>
<td>panel.bwplot.groups</td>
<td>(panel.bwplot.superpose), 198</td>
</tr>
<tr>
<td>panel.bwplot.intermediate.hh</td>
<td>197, 199, 210</td>
</tr>
<tr>
<td>panel.bwplot.superpose</td>
<td>198</td>
</tr>
<tr>
<td>panel.bwplott (201)</td>
<td></td>
</tr>
<tr>
<td>panel.cartesian (99, 202, 238)</td>
<td></td>
</tr>
<tr>
<td>panel.case (lm.case)</td>
<td>136</td>
</tr>
<tr>
<td>panel.ci.plot</td>
<td>204</td>
</tr>
<tr>
<td>panel.confintMMC</td>
<td>205</td>
</tr>
<tr>
<td>panel.dotplot (tb), 206</td>
<td></td>
</tr>
<tr>
<td>panel.gof (panel.acf)</td>
<td>196</td>
</tr>
<tr>
<td>panel.hov (plot.hov), 218</td>
<td></td>
</tr>
<tr>
<td>panel.interaction2wt (90, 91, 208, 227)</td>
<td></td>
</tr>
<tr>
<td>panel.intxplot (intxplot)</td>
<td>93</td>
</tr>
<tr>
<td>panel.isomeans (210)</td>
<td></td>
</tr>
<tr>
<td>panel.Likert</td>
<td>211</td>
</tr>
<tr>
<td>panel.pairs.hh (213)</td>
<td></td>
</tr>
<tr>
<td>panel.qqmathline (65)</td>
<td></td>
</tr>
<tr>
<td>panel.residSquare (regrresidplot)</td>
<td>248</td>
</tr>
<tr>
<td>panel.std.resid (panel.acf)</td>
<td>196</td>
</tr>
<tr>
<td>panel.superpose (30, 199)</td>
<td></td>
</tr>
<tr>
<td>panel.wireframe (49)</td>
<td></td>
</tr>
<tr>
<td>panel.xyplot (27, 31, 198, 209)</td>
<td></td>
</tr>
<tr>
<td>panel.xysplom (214)</td>
<td></td>
</tr>
<tr>
<td>panelOnly.trellis (print.TwoTrellisColumns)</td>
<td>234</td>
</tr>
<tr>
<td>paper (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>partial.corr (214)</td>
<td></td>
</tr>
<tr>
<td>patient (datasets)</td>
<td>60</td>
</tr>
<tr>
<td>pchisq (62, 74)</td>
<td></td>
</tr>
<tr>
<td>pchisq.intermediate</td>
<td>(dchisq.intermediate), 61</td>
</tr>
<tr>
<td>pdf.latex (215)</td>
<td></td>
</tr>
</tbody>
</table>
INDEX 293

pdiscunif, 216
persp, 246, 247
persp.back.wall.x (defunct), 62
persp.back.wall.y (defunct), 62
persp.floor (defunct), 62
persp.plane (defunct), 62
persp.setup (defunct), 62
perspBack.wall.x (perspPlane), 217
perspBack.wall.y (perspPlane), 217
perspFloor (perspPlane), 217
perspPlane, 217
pf, 62, 74
pf.intermediate (dchisq.intermediate), 61
plasma (datasets), 60
plot.ancova (ancova), 25
plot.case (lm.case), 136
plot.hov (defunct), 62
plot.hov (likert), 104
plot.likert (likert), 104
plot.likert.list, 257
plot.matchMMC (defunct), 62
plot.mmc.multicomp, 34, 151, 156, 220
plot.multicomp, 223
plot.odds.ratio (defunct), 62
plot.summaryHH.regsubsets (HH.regsubsets), 78
plotMatchMMC, 222
plotMatchMMC (plot.multicomp), 223
plotOddsRatio (OddsRatio), 190
political (datasets), 60
PoorChildren (datasets), 60
pop.vp.hh (push.vp.hh), 238
position, 198, 199, 226, 230
position< (position), 226
positioned, 88
positioned (position), 226
positioned-class, 230
potency (datasets), 60
power curve (NormalAndTPower), 182
power.t.test, 187
powerplot, 178
powerplot (NormalAndTPower), 182
pox (datasets), 60
predict.ancova (ancova), 25
predict.glm, 92
predict.lm, 51
print.AEdotplot (AEdotplot.data.frame), 20
print.ancova (ancova), 25
print.arma.loop (summary.arma.loop), 265
print.cp.object (cp.calc), 57
print.glht.mmc.multicomp (defunct), 62
print.latticeresids, 231, 253
print.listOfNamedMatrices
(as.matrix.listOfNamedMatrices), 41
print.MatrixList
(as.matrix.listOfNamedMatrices), 41
print.mmc.multicomp (as.multicomp), 44
print.multicomp (as.multicomp), 44
print.NormalAndTplot, 188, 232
print.positioned (position), 226
print.pyramidLikert (pyramidLikert), 239
print.summaryHH.regsubsets (HH.regsubsets), 78
print.trellis, 14, 231, 232, 235, 239, 265
print.tsciagplot, 233
print.TwoTrellisColumns, 234
print.TwoTrellisColumns5 (print.TwoTrellisColumns), 234
print1.tsciagplot (print.tsciagplot), 233
print2.tsciagplot (print.tsciagplot), 233
product (datasets), 60
ProfChal (datasets), 60
ProfDiv (datasets), 60
psycho (datasets), 60
pulmonary (datasets), 60
pulse (datasets), 60
push.vp.hh, 238
pyramid (likert), 104
pyramidLikert, 113, 239
qchisq.intermediate
(dchisq.intermediate), 61
qdiscunif (pdiscunif), 216
qf.intermediate (dchisq.intermediate), 61
qqmath, 65
R282 (datasets), 60
INDEX

295

tires (datasets), 60
title.grob (extra), 71

xyplot, 14, 18, 22, 23, 27, 52, 56, 64, 90, 103, 131, 134, 160, 163, 177, 178, 184, 188, 198, 202, 205, 206, 251–253, 269, 273, 284

title.trellis (extra), 71
toCQxR, 131, 267
tongue (datasets), 60
transpose (rbind.trellis), 241
trellis.device, 54
trellis.par.get, 54, 134, 273
treecfplots, 12, 260, 266, 268, 272
tsdiagplot, 12, 37, 38, 196, 233, 266, 270
tser.mystery.X (datasets), 60
tser.mystery.Y (datasets), 60
tser.mystery.Z (datasets), 60
tsq (datasets), 60
turkey (datasets), 60
tv (datasets), 60

unique, 227
unique.positioned (position), 226
unit, 238
unpositioned (position), 226
update.AEdotplot
(AEdotplot.data.frame), 20
usair (datasets), 60
uscrime (datasets), 60
useOuterScales, 262, 263, 273
useOuterStrips, 273, 274
useOuterStripsT2L1, 278

vcov.sufficient (defunct), 62
vcovSufficient (aovSufficient), 34
viewport, 238
vif, 279, 282
vocab (datasets), 60
vulcan (datasets), 60

washday (datasets), 60
water (datasets), 60
weightloss (datasets), 60
weld (datasets), 60
wheat (datasets), 60
WindowsPath (HHscriptnames), 81
wireframe, 49
wool (datasets), 60
workstation (datasets), 60

X.residuals, 281
xscale.components.top.HH (likert), 104