Package 'zenplots'

August 17, 2026

Title Zigzag Expanded Navigation Plots

Version 1.0.7

Date 2025-09-08

Maintainer Wayne Oldford <rwoldford@uwaterloo.ca>

Description Graphical tools for visualizing high-dimensional data along a path of alternating one- and two-dimensional plots. Includes optional interactive graphics via 'loon' (which uses 'tcltk' from base R). Support is provided for constructing graph structures and, when available, plotting them with Bioconductor packages (e.g., 'graph', 'Rgraphviz'); these are optional and examples/vignettes are skipped if they are not installed. For algorithms and further details, see <doi:10.18637/jss.v095.i04>.

License GPL-2 | GPL-3

URL https://great-northern-diver.github.io/zenplots/, https://github.com/great-northern-diver/zenplots/

BugReports https://github.com/great-northern-diver/zenplots/issues

Depends R (>= 3.4.0)

Imports graphics, grid, MASS, methods, PairViz, stats

Suggests ADGofTest, BiocManager, copula, crop, ggplot2, graph, gridExtra, knitr, lattice, loon, Matrix, mvtnorm, pcaPP, qqtest, qrmdata, qrmtools, Rgraphviz, rmarkdown, rugarch, scagnostics, testthat, zoo

VignetteBuilder knitr, rmarkdown

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

RoxygenNote 7.3.3

Author Marius Hofert [aut], Wayne Oldford [aut, cre]

Date/Publication 2025-09-09 07:50:03 UTC

Contents

adjust_bb . . . 4
arrow_1d_graphics . . . 4
arrow_1d_grid . . . 5
arrow_1d_loon . . . 7
arrow_2d_graphics . . . 8
arrow_2d_grid . . . 9
arrow_2d_loon . . . 10
as_numeric . . . 11
axes_2d_graphics . . . 12
axes_2d_grid . . . 13
axes_2d_loon . . . 14
boxplot_1d_graphics . . . 16
boxplot_1d_grid . . . 17
boxplot_1d_loon . . . 18
burst . . . 20
burst_aux . . . 21
check_zargs . . . 22
connect_pairs . . . 23
convert_occupancy . . . 24
density_1d_graphics . . . 25
density_1d_grid . . . 26
density_1d_loon . . . 27
density_2d_graphics . . . 28
density_2d_grid . . . 30
density_2d_loon . . . 31
de_elect . . . 33
extract_1d . . . 35
extract_2d . . . 37
extract_pairs . . . 39
get_layout . . . 40
get_path . . . 41
get_zigzag_turns . . . 42
graph_pairs . . . 42
groupData . . . 43
group_2d_graphics . . . 44
group_2d_grid . . . 46
group_2d_loon . . . 47
happiness . . . 48
hist_1d_graphics . . . 49
hist_1d_grid . . . 50
hist_1d_loon . . . 52
indexData . . . 53
is.standard . . . 54
jitter_1d_graphics . . . 55
jitter_1d_grid . . . 56
jitter_1d_loon . . . 57

label_1d_graphics . . . 58
label_1d_grid . . . 59
label_1d_loon . . . 60
label_2d_graphics . . . 62
label_2d_grid . . . 63
label_2d_loon . . . 64
layout_1d_graphics . . . 66
layout_1d_grid . . . 66
layout_1d_loon . . . 67
layout_2d_graphics . . . 68
layout_2d_grid . . . 68
layout_2d_loon . . . 69
lines_1d_graphics . . . 70
lines_1d_grid . . . 71
lines_1d_loon . . . 72
l_ispace_config . . . 73
move . . . 74
n2dcols_aux . . . 75
na_omit_loon . . . 75
next_move_tidy . . . 76
num_cols . . . 77
olive . . . 78
plot_exists . . . 78
plot_indices . . . 79
plot_region . . . 80
points_1d_graphics . . . 81
points_1d_grid . . . 82
points_1d_loon . . . 83
points_2d_graphics . . . 84
points_2d_grid . . . 85
points_2d_loon . . . 86
qq_2d_graphics . . . 88
qq_2d_grid . . . 89
rect_1d_graphics . . . 90
rect_1d_grid . . . 91
rect_1d_loon . . . 93
rect_2d_graphics . . . 94
rect_2d_grid . . . 95
rect_2d_loon . . . 96
rug_1d_graphics . . . 98
rug_1d_grid . . . 99
rug_1d_loon . . . 100
turn_checker . . . 101
unfold . . . 102
vport . . . 104
wine . . . 105
zenarrow . . . 106
zenpath . . . 107

zenplot 109

Index **122**

`adjust_bb` *Auxiliary function for adjusting a bounding box*

Description

Auxiliary function for adjusting a bounding box

Usage

```
adjust_bb(lastturn, coordslastBB, w, h)
```

Arguments

`lastturn`	last turn
`coordslastBB`	coordinates of the last bounding box
`w`	width
`h`	height

Value

Coordinates of the adjusted bounding box

Author(s)

Wayne Oldford

`arrow_1d_graphics` *Arrow plot in 1d using R's base graphics*

Description

Arrow plot in 1d using R's base graphics

Usage

```
arrow_1d_graphics(
  zargs,
  loc = c(0.5, 0.5),
  angle = 60,
  length = 0.6,
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
angle	angle in [0, 180]
length	length of the arrow in [0,1] from tip to base
add	logical indicating whether this plot should be added to the last one
plot...	additional arguments passed to plot_region()
...	additional arguments passed to segments()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: boxplot_1d_graphics(), density_1d_graphics(), hist_1d_graphics(), jitter_1d_graphics(), label_1d_graphics(), lines_1d_graphics(), points_1d_graphics(), rect_1d_graphics(), rug_1d_graphics()

Other default 1d plot functions: arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

arrow_1d_grid *Arrow plot in 1d using the grid package*

Description

Arrow plot in 1d using the grid package

Usage

```
arrow_1d_grid(
  zargs,
  loc = c(0.5, 0.5),
  angle = 60,
  length = 0.6,
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
angle	angle from the shaft to the edge of the arrow head
length	length of the arrow in [0,1] from tip to base
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: boxplot_1d_grid(), density_1d_grid(), hist_1d_grid(), jitter_1d_grid(), label_1d_grid(), lines_1d_grid(), points_1d_grid(), rect_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

`arrow_1d_loon` *Arrow plot in 1d using the interactive loon package*

Description

Arrow plot in 1d using the interactive loon package

Usage

```
arrow_1d_loon(
  zargs,
  loc = c(0.5, 0.5),
  length = 0.6,
  angle = NULL,
  linkingGroup = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  ...
)
```

Arguments

`zargs`	The argument list as passed from `zenplot()`
`loc`	The (x,y) location of the center of the arrow
`length`	The length of the arrow
`angle`	The angle from the shaft to the edge of the arrow head
`linkingGroup`	A string specifying the initial group of plots to be linked to this plot
`showLabels`	Logical determining whether axis labels are displayed
`showScales`	Logical determining whether scales are displayed
`showGuides`	Logical determining whether the background guidelines are displayed
`baseplot`	If non-null the base plot on which the plot should be layered
`parent`	The tk parent for this loon plot widget
`...`	Additional parameters passed to loon::l_layer_line(...)

Value

A loon loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: boxplot_1d_loon(), density_1d_loon(), hist_1d_loon(), jitter_1d_loon(), label_1d_loon(), lines_1d_loon(), points_1d_loon(), rect_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

arrow_2d_graphics *Arrow plot in 2d using R's base graphics*

Description

Arrow plot in 2d using R's base graphics

Usage

```
arrow_2d_graphics(
  zargs,
  loc = c(0.5, 0.5),
  angle = 60,
  length = 0.2,
  add = FALSE,
  group... = NULL,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location (in (0,1)^2) of the center of the arrow
angle	angle from the shaft to the edge of the arrow head
length	length of the arrow in [0,1] from tip to base
add	logical indicating whether this plot should be added to the last one
group...	list of arguments passed to group_2d_graphics (or NULL)
plot...	additional arguments passed to plot_region()
...	additional arguments passed to points()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using R's base graphics: axes_2d_graphics(), density_2d_graphics(), group_2d_graphics(), label_2d_graphics(), points_2d_graphics(), qq_2d_graphics(), rect_2d_graphics()

Other default 2d plot functions: arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

arrow_2d_grid *Arrow plot in 2d using the grid package*

Description

Arrow plot in 2d using the grid package

Usage

```
arrow_2d_grid(
  zargs,
  loc = c(0.5, 0.5),
  angle = 60,
  length = 0.2,
  group... = list(cex = 0.66),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location of the center of the arrow
angle	angle from the shaft to the edge of the arrow head
length	length of the arrow in [0,1] from tip to base
group...	list of arguments passed to group_2d_grid (or NULL)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the grid package: axes_2d_grid(), density_2d_grid(), group_2d_grid(), label_2d_grid(), points_2d_grid(), qq_2d_grid(), rect_2d_grid()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

arrow_2d_loon *Arrow plot in 2d using the interactive loon package*

Description

Arrow plot in 2d using the interactive loon package

Usage

```
arrow_2d_loon(
  zargs,
  loc = rep(0.5, 2),
  length = 0.2,
  angle = 30,
  linkingGroup = NULL,
  color = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  group... = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
loc	The (x,y) location of the center of the arrow
length	The length of the arrow
angle	The angle from the shaft to the edge of the arrow head
linkingGroup	The initial linking group
color	The color
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
group...	A list of arguments passed to group_2d_loon (or NULL)
...	Additional parameters passed to loon::l_layer_line()

Value

the plot (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the interactive loon package: axes_2d_loon(), density_2d_loon(), group_2d_loon(), label_2d_loon(), points_2d_loon(), rect_2d_loon()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

as_numeric *A list of columns*

Description

A list of columns

Usage

```
as_numeric(x)
```

Arguments

x	A list of columns

Value

A list where each column is converted to data (range() works, can be plotted, etc.)

Note

See plot.default -> xy.coords()

Author(s)

Marius Hofert

---

axes_2d_graphics *Axes arrows in 2d using R's base graphics*

Description

Axes arrows in 2d using R's base graphics

Usage

```
axes_2d_graphics(
  zargs,
  length = 0.1,
  eps = 0.04,
  code = 2,
  xpd = NA,
  add = FALSE,
  group... = NULL,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
length	length of the arrow head
eps	distance by which the axes are moved away from the plot region
code	integer code determining the kind of arrows to be drawn; see ?arrows
xpd	logical or NA, determining the region with respect to which clipping takes place; see ?par
add	logical indicating whether this plot should be added to the last one

group...	list of arguments passed to group_2d_graphics (or NULL)
plot...	additional arguments passed to plot_region()
...	additional arguments passed to points()

Value

invisible()

Note

Inspired by https://stat.ethz.ch/pipermail/r-help/2004-October/059525.html

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using R's base graphics: arrow_2d_graphics(), density_2d_graphics(), group_2d_graphics(), label_2d_graphics(), points_2d_graphics(), qq_2d_graphics(), rect_2d_graphics()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

axes_2d_grid *Axes arrow using the grid package*

Description

Axes arrow using the grid package

Usage

```
axes_2d_grid(
  zargs,
  angle = 30,
  length = unit(0.05, "npc"),
  type = "open",
  eps = 0.02,
  group... = list(cex = 0.66),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
angle	angle of the arrow head (see ?arrow)
length	length of the arrow in [0,1] from tip to base
type	type of the arrow head (see ?arrow)
eps	distance by which the axes are moved away from the plot region
group...	list of arguments passed to group_2d_grid (or NULL)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Note

Inspired by https://stat.ethz.ch/pipermail/r-help/2004-October/059525.html

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the grid package: arrow_2d_grid(), density_2d_grid(), group_2d_grid(), label_2d_grid(), points_2d_grid(), qq_2d_grid(), rect_2d_grid()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

axes_2d_loon *Axes arrows in 2d using the interactive loon package*

Description

Axes arrows in 2d using the interactive loon package

Usage

```
axes_2d_loon(
  zargs,
  angle = 30,
  length = 0.05,
  eps = 0.02,
  linkingGroup = NULL,
  color = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  group... = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
angle	The angle of the arrow head
length	The length of the arrow head
eps	The distance by which the axes are moved away from the plot region
linkingGroup	The initial linking group
color	Colour used fill if ccol is NULL, a grey palette is used otherwise.
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
group...	A list of arguments passed to group_2d_loon (or NULL)
...	Additional arguments passed to loon::l_plot()

Value

the loon plot

Note

Inspired by https://stat.ethz.ch/pipermail/r-help/2004-October/059525.html

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the interactive loon package: arrow_2d_loon(), density_2d_loon(), group_2d_loon(), label_2d_loon(), points_2d_loon(), rect_2d_loon()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

boxplot_1d_graphics *Box plot in 1d using R's base graphics*

Description

Box plot in 1d using R's base graphics

Usage

```
boxplot_1d_graphics(
  zargs,
  cex = 0.4,
  range = NULL,
  axes = FALSE,
  add = FALSE,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
cex	The character expansion factor
range	A numerical value which determines how far the plot whiskers extend. If NULL, the whiskers (range) grows with sample size.
axes	A logicial indicating whether axes should be drawn
add	A logical indicating whether this plot should be added to the last one
...	Additional arguments passed to boxplot()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: arrow_1d_graphics(), density_1d_graphics(), hist_1d_graphics(), jitter_1d_graphics(), label_1d_graphics(), lines_1d_graphics(), points_1d_graphics(), rect_1d_graphics(), rug_1d_graphics()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

boxplot_1d_grid *Boxplot in 1d using the grid package*

Description

Boxplot in 1d using the grid package

Usage

```
boxplot_1d_grid(
  zargs,
  pch = 21,
  size = 0.02,
  col = NULL,
  lwd = 2,
  bpwidth = 0.5,
  range = NULL,
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
pch	plot symbol
size	size of the plot symbol
col	color
lwd	graphical parameter line width for whiskers and median
bpwidth	width of boxplot on scale of default.units
range	numerical value used to determine how far the plot whiskers extend. If NULL, the whiskers (range) grows with sample size.
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

gTree grob containing the boxplot components as grobs

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), density_1d_grid(), hist_1d_grid(), jitter_1d_grid(), label_1d_grid(), lines_1d_grid(), points_1d_grid(), rect_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

boxplot_1d_loon *Boxplot in 1d using the interactive loon package*

Description

Boxplot in 1d using the interactive loon package

Usage

```
boxplot_1d_loon(
  zargs,
  color = NULL,
  linecolor = NULL,
  lwd = 2,
  range = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  linkingGroup = NULL,
  baseplot = NULL,
  parent,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
color	colour for boxplot
linecolor	Colour used for the lines to draw the boxplot
lwd	The parameter line width for whiskers and median and box boundaries
range	numerical value used to determine how far the plot whiskers extend. If NULL, the whiskers (range) grows with sample size.
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
linkingGroup	A string specifying the initial group of plots to be linked to this plot
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
...	Additional parameters passed to gpar()

Value

A loon loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), density_1d_loon(), hist_1d_loon(), jitter_1d_loon(), label_1d_loon(), lines_1d_loon(), points_1d_loon(), rect_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

`burst` *Splitting an Input Object into a List of Columns*

Description

Splits a (numeric/logical/character) vector, matrix, data.frame or a list of such into a list of columns, with corresponding group and variable information as well as labels. This is an auxiliary function for checking and converting the data argument of zenplot().

Usage

```
burst(x, labs = list())
```

Arguments

`x`	A `numeric vector`, `matrix`, `data.frame` or, for `burst()`, a `list` of such.
`labs`	Either `NULL` (in which case neither group nor variable labels are used or computed) or a list with components `group` - the group label basename or labels for the groups (or `NULL` for no group labels) `var` - the variable label basename or labels for the variables (or `NULL` for no variable labels) `sep` - the string used as the separator between group and variable labels `group2d` - a `logical` indicating whether labels of `group_2d_*()` plots are affected by `group = NULL` (or printed anyway) If any of these components is not given, it is set to the defaults as described in `zenplot()`. Note that if at least one (group or variable) label is given in `x`, then those (original) labels will be used. If labs = NULL, neither group nor variable labels are used.

Value

A `list` with components

`xcols` - a list containing the column vectors of `x`

`groups` - the group number for each column of `x`

`vars` - the variable number (within each group) for each column of `x`

`glabs` - the group label for each column of `x`

`labs` - the group and variable labels for each column of `x`

Note

Performance critical

Author(s)

Marius Hofert

See Also

Other tools for constructing your own plot1d and plot2d functions: burst_aux(), check_zargs(), extract_1d(), extract_2d(), plot_indices()

Examples

```
## Unnamed list of (some named, some unnamed) valid components
A <- matrix(1:12, ncol = 3)
x <- list(A, 1:4, as.data.frame(A))

burst(x, labs = list(group = "G", var = "V", sep = ", "))
burst(x) # the same defaults as above
burst(x, labs = list(sep = " ")) # only changing the separator
## Note: - No group labels are given in 'x' and thus they are constructed
##         in the above call
##       - The variable names are only constructed if not given

burst(x, labs = list(group = ""))
burst(x, labs = list(group = NULL, group2d = TRUE)) # no group labels
##  Note: There's no effect of 'group2d = TRUE' visible here as
##        'x' doesn't contain group labels

burst(x, labs = list(group = NULL)) # no group labels unless groups change
burst(x, labs = list(var = NULL)) # no variable labels
burst(x, labs = list(group = NULL, var = NULL)) # neither one
burst(x, labs = NULL) # similarly, without any labels at all

##  Named list
x <- list(mat = A, vec = 1:4, df = as.data.frame(A))
burst(x)
##  Note: - The given group labels are used
##        - The variable names are only constructed if not given

burst(x, labs = list(group = NULL, group2d = TRUE)) # no group labels
burst(x, labs = list(group = NULL)) # no group labels unless groups change
##  Note: Now the effect of 'group2d' is visible.

##  Partially named list
x <- list(mat = A, vec = 1:4, as.data.frame(A))
burst(x)
burst(x, labs = list(group = NULL, group2d = TRUE)) # no group labels
burst(x, labs = list(group = NULL)) # no group labels unless groups change
burst(x, labs = list(var = NULL)) # no variable labels
burst(x, labs = list(group = NULL, var = NULL)) # only group labels and only if groups change
burst(x, labs = NULL) # neither group nor variable labels
```

---

burst_aux *Auxiliary function for burst()*

Description

Auxiliary function for burst()

Usage

```
burst_aux(x, labs = "V")
```

Arguments

x	A vector, matrix or data.frame (or a (pure) list, but that we don't use here)
labs	The variable labels: - if NULL, no labels are used - if of length 1, use this label and append 1:ncol(x) but only if x doesn't have any column names (otherwise use the latter) - if of length ncol(x), use that but only if x doesn't have any column names (otherwise use the latter)

Value

'x' as a list of named columns

Note

- Performance critical (no checks here) - Data frames always have default names. They are possibly ugly but we have to use them here as we cannot determine whether they were assigned automatically or on purpose.

Author(s)

Marius Hofert

See Also

Other tools for constructing your own plot1d and plot2d functions: `burst()`, `check_zargs()`, `extract_1d()`, `extract_2d()`, `plot_indices()`

---

check_zargs *Checking whether certain arguments appear in zargs*

Description

Checking whether certain arguments appear in zargs

Usage

```
check_zargs(zargs, ...)
```

Arguments

zargs	The argument list as passed from zenplot()
...	The arguments to be checked for presence in zargs

Value

A logical indicating whether some arguments are missing in zargs

Author(s)

Marius Hofert

See Also

Other tools for constructing your own plot1d and plot2d functions: burst(), burst_aux(), extract_1d(), extract_2d(), plot_indices()

---

connect_pairs *Connecting Possibly Overlapping Pairs Into a List of Paths*

Description

Pairs, given as rows of a matrix, data.frame, or list, are processed to return a list of paths, each identifying the connected pairs in the rows of x.

Usage

```
connect_pairs(x, duplicate.rm = FALSE)
```

Arguments

x	two-column matrix, data.frame, or a list containing vectors of length two representing the pairs to be connected.
duplicate.rm	logical indicating whether equal pairs (up to permutation) are to be omitted.

Details

This function is self-contained and has no external package dependencies.

Value

A list each of whose elements give a path of connected pairs. Each list element is a vector of length at least 2 (longer vectors > 2 in length identify the pairs connected in a path).

Author(s)

Marius Hofert and Wayne Oldford

See Also

zenplot() which provides the zenplot.

Other tools related to constructing zenpaths: extract_pairs(), graph_pairs(), groupData(), indexData(), zenpath()

Examples

```
(pairs <- matrix(c(1,2,2,3,3,5,5,7,8,9), ncol = 2, byrow = TRUE))
connect_pairs(pairs)
nVars <- 5
pairs <- expand.grid(1:nVars, 1:nVars)
(pairs <- pairs[pairs[,1] < pairs[,2],])
connect_pairs(pairs)
```

---

convert_occupancy *Converting an Occupancy Matrix*

Description

Convert an occupancy matrix to matrix with different symbols.

Usage

```
convert_occupancy(x, to = c("", "<", ">", "v", "^"))
```

Arguments

x	an occupancy matrix consisting of the character "" (unoccupied), "l" (left), "r" (right), "d" (down) or "u" (up) as returned by zenplot().
to	a vector of symbols to which "", "l", "r", "d" and "u" should be mapped.

Value

matrix as the occupancy matrix but with entries replaced by those in to.

Author(s)

Marius Hofert

See Also

Other zenplot technical tools: is.standard(), n2dcols_aux(), num_cols(), turn_checker()

Examples

```
## Generate some data
n <- 1000 # sample size
d <- 20 # dimension
set.seed(271) # set seed (for reproducibility)
x <- matrix(rnorm(n * d), ncol = d) # i.i.d. N(0,1) data

## Extract the occupancy matrix from a zenplot
res <- zenplot(x)
(occ <- res[["path"]][["occupancy"]])

## Convert the occupancy matrix
convert_occupancy(occ)
```

`density_1d_graphics` *Density plot in 1d using R's base graphics*

Description

Density plot in 1d using R's base graphics

Usage

```
density_1d_graphics(
  zargs,
  density... = NULL,
  offset = 0.08,
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

`zargs`	argument list as passed from `zenplot()`
`density...`	list of arguments for density()
`offset`	number in [0, 0.5] determining how far away the density stays from the plot margins (for creating space between the two)
`add`	logical indicating whether this plot should be added to the last one
`plot...`	additional arguments passed to plot_region()
`...`	additional arguments passed to polygon()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: arrow_1d_graphics(), boxplot_1d_graphics(), hist_1d_graphics(), jitter_1d_graphics(), label_1d_graphics(), lines_1d_graphics(), points_1d_graphics(), rect_1d_graphics(), rug_1d_graphics()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

density_1d_grid *Density plot in 1d using the grid package*

Description

Density plot in 1d using the grid package

Usage

```
density_1d_grid(zargs, density... = NULL, offset = 0.08, draw = FALSE, ...)
```

Arguments

zargs	argument list as passed from zenplot()
density...	list of arguments for density()
offset	numerical value in $[0, 0.5]$ used to offset the density within the height 1 box in which it appears
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), boxplot_1d_grid(), hist_1d_grid(), jitter_1d_grid(), label_1d_grid(), lines_1d_grid(), points_1d_grid(), rect_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

density_1d_loon *Density plot in 1d using the interactive loon package*

Description

Density plot in 1d using the interactive loon package

Usage

```
density_1d_loon(
  zargs,
  density.args = list(),
  method = c("single", "double"),
  lwd = NULL,
  linewidth = NULL,
  color = NULL,
  fill = NULL,
  linecolor = NULL,
  linkingGroup = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
density.args	A list of arguments for density()
method	A character specifying the type of density used
lwd	Line width used only when linewidth = NULL, value of 1 used otherwise.

linewidth	Line width of outline for density polygons (highest priority)
color	Colour used to fill the density when fill is NULL and to outline the density when linecolor is NULL, foreground colour used otherwise.
fill	Colour used to fill the density polygon
linecolor	Colour used for the outline of the density
linkingGroup	A string specifying the initial group of plots to be linked to this plot
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
...	Additional parameters passed to loon::l_layer()

Value

A loon loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), boxplot_1d_loon(), hist_1d_loon(), jitter_1d_loon(), label_1d_loon(), lines_1d_loon(), points_1d_loon(), rect_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

density_2d_graphics *Density plot in 2d using R's base graphics*

Description

Density plot in 2d using R's base graphics

Usage

```
density_2d_graphics(
  zargs,
  ngrids = 25,
  drawlabels = FALSE,
  axes = FALSE,
  box = FALSE,
  add = FALSE,
  group... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
ngrids	number of grid points in each dimension. Can be scalar or a length-2 integer vector.
drawlabels	logical indicating whether the contours should be labelled
axes	logicial indicating whether axes should be drawn
box	logical indicating whether a box should be drawn
add	logical indicating whether this plot should be added to the last one
group...	list of arguments passed to group_2d_graphics (or NULL)
...	additional arguments passed to points()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using R's base graphics: arrow_2d_graphics(), axes_2d_graphics(), group_2d_graphics(), label_2d_graphics(), points_2d_graphics(), qq_2d_graphics(), rect_2d_graphics()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

density_2d_grid *Density plot in 2d using the grid package*

Description

Density plot in 2d using the grid package

Usage

```
density_2d_grid(
  zargs,
  ngrids = 25,
  ccol = NULL,
  clwd = 1,
  clty = 1,
  box = FALSE,
  box.width = 1,
  box.height = 1,
  group... = list(cex = 0.66),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
ngrids	number of grid points in each direction. Can be scalar or a length-2 integer vector.
ccol	vector (which is then recycled to the appropriate length) giving the color of the contours
clwd	vector (which is then recycled to the appropriate length) giving the line widths of the contours
clty	vector (which is then recycled to the appropriate length) giving the line types of the contours
box	logical indicating whether a box should be drawn
box.width	width of the box
box.height	height of the box
group...	list of arguments passed to group_2d_grid (or NULL)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Note

- We use names depending on the 'type' here since otherwise, if one calls it once for 'p' and once for 'l', only one of them is plotted - The default point size was chosen to match the default of graphics

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the grid package: arrow_2d_grid(), axes_2d_grid(), group_2d_grid(), label_2d_grid(), points_2d_grid(), qq_2d_grid(), rect_2d_grid()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

density_2d_loon *Density plot in 2d using the interactive loon package*

Description

Density plot in 2d using the interactive loon package

Usage

```
density_2d_loon(
  zargs,
  ngrids = 25,
  ccol = NULL,
  color = NULL,
  clwd = NULL,
  lwd = NULL,
  linewidth = 1,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  linkingGroup = NULL,
  baseplot = NULL,
  parent = NULL,
  group... = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
ngrids	Number of grid points in each direction. Can be scalar or a length-2 integer vector.
ccol	A vector (which is then recycled to the appropriate length) giving the color of the contours
color	Colour used fill if ccol is NULL, a grey palette is used otherwise.
clwd	A vector (which is then recycled to the appropriate length) giving the line widths of the contours
lwd	Line width used only when clwd = NULL
linewidth	Line width used when both clwd and lwd are NULL, value of 1 used otherwise.
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
linkingGroup	The initial linking group
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
group...	A list of arguments passed to group_2d_loon (or NULL)
...	Additional parameters passed to loon::l_layer_line()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the interactive loon package: arrow_2d_loon(), axes_2d_loon(), group_2d_loon(), label_2d_loon(), points_2d_loon(), rect_2d_loon()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

de_elect *German Election Data from 2002 and 2005*

Description

Data set consisting of 68 columns of data about the German elections 2002 and 2005.

Usage

```
data("de_elect")
```

Format

A `data.frame()` with 68 columns:

`District`: electoral district

`State`: federal state (Bundesland)

`Num.comm`: number of communities

`Area`: area 2004-12-31 (in square km)

`Pop`: population 2004-12-31 (in 1000)

`Men`: men (in 1000)

`Citizens`: germans (in 1000)

`Density`: population density 2004-12-31 (in square km)

`Pop.le.15`: population younger than (or equal to) 15 years 2002-12-31 (in percent)

`Pop.15.18`: population between 15 and 18 years old 2002-12-31 (in percent)

`Pop.18.25`: population between 18 and 25 years old 2002-12-31 (in percent)

`Pop.25.35`: population between 25 and 35 years old 2002-12-31 (in percent)

`Pop.35.60`: population between 35 and 60 years old 2002-12-31 (in percent)

`Pop.g.60`: population older than 60 years 2002-12-31 (in percent)

`Births`: live births (per 1000)

`Deaths`: deaths (per 1000)

`Move.in`: moving there in 2003 (per 1000)

`Move.out`: moving away in 2003 (per 1000)

`Increase`: increase in population (per 1000)

`Farms`: number of farms in 2001 (per 1000)

`Agriculture`: agriculturally used land (in ha)

`Mining`: mining companies and processing trade 2002-09-30 (per 1000)

`Mining.employees`: employees in mining and processing trade 2002-09-30 (per 1000)

`Apt.new`: new apartments 2002 (per 1000)

`Apt`: apartments 2002-12-31 (per 1000)

`Motorized`: motor vehicles 2003-01-31 (per 1000)

`School.finishers`: school finishers 2002 (per 1000)

`School.wo.2nd`: without secondary school (ohne Hauptschule) 2002 (in percent)

`School.2nd`: with secondary school (Hauptschule) 2002 (in percent)

`School.Real`: with graduation from Realschule 2002 (in percent)

`School.UED`: with university-entrance diploma (Gymnasium) 2002 (in percent)

`Unemployment.03`: unemployment 2003-12-31 (in percent)

`Unemployment.04`: unemployment 2004-12-31 (in percent)

`Employed`: employed subject to social insurance contribution (per 1000)

`FFF`: farmers, foresters, fishermen (in percent)

`Industry`: industry employees subject to social insurance contribution (in percent)

`CTT`: commerce, transportation and telecommunication employees subject to social insurance contribution (in percent)

`OS`: other services (in percent)

`Voters.05`: eligible voters 2005

`Voters.02`: eligible voters 2002

`Votes.05`: number of votes 2005

`Votes.02`: number of votes 2002

`Invalid.05`: invalid votes 2005

`Invalid.02`: invalid votes 2002

`Valid.05`: valid votes 2005

`Valid.02`: valid votes 2002

`Votes.SPD.05`: votes for SPD 2005

`Votes.SPD.02`: votes for SPD 2002

`Votes.CDU.CSU.05`: votes for CDU/CSU 2005

`Votes.CDU.CSU.02`: votes for CDU/CSU 2002

`Votes.Gruene.05`: votes for Gruene 2005

`Votes.Gruene.02`: votes for Gruene 2002

`Votes.FDP.05`: votes for FDP 2005

`Votes.FDP.02`: votes for FDP 2002

`Votes.Linke.05`: votes for Linke 2005

`Votes.Linke.02`: votes for Linke 2002

`SPD.05`: SPD 2005 (as a fraction in [0,1])

`CDU.CSU.05`: CDU/CSU 2005 (as a fraction in [0,1])

`Gruene.05`: Gruene 2005 (as a fraction in [0,1])

`FDP.05`: FDP 2005 (as a fraction in [0,1])

`Linke.05`: Linke 2005 (as a fraction in [0,1])

`Others.05`: Other parties 2005 (as a fraction in [0,1])

`SPD.02`: SPD 2002 (as a fraction in [0,1])

`CDU.CSU.02`: CDU/CSU 2002 (as a fraction in [0,1])

`Gruene.02`: Gruene 2002 (as a fraction in [0,1])

`FDP.02`: FDP 2002 (as a fraction in [0,1])

`Linke.02`: Linke 2002 (as a fraction in [0,1])

`Others.02`: other parties 2002 (as a fraction in [0,1])

Source

The data was obtained from http://www.bundeswahlleiter.de but is not available under this link anymore. Furthermore, the first column of the original data set is ommitted as it only contained the row numbers.

Examples

```
data("de_elect")
```

---

extract_1d *Extracting information for our default/provided plot1d()*

Description

Extracting information for our default/provided plot1d()

Usage

```
extract_1d(zargs)
```

Arguments

`zargs`	The argument list as passed from `zenplot()`. This must at least contain `x`, `orientations`, `vars`, `num`, `lim` and `labs`; see `zenplot()` for an explanation of these variables.

Details

This is an auxiliary function called on `zargs` within any 1d plotting function (e.g. `hist_1d_grid`, `density_1d_graphics`, or `points_1d_loon`) to extract the 1d data from `zargs` needed for plotting. For performance reasons, no checking of the input object is done.

Value

A list `list` with

`x`: the data to be plotted in the 1d plot

`xcols`: a list with all columns of `x`

`groups`: the group numbers for each column of `x`

`vars`: the variable numbers for each column of `x`

`glabs`: the group labels for each column of `x`

`vlabs`: the variable labels for each column of `x`

`horizontal`: a `logical` indicating whether the plot is horizontal or vertical, and

`xlim`: the axis limits.

Note

Performance critical

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other tools for constructing your own plot1d and plot2d functions: `burst()`, `burst_aux()`, `check_zargs()`, `extract_2d()`, `plot_indices()`

Other data extraction functions to build plots: `extract_2d()`

Other default 1d plot functions: `arrow_1d_graphics()`, `arrow_1d_grid()`, `arrow_1d_loon()`, `boxplot_1d_graphics()`, `boxplot_1d_grid()`, `boxplot_1d_loon()`, `density_1d_graphics()`, `density_1d_grid()`, `density_1d_loon()`, `hist_1d_graphics()`, `hist_1d_grid()`, `hist_1d_loon()`, `jitter_1d_graphics()`, `jitter_1d_grid()`, `jitter_1d_loon()`, `label_1d_graphics()`, `label_1d_grid()`, `label_1d_loon()`, `lines_1d_graphics()`, `lines_1d_grid()`, `lines_1d_loon()`, `points_1d_graphics()`, `points_1d_grid()`, `points_1d_loon()`, `rect_1d_graphics()`, `rect_1d_grid()`, `rect_1d_loon()`, `rug_1d_graphics()`, `rug_1d_grid()`, `rug_1d_loon()`

Examples

```
## This function is used within the default (any user defined)
## 1d plots
my_1d_plot <- function(zargs, your_name = "Bob", ...) {
                  data_1d <- extract_1d(zargs)
                  msg <- paste("Components of zargs available",
                               "to construct a 1d plot for ",
                               your_name)
                  print(msg)
                  ## just print the names of the data components
                  ## which you might want to use in your plot
                  print(names(data_1d))
                  ## You might have to draw your 1d plot differently depending
```

```
                    ## upon whether it is to appear horizontally or vertically
                    if (data_1d$horizontal) {
                           print("This plot would be horizontal")
                           } else {
                           print("This one would be vertical")
                     }
                     ## You can plot whatever you want using the information in
                     ## could use any of these to construct any 1d plot you want
                     ## using R's graphics or any of zemplot's built in 1d plots.
                     ##
                     ## For example, here we use zenplot's base graphics functions
                     ## First a histogram
                     hist_1d_graphics(zargs, ...)
                     ## to which we add the variable label
                     label_1d_graphics(zargs, add = TRUE, col = "red", ...)
                     ## similar functions could be called for the other packages.
                     ## You can print the source of anyone of the default functions
                     ## to get some idea of managing details.
                     }

## And now try it out
zenplot(iris[,1:3], plot1d = my_1d_plot)
```

`extract_2d` *Extracting information for our default/provided plot2d()*

Description

Extracting information for our default/provided plot2d()

Usage

```
extract_2d(zargs)
```

Arguments

`zargs`	The argument list as passed from `zenplot()`. This must at least contain `x`, `vars`, `num`, `lim` and `labs` (for `extract_2d()`); see `zenplot()` for an explanation of these variables.

Details

This is an auxiliary function called on `zargs` within any 1d plotting function (e.g. `hist_1d_grid`, `density_1d_graphics`, or `points_1d_loon`) to extract the 1d data from `zargs` needed for plotting. For performance reasons, no checking of the input object is done.

Value

A list `list` with

`x` **and** `y`**:** the data to be plotted in the 2d plot

`xcols`**:** a list with all columns of `x`

`groups`**:** the group numbers for each column of `x`

`vars`**:** the variable numbers for each column of `x`

`glabs`**:** the group labels for each column of `x`

`vlabs`**:** the variable labels for each column of `x`

`xlim` **and** `ylim`**:** the x-axis and y-axis limits, and

`same.group`**:** a `logical` indicating whether the x and y variables belong to the same group.

Note

Performance critical

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other tools for constructing your own plot1d and plot2d functions: `burst()`, `burst_aux()`, `check_zargs()`, `extract_1d()`, `plot_indices()`

Other data extraction functions to build plots: `extract_1d()`

Other default 2d plot functions: `arrow_2d_graphics()`, `arrow_2d_grid()`, `arrow_2d_loon()`, `axes_2d_graphics()`, `axes_2d_grid()`, `axes_2d_loon()`, `density_2d_graphics()`, `density_2d_grid()`, `density_2d_loon()`, `group_2d_graphics()`, `group_2d_grid()`, `group_2d_loon()`, `label_2d_graphics()`, `label_2d_grid()`, `label_2d_loon()`, `points_2d_graphics()`, `points_2d_grid()`, `points_2d_loon()`, `qq_2d_graphics()`, `qq_2d_grid()`, `rect_2d_graphics()`, `rect_2d_grid()`, `rect_2d_loon()`

Examples

```
## This function is used within the default (any user defined)
## 2d plot functions
##
my_2d_plot <- function(zargs, your_name = "BillyBob", ...) {
                   data_2d <- extract_2d(zargs)
                   msg <- paste("Components of zargs available",
                                "to construct a 2d plot for ",
                                your_name)
                   print(msg)
                   ## just print the names of the data components
                   ## which you might want to use in your plot
                   print(names(data_2d))

                   ## You can plot whatever you want using the information in
                   ## could use any of these to construct any 1d plot you want
```

```
                    ## using R's graphics or any of zemplot's built in 1d plots.
                    ##
                    ## For example, here we could use
                    ## use zenplot's base graphics functions
                    ## First a scatterplot
                    points_2d_graphics(zargs, ...)
                    ## to which we overlay density contours
                    density_2d_graphics(zargs, add = TRUE, col = "steelblue", ...)
                    ## similar functions could be called for the other packages.
                    ## You can print the source of anyone of the default functions
                    ## to get some idea of managing details.
                  }

## And now try it out
zenplot(iris, plot2d = my_2d_plot)
```

`extract_pairs` *Extract Pairs from a Path of Indices*

Description

Extracts pairs from a path of indices, representing the path by the pairs (connected by common variable) and return a shortened path.

Usage

```
extract_pairs(x, n)
```

Arguments

`x`	the path, a `vector` or `list` of indices of the variables to be plotted.
`n`	A `vector` of length two giving the number of pairs to extract from the path `x` (if `NULL`, all pairs are returned (nothing extracted); if of length one, it is replicated in the pair). The first number corresponds to the beginning of the path, the second to the end; at least one of the two numbers should be >= 1.

Details

This function is self-contained and has no external package dependencies.

Value

returns an object of the same type as the input `x` but (possibly) shortened. It extracts the first/last so-many pairs of `x`.

Author(s)

Marius Hofert and Wayne Oldford

See Also

zenplot() which provides the zenplot.

Other tools related to constructing zenpaths: connect_pairs(), graph_pairs(), groupData(), indexData(), zenpath()

Examples

```
(zp <- zenpath(c(3, 5), method = "eulerian.cross"))
extract_pairs(zp, n = c(2, 4))
extract_pairs(zp, n = 3)
```

---

get_layout *Compute the layout of the zen plot*

Description

Compute the layout of the zen plot

Usage

```
get_layout(
  turns,
  n2dplots,
  first1d = TRUE,
  last1d = TRUE,
  width1d = 1,
  width2d = 10
)
```

Arguments

turns	turns (character vector consisting if "u", "d", "l", "r")
n2dplots	the number of 2d plots (faces of the hypercube to be laid out)
first1d	logical indicating whether the first 1d plot should be plotted
last1d	logical indicating whether the last 1d plot should be plotted
width1d	width of 1d plots
width2d	width of 2d plots

Value

list containing 1) the plot orientations (c("h", "s", "v", "s", ...)) 2) the plot dimensions (1d plot, 2d plot, 1d plot, ...) 3) the variable numbers plotted (an (nPlots, 2)-matrix) 4) the total width of the layout 5) the total height of the layout 6) coordinates of the bounding boxes

Author(s)

Marius Hofert and Wayne Oldford

`get_path` *Computing the path according to the provided method*

Description

Computing the path according to the provided method

Usage

```
get_path(
  turns = NULL,
  n2dcols = c("letter", "square", "A4", "golden", "legal"),
  n2dplots,
  method = c("tidy", "double.zigzag", "single.zigzag", "rectangular"),
  first1d = TRUE,
  last1d = TRUE
)
```

Arguments

`turns`	The turns
`n2dcols`	The number of columns of 2d plots (>= 1) or one of "letter", "square", "A4", "golden", "legal". Note that n2dcols is ignored if turns is not NULL.
`n2dplots`	The number of 2d plots to be laid out
`method`	A character string indicating the method according to which the path is built
`first1d`	A logical indicating whether the first 1d plot should be plotted
`last1d`	A logical indicating whether the last 1d plot should be plotted

Value

the path, a list containing the turns, the positions (indices in the occupancy matrix) and the the occupancy matrix

Author(s)

Marius Hofert and Wayne Oldford

get_zigzag_turns *Compute turns for zigzag*

Description

Compute turns for zigzag

Usage

```
get_zigzag_turns(
  nPlots,
  n2dcols,
  method = c("tidy", "double.zigzag", "single.zigzag")
)
```

Arguments

nPlots	total number of plots
n2dcols	number of columns of 2d plots (>= 1)
method	character string indicating which zigzag method to use

Value

turns

Author(s)

Marius Hofert and Wayne Oldford

graph_pairs *Turn pairs or paths into a graph*

Description

Pairs are processed to produce a graph with the elements of the pairs as vertices and the pairs as undirected edges. The result can be displayed using plot() if the Bioconductor packages **graph** (for the graphNEL object) and optionally **Rgraphviz** (for plotting) are installed.

Usage

```
graph_pairs(x, var.names = NULL, edgemode = c("undirected", "directed"))
```

Arguments

x	matrix or list of pairs along a zenpath. Can also be a list containing vectors representing paths in the graph. Every path must be of length at least 2 (i.e. each vector element of the list).
var.names	names of the variables appearing in x.
edgemode	type of edges to be used: either "undirected" (the default) or "directed" (in which case the order of the nodes in each pair matters).

Value

a graphNEL object; can be displayed using plot() if **Rgraphviz** is installed.

Note

This function requires the Bioconductor package **graph**. If **graph** is not installed, an informative error is raised. Plotting also requires **Rgraphviz**, but all functionality of graph_pairs() is available without it.

Author(s)

Marius Hofert and Wayne Oldford

See Also

zenplot() which provides the zenplot.

Other tools related to constructing zenpaths: connect_pairs(), extract_pairs(), groupData(), indexData(), zenpath()

Examples

```
has_Rgraphviz <- requireNamespace("Rgraphviz", quietly = TRUE)
pairs <- matrix(c(1,2, 5,1, 3,4, 2,3, 4,2), ncol = 2, byrow = TRUE)
g <- graph_pairs(pairs)
if (has_Rgraphviz){
  Rgraphviz::plot(g)
}
```

---

groupData *Splitting a Matrix into a List of Matrices*

Description

Takes a matrix x and groups its rows (or columns) as specified by indices. Returns a list of matrices, one for each group.

Usage

```
groupData(x, indices, byrow = FALSE)
```

Arguments

x	A matrix (or an object convertible to such via as.matrix()).
indices	list of vectors of indices according to which x is grouped; each vector of indices define a group.
byrow	logical indicating whether the grouping is done by row (byrow = TRUE) or by column (byrow = FALSE, the default).

Details

No external packages are required.

Value

A list of matrices (one per group). Such a list, grouped by columns, is then typically passed on to zenplot().

Author(s)

Marius Hofert and Wayne Oldford

See Also

zenplot() which provides the zenplot.

Other tools related to constructing zenpaths: connect_pairs(), extract_pairs(), graph_pairs(), indexData(), zenpath()

Examples

```
## get a matrix
x <- matrix(1:15, ncol = 3)
colGroups <- list(c(1,2), list(2:3))
rowGroups <- list(c(1,4), list(2:3))
groupData(x, indices = colGroups)
groupData(x, indices = rowGroups, byrow = TRUE)
```

---

group_2d_graphics *Plot of labels indicating adjacent groups using R's base graphics*

Description

Plot of labels indicating adjacent groups using R's base graphics

Usage

```
group_2d_graphics(
  zargs,
  glabs = NULL,
  sep = "\n",
  loc = c(0.5, 0.5),
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
glabs	group labels being indexed by the plot variables (and thus of length as the number of variables); if NULL then they are determined with extract_2d()
sep	group label separator
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
add	logical indicating whether this plot should be added to the last one
plot...	additional arguments passed to plot_region()
...	additional arguments passed to text()

Value

invisible()

Note

For performance reasons (avoiding having to call extract_2d() twice), 'glabs' is an extra argument

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using R's base graphics: arrow_2d_graphics(), axes_2d_graphics(), density_2d_graphics(), label_2d_graphics(), points_2d_graphics(), qq_2d_graphics(), rect_2d_graphics()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

group_2d_grid *Plot of labels indicating adjacent groups using the grid package*

Description

Plot of labels indicating adjacent groups using the grid package

Usage

```
group_2d_grid(
  zargs,
  glabs = NULL,
  sep = "\n",
  loc = c(0.5, 0.5),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
glabs	group labels being indexed by the plot variables (and thus of length as the number of variables); if NULL then they are determined with extract_2d()
sep	group label separator
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Note

For performance reasons (avoiding having to call extract_2d() twice), 'glabs' is an extra argument

Author(s)

Marius Hofert

See Also

Other default 2d plot functions using the grid package: arrow_2d_grid(), axes_2d_grid(), density_2d_grid(), label_2d_grid(), points_2d_grid(), qq_2d_grid(), rect_2d_grid()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

`group_2d_loon` *Plot of labels indicating adjacent groups using the interactive loon package*

Description

Plot of labels indicating adjacent groups using the interactive loon package

Usage

```
group_2d_loon(
  zargs,
  glabs = NULL,
  sep = "\n",
  size = 8,
  rot = 0,
  baseplot = NULL,
  parent = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
glabs	group labels being indexed by the plot variables (and thus of length as the number of variables); if NULL then they are determined with extract_2d()
sep	group label separator
size	plot size
rot	rotation
baseplot	If non-NULL the base plot on which the plot should be layered
parent	tk parent for this loon plot widget
...	Additional arguments passed to text()

Value

invisible()

Note

For performance reasons (avoiding having to call extract_2d() twice), 'glabs' is an extra argument

Author(s)

Marius Hofert & Wayne Oldford

See Also

Other default 2d plot functions using the interactive loon package: `arrow_2d_loon()`, `axes_2d_loon()`, `density_2d_loon()`, `label_2d_loon()`, `points_2d_loon()`, `rect_2d_loon()`

Other default 2d plot functions: `arrow_2d_graphics()`, `arrow_2d_grid()`, `arrow_2d_loon()`, `axes_2d_graphics()`, `axes_2d_grid()`, `axes_2d_loon()`, `density_2d_graphics()`, `density_2d_grid()`, `density_2d_loon()`, `extract_2d()`, `group_2d_graphics()`, `group_2d_grid()`, `label_2d_graphics()`, `label_2d_grid()`, `label_2d_loon()`, `points_2d_graphics()`, `points_2d_grid()`, `points_2d_loon()`, `qq_2d_graphics()`, `qq_2d_grid()`, `rect_2d_graphics()`, `rect_2d_grid()`, `rect_2d_loon()`

`happiness` *World Happiness Data Set*

Description

Data set consisting of 498 rows and 12 columns containing data from the World Happiness Report over three years.

Usage

```
data("happiness")
```

Format

`data.frame()` with 12 columns:

`Time`: year of the World Happiness Report.

`Region`: region of the world.

`Country`: country.

`Happiness`: happiness score measured in the respective year (see `Time`) by asking "How would you rate your happiness on a scale of 0 to 10 where 10 is happiest?".

`Rank`: rank of the country based on `Happiness`.

`GDP`: extent to which the gross domestic product per capita contributed to the calculation of `Happiness`.

`Family`: extent to which family contributed to the calculation of `Happiness`.

`Health`: extent to which life expectancy contributed to the calculation of `Happiness`.

`Freedom`: extent to which freedom contributed to the calculation of `Happiness`.

`Corruption`: extent to which the perception of corruption contributed to the calculation of `Happiness`.

`Generosity`: extent to which generosity contributed to the calculation of `Happiness`.

`Dystopia`: extent to which the dystopia residual contributed to the calculation of `Happiness`. Dystopia is an imaginary country with the world's least-happy people (which can act as a benchmark against which all countries can be favorably compared).

Details

GDP, Family, Health, Freedom, Corruption and Generosity describe the extent to which these factors contribute in evaluating the happiness in each country. If added together with Dystopia, one receives the happiness score.

The data set is based on the World Happiness Report (https://www.worldhappiness.report/) and derived from the public tables provided with the report and provided by Kaggle on 2018-04-20 (historical mirrors such as Kaggle may require an account and change over time, so we refer to the official site above). Downloaded from Kaggle as three separate .csv files (one for each year), joint columns (variables) were then built, rows expanded (to be the same for each year) and sorted according to Region and Country, and Time was added to obtain a single data set.

References

Helliwell, Layard, Sachs, et al. World Happiness Report (annual). Available at https://www.worldhappiness.report/.

Examples

```
data("happiness")
stopifnot(all.equal(rowSums(happiness[,c("GDP", "Family", "Health", "Freedom",
                                         "Corruption", "Generosity",
                                         "Dystopia")]),
                    happiness[, "Happiness"], tol = 5e-5))
```

---

hist_1d_graphics *Histogram as 1d plot using R's base graphics*

Description

Histogram as 1d plot using R's base graphics

Usage

```
hist_1d_graphics(
  zargs,
  breaks = NULL,
  length.out = 21,
  col = NULL,
  axes = FALSE,
  add = TRUE,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
breaks	see ?hist; the default is 20 equi-width bins covering the data range
length.out	number of break points if breaks = NULL
col	vector of colors for the bars or bar components; see ?barplot
axes	logicial indicating whether axes should be drawn
add	logical indicating whether this plot should be added to the last one
plot...	additional arguments passed to plot_region()
...	additional arguments passed to barplot()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: arrow_1d_graphics(), boxplot_1d_graphics(), density_1d_graphics(), jitter_1d_graphics(), label_1d_graphics(), lines_1d_graphics(), points_1d_graphics(), rect_1d_graphics(), rug_1d_graphics()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

hist_1d_grid *Histogram in 1d using the grid package*

Description

Histogram in 1d using the grid package

Usage

```
hist_1d_grid(
  zargs,
  breaks = NULL,
  length.out = 21,
  col = NULL,
  fill = NULL,
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
breaks	see ?hist; the default is 20 equi-width bins covering the data range
length.out	number of break points if breaks = NULL
col	colour of the histogram bar interiors, unless fill is specified, then this is the colour of the border
fill	logical passed to the underlying rectGrob()
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), boxplot_1d_grid(), density_1d_grid(), jitter_1d_grid(), label_1d_grid(), lines_1d_grid(), points_1d_grid(), rect_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

hist_1d_loon *Histogram in 1d using the interactive loon package*

Description

Histogram in 1d using the interactive loon package

Usage

```
hist_1d_loon(
  zargs,
  breaks = NULL,
  color = NULL,
  fill = NULL,
  showStackedColors = TRUE,
  showBinHandle = FALSE,
  showLabels = FALSE,
  linkingGroup = NULL,
  showScales = FALSE,
  showGuides = FALSE,
  parent = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
breaks	Argument passed to hist() to get information on bins. Default is 20 equi-width bins covering the range of x
color	colour of the histogram bar interiors, unless fill is specified, then this is the colour of the border
fill	colour of the histogram bar interior if given
showStackedColors	Logical determining whether to show the individual point colours stacked in the histogram
showBinHandle	Logical to show a handle to adjust bins
showLabels	Logical determining whether axis labels are displayed
linkingGroup	A string specifying the initial group of plots to be linked to this plot
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
parent	The tk parent for this loon plot widget
...	Additional parameters passed to loon::l_hist()

Value

A loon loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), boxplot_1d_loon(), density_1d_loon(), jitter_1d_loon(), label_1d_loon(), lines_1d_loon(), points_1d_loon(), rect_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

indexData *Indexing a Matrix or Data Frame According to Given Indices*

Description

Indexing a Matrix or Data Frame According to Given Indices

Usage

```
indexData(x, indices)
```

Arguments

x	A matrix or data.frame (most useful for the latter).
indices	vector of column indices of x (typically obtained from zenpath()).

Details

No external packages are required.

Value

An object as x (typically a data.frame or matrix) containing x indexed by indices.

Note

Useful for constructing data.frames without .1, .2, ... in their names when indexing a data.frame with a zenpath.

Author(s)

Marius Hofert and Wayne Oldford

See Also

`zenplot()` which provides the zenplot.

Other tools related to constructing zenpaths: `connect_pairs()`, `extract_pairs()`, `graph_pairs()`, `groupData()`, `zenpath()`

Examples

```
## The function is handiest for data frames
## where we want to reuse the variable names
## without adding a suffix like ".1" etc.
## For example,
x <-  BOD  # Biochemical Oxygen Demand data in base R
indices <- rep(1:2, 2)
## now compare
indexData(x, indices)
## to
x[, indices]
## zenplots prefer not to have the suffixes.
```

---

is.standard *Check Argument for Being a Vector, Matrix, Data Frame or a List of such*

Description

Check Argument for Being a Vector, Matrix, Data Frame or a List of such

Usage

```
is.standard(x)
```

Arguments

x	A vector, matrix, data.frame or list of such

Value

A logical indicating whether x is of the above type

Author(s)

Marius Hofert

See Also

Other zenplot technical tools: `convert_occupancy()`, `n2dcols_aux()`, `num_cols()`, `turn_checker()`

`jitter_1d_graphics` *Jittered dot plot in 1d using R's base graphics*

Description

Jittered dot plot in 1d using R's base graphics

Usage

```
jitter_1d_graphics(
  zargs,
  loc = 0.5,
  offset = 0.25,
  cex = 0.4,
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

`zargs`	argument list as passed from `zenplot()`
`loc`	location in [0,1]; 0 corresponds to left, 1 to right (in the direction of the path)
`offset`	number in [0,0.5] determining how far off the center the jittered points reach maximally
`cex`	character expansion factor
`add`	logical indicating whether this plot should be added to the last one
`plot...`	additional arguments passed to plot_region()
`...`	additional arguments passed to points()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: `arrow_1d_graphics()`, `boxplot_1d_graphics()`, `density_1d_graphics()`, `hist_1d_graphics()`, `label_1d_graphics()`, `lines_1d_graphics()`, `points_1d_graphics()`, `rect_1d_graphics()`, `rug_1d_graphics()`

Other default 1d plot functions: `arrow_1d_graphics()`, `arrow_1d_grid()`, `arrow_1d_loon()`, `boxplot_1d_graphics()`, `boxplot_1d_grid()`, `boxplot_1d_loon()`, `density_1d_graphics()`,

density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

`jitter_1d_grid` *Jittered dot plot in 1d using the grid package*

Description

Jittered dot plot in 1d using the grid package

Usage

```
jitter_1d_grid(
  zargs,
  loc = 0.5,
  offset = 0.25,
  pch = 21,
  size = 0.02,
  draw = FALSE,
  ...
)
```

Arguments

`zargs`	argument list as passed from `zenplot()`
`loc`	location in [0,1]; 0 corresponds to left, 1 to right (in the direction of the path)
`offset`	number in [0,0.5] determining how far off the center the jittered points reach maximally
`pch`	plotting symbol
`size`	size of the plotting symbol
`draw`	logical indicating whether drawing should take place
`...`	additional arguments passed to gpar()

Value

grob (invisibly)

Note

The default point size was chosen to match the default of graphics

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), boxplot_1d_grid(), density_1d_grid(), hist_1d_grid(), label_1d_grid(), lines_1d_grid(), points_1d_grid(), rect_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

jitter_1d_loon *Jittered dot plot in 1d using the interactive loon package*

Description

Jittered dot plot in 1d using the interactive loon package

Usage

```
jitter_1d_loon(
  zargs,
  linkingGroup = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  glyph = "ocircle",
  itemLabel = NULL,
  showItemLabels = TRUE,
  parent = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
linkingGroup	A string specifying the initial group of plots to be linked to this plot
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
glyph	Glyph to be used for points, default is the open circle: "ocircle"
itemLabel	A vector of strings to serve as the item labels
showItemLabels	Logical determing whether item labels display on mouse hover
parent	The tk parent for this loon plot widget
...	Additional parameters passed to loon::l_plot()

Value

A loon loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), boxplot_1d_loon(), density_1d_loon(), hist_1d_loon(), label_1d_loon(), lines_1d_loon(), points_1d_loon(), rect_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

label_1d_graphics *Label plot in 1d using R's base graphics*

Description

Label plot in 1d using R's base graphics

Usage

```
label_1d_graphics(
  zargs,
  loc = c(0.5, 0.5),
  label = NULL,
  box = FALSE,
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
label	label to be used
box	logical indicating whether a box is to be drawn.

add	logical indicating whether this plot should be added to the last one
plot...	additional arguments passed to plot_region()
...	additional arguments passed to text() and box()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: arrow_1d_graphics(), boxplot_1d_graphics(), density_1d_graphics(), hist_1d_graphics(), jitter_1d_graphics(), lines_1d_graphics(), points_1d_graphics(), rect_1d_graphics(), rug_1d_graphics()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

label_1d_grid *Label plot in 1d using the grid package*

Description

Label plot in 1d using the grid package

Usage

```
label_1d_grid(
  zargs,
  loc = c(0.5, 0.5),
  label = NULL,
  cex = 0.66,
  box = FALSE,
  box.width = 1,
  box.height = 1,
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
label	label to be used
cex	character expansion factor
box	logical indicating whether a box should be drawn around the text
box.width	width of the box
box.height	height of the box
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), boxplot_1d_grid(), density_1d_grid(), hist_1d_grid(), jitter_1d_grid(), lines_1d_grid(), points_1d_grid(), rect_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

label_1d_loon *Label plot in 1d using the interactive loon package*

Description

Label plot in 1d using the interactive loon package

Usage

```
label_1d_loon(
  zargs,
  loc.x = NULL,
  loc.y = NULL,
  label = NULL,
  rot = NULL,
  size = 8,
  box = FALSE,
  color = NULL,
  linkingGroup = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
loc.x	x-location of the label
loc.y	y-location of the label
label	The label to be used
rot	The rotation of the label
size	The font size
box	A logical indicating whether the label is to be enclosed in a box.
color	Color of the label (and of box when box = TRUE).
linkingGroup	A string specifying the initial group of plots to be linked to this plot
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
...	Additional parameters passed to loon::l_layer_text(...)

Value

A loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), boxplot_1d_loon(), density_1d_loon(), hist_1d_loon(), jitter_1d_loon(), lines_1d_loon(), points_1d_loon(), rect_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

label_2d_graphics *Label plot in 2d using R's base graphics*

Description

Label plot in 2d using R's base graphics

Usage

```
label_2d_graphics(
  zargs,
  loc = c(0.98, 0.05),
  label = NULL,
  adj = 1:0,
  box = FALSE,
  add = FALSE,
  group... = NULL,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location (in (0,1)^2) of the center of the rectangle
label	label to be used
adj	x (and optionally y) adjustment of the label
box	logical indicating whether a box should be drawn
add	logical indicating whether this plot should be added to the last one
group...	list of arguments passed to group_2d_graphics (or NULL)
plot...	additional arguments passed to plot_region()
...	additional arguments passed to rect()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using R's base graphics: arrow_2d_graphics(), axes_2d_graphics(), density_2d_graphics(), group_2d_graphics(), points_2d_graphics(), qq_2d_graphics(), rect_2d_graphics()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

label_2d_grid *Label plot in 2d using the grid package*

Description

Label plot in 2d using the grid package

Usage

```
label_2d_grid(
  zargs,
  loc = c(0.98, 0.05),
  label = NULL,
  cex = 0.66,
  just = c("right", "bottom"),
  rot = 0,
  box = FALSE,
  box.width = 1,
  box.height = 1,
  group... = list(cex = cex),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
label	label to be used
cex	character expansion factor
just	(x,y)-justification of the label
rot	rotation of the label
box	logical indicating whether a box should be drawn
box.width	width of the box
box.height	height of the box
group...	list of arguments passed to group_2d_grid (or NULL)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the grid package: arrow_2d_grid(), axes_2d_grid(), density_2d_grid(), group_2d_grid(), points_2d_grid(), qq_2d_grid(), rect_2d_grid()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

label_2d_loon *Label plot in 2d using the interactive loon package*

Description

Label plot in 2d using the interactive loon package

Usage

```
label_2d_loon(
  zargs,
  loc = NULL,
  label = NULL,
  rot = 0,
  size = 8,
  box = FALSE,
  color = NULL,
  linkingGroup = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  group... = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
loc	The location of the label
label	The label to be used
rot	The rotation of the label
size	The font size
box	A logical indicating whether the label is to be enclosed in a box.
color	Color of the label (and of box when box = TRUE).
linkingGroup	The initial linking group
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
group...	A list of arguments passed to group_2d_loon (or NULL)
...	Additional parameters passed to loon::l_layer_text(...)

Value

The base loon::l_plot with the added text layer

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the interactive loon package: arrow_2d_loon(), axes_2d_loon(), density_2d_loon(), group_2d_loon(), points_2d_loon(), rect_2d_loon()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

layout_1d_graphics *Layout plot in 1d*

Description

Layout plot in 1d

Usage

```
layout_1d_graphics(zargs, ...)
```

Arguments

zargs	argument list as passed from zenplot()
...	additional arguments passed to label_1d_graphics()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

---

layout_1d_grid *Layout plot in 1d using the grid package*

Description

Layout plot in 1d using the grid package

Usage

```
layout_1d_grid(zargs, ...)
```

Arguments

zargs	argument list as passed from zenplot()
...	additional arguments passed to label_1d_grid()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

---

layout_1d_loon *Layout plot in 1d using the interactive loon package*

Description

Layout plot in 1d using the interactive loon package

Usage

```
layout_1d_loon(zargs, ...)
```

Arguments

zargs	The argument list as passed from zenplot()
...	Additional arguments passed to label_1d_loon()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

layout_2d_graphics *Layout plot in 2d*

Description

Layout plot in 2d

Usage

```
layout_2d_graphics(zargs, ...)
```

Arguments

zargs	argument list as passed from zenplot()
...	additional arguments passed to label_2d_graphics()

Value

invisible()

Note

Here we also pass '...' to group_2d_grid() (to easily adjust font size etc.)

Author(s)

Marius Hofert and Wayne Oldford

layout_2d_grid *Layout plot in 2d using the grid package*

Description

Layout plot in 2d using the grid package

Usage

```
layout_2d_grid(zargs, ...)
```

Arguments

zargs	argument list as passed from zenplot()
...	additional arguments passed to label_2d_grid()

Value

grob (invisibly)

Note

Here we also pass '...' to group_2d_grid() (to easily adjust font size etc.)

Author(s)

Marius Hofert and Wayne Oldford

`layout_2d_loon` *Layout plot in 2d using the interactive loon package*

Description

Layout plot in 2d using the interactive loon package

Usage

```
layout_2d_loon(zargs, ...)
```

Arguments

`zargs`	The argument list as passed from `zenplot()`
`...`	Additional arguments passed to label_2d_grid()

Value

A loon plot

Note

Here we also pass '...' to group_2d_loon() (to easily adjust font size etc.)

Author(s)

Marius Hofert and Wayne Oldford

`lines_1d_graphics` *Line plot in 1d using R's base graphics*

Description

Line plot in 1d using R's base graphics

Usage

```
lines_1d_graphics(
  zargs,
  loc = c(0.5, 0.5),
  length = 1,
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

`zargs`	argument list as passed from `zenplot()`
`loc`	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
`length`	length of the line (in [0,1])
`add`	logical indicating whether this plot should be added to the last one
`plot...`	additional arguments passed to plot_region()
`...`	additional arguments passed to lines()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: `arrow_1d_graphics()`, `boxplot_1d_graphics()`, `density_1d_graphics()`, `hist_1d_graphics()`, `jitter_1d_graphics()`, `label_1d_graphics()`, `points_1d_graphics()`, `rect_1d_graphics()`, `rug_1d_graphics()`

Other default 1d plot functions: `arrow_1d_graphics()`, `arrow_1d_grid()`, `arrow_1d_loon()`, `boxplot_1d_graphics()`, `boxplot_1d_grid()`, `boxplot_1d_loon()`, `density_1d_graphics()`, `density_1d_grid()`, `density_1d_loon()`, `extract_1d()`, `hist_1d_graphics()`, `hist_1d_grid()`, `hist_1d_loon()`, `jitter_1d_graphics()`, `jitter_1d_grid()`, `jitter_1d_loon()`, `label_1d_graphics()`, `label_1d_grid()`, `label_1d_loon()`, `lines_1d_grid()`, `lines_1d_loon()`, `points_1d_graphics()`,

points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

lines_1d_grid *Lines plot in 1d using the grid package*

Description

Lines plot in 1d using the grid package

Usage

```
lines_1d_grid(
  zargs,
  loc = c(0.5, 0.5),
  length = 1,
  arrow = NULL,
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
length	length of the line (in [0,1])
arrow	list describing the arrow head
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), boxplot_1d_grid(), density_1d_grid(), hist_1d_grid(), jitter_1d_grid(), label_1d_grid(), points_1d_grid(), rect_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

lines_1d_loon *Lines plot in 1d using the interactive loon package*

Description

Lines plot in 1d using the interactive loon package

Usage

```
lines_1d_loon(
  zargs,
  loc.x = NULL,
  loc.y = NULL,
  color = NULL,
  lwd = 1,
  linkingGroup = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
loc.x	x-coordinates of the points on the line
loc.y	y-coordinates of the pointson the line
color	Colour of the line
lwd	line width
linkingGroup	A string specifying the initial group of plots to be linked to this plot (ignored)

showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
...	Additional parameters passed to loon::l_layer_text(...)

Value

A loon loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), boxplot_1d_loon(), density_1d_loon(), hist_1d_loon(), jitter_1d_loon(), label_1d_loon(), points_1d_loon(), rect_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

l_ispace_config *Configuring a loon plot to accommodate ispace*

Description

Configuring a loon plot to accommodate ispace

Usage

```
l_ispace_config(
  baseplot,
  ispace = NULL,
  x = NULL,
  y = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

baseplot	The plot to be modified
ispace	The inner space (in [0,1])
x	The x data
y	The y data
xlim	The x-axis limits; if NULL, the data limits are used
ylim	The y-axis limits; if NULL, the data limits are used
...	Additional arguments passed to loon::l_configure

Value

The baseplot

Author(s)

R. W. Oldford

See Also

Other graphical tools: na_omit_loon(), plot_region(), vport(), zenarrow()

---

move *Determine the new position when moving from the current position in a given direction*

Description

Determine the new position when moving from the current position in a given direction

Usage

```
move(curpos, dir, method = c("in.occupancy", "in.plane"))
```

Arguments

curpos	current position (i, j) in the occupancy matrix
dir	direction in which we move ("d", "u", "r" or "l")
method	choice of method ("in.occupancy" means the (current/new) position is given in terms of (row, column) indices in the occupancy matrix; "in.plane" means the directions are interpreted as in the (x,y)-plane).

Value

new position in the occupancy matrix

Author(s)

Marius Hofert and Wayne Oldford

`n2dcols_aux` *Auxiliary Function for Constructing Default n2dcols*

Description

Auxiliary Function for Constructing Default n2dcols

Usage

```
n2dcols_aux(n2dplots, method = c("letter", "square", "A4", "golden", "legal"))
```

Arguments

`n2dplots`	The number of variates (= nfaces)
`method`	One of "letter", "square", "A4", "golden", "legal"

Value

An odd integer for n2dcols

Author(s)

Wayne Oldford

See Also

Other zenplot technical tools: `convert_occupancy()`, `is.standard()`, `num_cols()`, `turn_checker()`

`na_omit_loon` *Helper function to remove NAs for loon plots*

Description

Helper function to remove NAs for loon plots

Usage

```
na_omit_loon(x, y = NULL, linkingKey = NULL, itemLabel = NULL)
```

Arguments

x	The vector of x values (required)
y	The vector of y values (optional) of the same length as x; if NULL then it's ignored.
linkingKey	The vector of keys used to define links between points, of the same length as x; if NULL it will be 0:(length(x)-1).
itemLabel	The vector of labels for the points, of the same length as x; if NULL it will be constructed.

Value

A list(x, y, linkingKey, itemLabel) where any NA in x or y will have been omitted from all

Author(s)

R. W. Oldford

See Also

Other graphical tools: `l_ispace_config()`, `plot_region()`, `vport()`, `zenarrow()`

---

next_move_tidy *Determine the next position to move to and the turn out of there*

Description

Determine the next position to move to and the turn out of there

Usage

```
next_move_tidy(plotNo, nPlots, curpath)
```

Arguments

plotNo	current plot number
nPlots	total number of plots
curpath	the current path

Value

a list containing the next position to move to (nextpos) and the turn out of there (nextout); Interpretation: nextpos: position of plot number plotNo+1 in the (non-trimmed) occupancy matrix nextout: turn out of nextpos

Note

- This assumes that the last plot is a 1d plot! - It also assumes that first1d = TRUE; will be adapted later in get_path() in case first1d = FALSE. - We start in (1, 2) and also have an additional last column in the occupancy matrix to have the first and last column left in case we end up there with the last 1d plot; this cannot happen for 'zigzag' but for 'tidy'.

Author(s)

Marius Hofert and Wayne Oldford

`num_cols` *Determine the number of columns if is.standard(x)*

Description

Determine the number of columns if is.standard(x)

Usage

```
num_cols(x)
```

Arguments

`x`	A numeric vector, matrix, data.frame or a list of such.

Value

The number of data columns of 'x'

Author(s)

Marius Hofert

See Also

Other zenplot technical tools: `convert_occupancy()`, `is.standard()`, `n2dcols_aux()`, `turn_checker()`

olive *Olive Oil Data Set*

Description

Data set consisting of 572 rows and 10 columns containing data about olive oil.

Usage

```
data("olive")
```

Format

A `data.frame()` with 10 columns:

`Area`**:** (larger) area.

`Region`**:** (local) region.

`palmitic`**,** `palmitoleic`**,** `stearic`**,** `oleic`**,** `linoleic`**,** `linolenic`**,** `arachidic`**,** `eicosenoic`**:** the fatty acids measured.

Source

The data set was obtained from the package **pdfCluster** (for convenience). It contains 572 rows of observations. The first and the second column correspond to the area (Centre-North, South, Sardinia) and the geographical region of origin of the olive oils (northern Apulia, southern Apulia, Calabria, Sicily, inland Sardinia and coast Sardinia, eastern and western Liguria, Umbria), respectively. The remaining columns represent the chemical measurements (on the acid components for the oil specimens) palmitic, palmitoleic, stearic, oleic, linoleic, linolenic, arachidic, eicosenoic.

Examples

```
data("olive")
```

plot_exists *Check whether functions (plot*d to zenplot()) exist*

Description

Check whether functions (plot*d to zenplot()) exist

Usage

```
plot_exists(x)
```

Arguments

x	arguments plot1d or plot2d of zenplot()

Value

logical indicating whether x exists

Note

Check first whether it's a function (have to rely on it being able to be evaluated, cannot do more checks then) or, if a string, whether it exists

Author(s)

Marius Hofert

---

plot_indices *Plot Indices of the Current Plot*

Description

Determining the indices of the x and y variables of the current plot

Usage

```
plot_indices(zargs)
```

Arguments

zargs	argument list as passed from zenplot(). This must at least contain vars and num; see zenplot() for an explanation of these variables..

Details

This is an auxiliary function useful, for example, when writing user-provided 1d and 2d plot functions.

Value

A numeric(2) containing the indices of the x and y variables to be plotted in the current plot (the plot with number num). If the current plot is a 2d plot, the same variable is used twice.

Note

This is exported so that one doesn't always have to figure out whether the variables (axes) in the current plot need to be switched manually.

Author(s)

Marius Hofert

See Also

Other tools for constructing your own plot1d and plot2d functions: burst(), burst_aux(), check_zargs(), extract_1d(), extract_2d()

---

plot_region *Function to set up the plot region for graphics plots*

Description

Auxiliary function for setting up the plot region of 1d and 2d graphics plots.

Usage

```
plot_region(xlim, ylim, plot... = NULL)
```

Arguments

xlim	x-axis limits
ylim	y-axis limits
plot...	arguments passed to the underlying plot()

Details

This is an auxiliary function used by the provided **graphics**-related 1d and 2d plots.

Value

invisible()

Author(s)

Marius Hofert

See Also

Other graphical tools: l_ispace_config(), na_omit_loon(), vport(), zenarrow()

points_1d_graphics *Dot plot in 1d using R's base graphics*

Description

Dot plot in 1d using R's base graphics

Usage

```
points_1d_graphics(
  zargs,
  loc = 0.5,
  cex = 0.4,
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	location in [0,1]; 0 corresponds to left, 1 to right (in the direction of the path)
cex	character expansion factor
add	logical indicating whether this plot should be added to the last one
plot...	additional arguments passed to plot_region()
...	additional arguments passed to points()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: arrow_1d_graphics(), boxplot_1d_graphics(), density_1d_graphics(), hist_1d_graphics(), jitter_1d_graphics(), label_1d_graphics(), lines_1d_graphics(), rect_1d_graphics(), rug_1d_graphics()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

points_1d_grid *Dot plot in 1d using the grid package*

Description

Dot plot in 1d using the grid package

Usage

```
points_1d_grid(zargs, loc = 0.5, pch = 21, size = 0.02, draw = FALSE, ...)
```

Arguments

zargs	argument list as passed from zenplot()
loc	location in [0,1]; 0 corresponds to left, 1 to right (in the direction of the path)
pch	plotting symbol
size	size of the plotting symbol
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

invisible()

Note

The default point size was chosen to match the default of graphics

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), boxplot_1d_grid(), density_1d_grid(), hist_1d_grid(), jitter_1d_grid(), label_1d_grid(), lines_1d_grid(), rect_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

points_1d_loon *Dot plot in 1d using the interactive loon package*

Description

Dot plot in 1d using the interactive loon package

Usage

```
points_1d_loon(
  zargs,
  linkingGroup = NULL,
  linkingKey = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  glyph = "ocircle",
  itemLabel = NULL,
  showItemLabels = TRUE,
  parent = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
linkingGroup	A string specifying the initial group of plots to be linked to this plot
linkingKey	List of IDs to link on
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
glyph	The plot glyph
itemLabel	A vector of strings to serve as the item labels
showItemLabels	Logical determing whether item labels display on mouse hover
parent	The tk parent for this loon plot widget
...	Additional parameters passed to loon::l_plot()

Value

A loon loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), boxplot_1d_loon(), density_1d_loon(), hist_1d_loon(), jitter_1d_loon(), label_1d_loon(), lines_1d_loon(), rect_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

points_2d_graphics *Point plot in 2d using R's base graphics*

Description

Point plot in 2d using R's base graphics

Usage

```
points_2d_graphics(
  zargs,
  cex = 0.4,
  box = FALSE,
  add = FALSE,
  group... = NULL,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
cex	character expansion factor
box	logical indicating whether a box should be drawn
add	logical indicating whether this plot should be added to the last one
group...	list of arguments passed to group_2d_graphics (or NULL)
plot...	additional arguments passed to plot_region()
...	additional arguments passed to points()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using R's base graphics: arrow_2d_graphics(), axes_2d_graphics(), density_2d_graphics(), group_2d_graphics(), label_2d_graphics(), qq_2d_graphics(), rect_2d_graphics()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

`points_2d_grid` *Point plot in 2d using the grid package*

Description

Point plot in 2d using the grid package

Usage

```
points_2d_grid(
  zargs,
  type = c("p", "l", "o"),
  pch = NULL,
  size = 0.02,
  box = FALSE,
  box.width = 1,
  box.height = 1,
  group... = list(cex = 0.66),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
type	line type
pch	plot symbol
size	size of the plot symbol
box	logical indicating whether a box should be drawn
box.width	width of the box
box.height	height of the box

group...	list of arguments passed to group_2d_grid (or NULL)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Note

- We use names depending on the 'type' here since otherwise, if one calls it once for 'p' and once for 'l', only one of them is plotted - The default point size was chosen to match the default of graphics

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the grid package: arrow_2d_grid(), axes_2d_grid(), density_2d_grid(), group_2d_grid(), label_2d_grid(), qq_2d_grid(), rect_2d_grid()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

points_2d_loon *Point plot in 2d using the interactive loon package*

Description

Point plot in 2d using the interactive loon package

Usage

```
points_2d_loon(
  zargs,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  linkingGroup = NULL,
  linkingKey = NULL,
  glyph = "ocircle",
  itemLabel = NULL,
  showItemLabels = TRUE,
  parent = NULL,
```

```
  group... = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
linkingGroup	The initial linking group
linkingKey	List of IDs to link on
glyph	String determining the glyph type to be displayed for points, default is an open circle: "ocircle"
itemLabel	A vector of strings to serve as the item label
showItemLabels	Logical determing whether item labels display on mouse hover
parent	The tk parent for this loon plot widget
group...	A list of arguments passed to group_2d_loon (or NULL)
...	Additional arguments passed to loon::l_plot()

Value

A loon plot

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the interactive loon package: arrow_2d_loon(), axes_2d_loon(), density_2d_loon(), group_2d_loon(), label_2d_loon(), rect_2d_loon()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

qq_2d_graphics *Quantile-quantile plot in 2d using R's base graphics*

Description

Quantile-quantile plot in 2d using R's base graphics

Usage

```
qq_2d_graphics(
  zargs,
  do.line = TRUE,
  lines... = NULL,
  cex = 0.4,
  box = FALSE,
  add = FALSE,
  group... = NULL,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
do.line	logical indicating whether a line is drawn (through both empirical c(0.25, 0.75)-quantiles)
lines...	additional arguments passed to lines()
cex	character expansion factor
box	logical indicating whether a box should be drawn
add	logical indicating whether this plot should be added to the last one
group...	list of arguments passed to group_2d_graphics (or NULL)
plot...	additional arguments passed to plot_region()
...	additional arguments passed to qqplot()

Value

invisible()

Note

line iff both margins are of the same *type*

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using R's base graphics: arrow_2d_graphics(), axes_2d_graphics(), density_2d_graphics(), group_2d_graphics(), label_2d_graphics(), points_2d_graphics(), rect_2d_graphics()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

`qq_2d_grid` *Quantile-quantile plot in 2d using the grid package*

Description

Quantile-quantile plot in 2d using the grid package

Usage

```
qq_2d_grid(
  zargs,
  do.line = TRUE,
  lines... = NULL,
  pch = NULL,
  size = 0.02,
  box = FALSE,
  box.width = 1,
  box.height = 1,
  group... = list(cex = 0.66),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
do.line	logical indicating whether a line is drawn (through both empirical c(0.25, 0.75)-quantiles)
lines...	additional arguments passed to lines()
pch	plot symbol
size	size of the plot symbol
box	logical indicating whether a box should be drawn
box.width	width of the box
box.height	height of the box

group...	list of arguments passed to group_2d_grid (or NULL)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Note

- line iff both margins are of the same *type* - The default point size was chosen to match the default of graphics

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the grid package: arrow_2d_grid(), axes_2d_grid(), density_2d_grid(), group_2d_grid(), label_2d_grid(), points_2d_grid(), rect_2d_grid()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), rect_2d_graphics(), rect_2d_grid(), rect_2d_loon()

---

rect_1d_graphics *Rectangle plot in 1d using R's base graphics*

Description

Rectangle plot in 1d using R's base graphics

Usage

```
rect_1d_graphics(
  zargs,
  loc = c(0.5, 0.5),
  width = 1,
  height = 1,
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location in [0,1]^2; 0 corresponds to left, 1 to right (in the direction of the path)
width	width of the rectangle (when viewed in walking direction)
height	height of the rectangle (when viewed in walking direction)
add	logical indicating whether this plot should be added to the last one
plot...	additional arguments passed to plot_region()
...	additional arguments passed to lines()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: arrow_1d_graphics(), boxplot_1d_graphics(), density_1d_graphics(), hist_1d_graphics(), jitter_1d_graphics(), label_1d_graphics(), lines_1d_graphics(), points_1d_graphics(), rug_1d_graphics()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

rect_1d_grid *Rectangle plot in 1d using the grid package*

Description

Rectangle plot in 1d using the grid package

Usage

```
rect_1d_grid(
  zargs,
  loc = c(0.5, 0.5),
  width = 1,
  height = 1,
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location of the rectangle
width	width of the rectangle (when viewed in walking direction)
height	height of the rectangle (when viewed in walking direction)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), boxplot_1d_grid(), density_1d_grid(), hist_1d_grid(), jitter_1d_grid(), label_1d_grid(), lines_1d_grid(), points_1d_grid(), rug_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

rect_1d_loon *Rectangle plot in 1d using the interactive loon package*

Description

Rectangle plot in 1d using the interactive loon package

Usage

```
rect_1d_loon(
  zargs,
  loc.x = NULL,
  loc.y = NULL,
  color = NULL,
  fill = NULL,
  lwd = 1,
  linkingGroup = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
loc.x	x-location of rectangle
loc.y	y-location of rectangle
color	Colour of the rectangle outline
fill	Colour of the rectangle interior
lwd	line width for rectangle outline
linkingGroup	A string specifying the initial group of plots to be linked to this plot (ignored)
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
baseplot	If non-NULL the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
...	Additional parameters passed to loon::l_layer_text(...)

Value

A loon loon::l_plot(...)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), boxplot_1d_loon(), density_1d_loon(), hist_1d_loon(), jitter_1d_loon(), label_1d_loon(), lines_1d_loon(), points_1d_loon(), rug_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rug_1d_graphics(), rug_1d_grid(), rug_1d_loon()

---

rect_2d_graphics *Rectangle plot in 2d using R's base graphics*

Description

Rectangle plot in 2d using R's base graphics

Usage

```
rect_2d_graphics(
  zargs,
  loc = c(0.5, 0.5),
  width = 1,
  height = 1,
  add = FALSE,
  group... = NULL,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location (in (0,1)^2) of the center of the rectangle
width	width of the rectangle as a fraction of 1
height	height of the rectangle as a fraction of 1
add	logical indicating whether this plot should be added to the last one
group...	list of arguments passed to group_2d_graphics (or NULL)
plot...	additional arguments passed to plot_region()
...	additional arguments passed to rect()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using R's base graphics: arrow_2d_graphics(), axes_2d_graphics(), density_2d_graphics(), group_2d_graphics(), label_2d_graphics(), points_2d_graphics(), qq_2d_graphics()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_grid(), rect_2d_loon()

---

rect_2d_grid *Rectangle plot in 2d using the grid package*

Description

Rectangle plot in 2d using the grid package

Usage

```
rect_2d_grid(
  zargs,
  loc = c(0.5, 0.5),
  width = 1,
  height = 1,
  group... = list(cex = 0.66),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	(x,y)-location of the rectangle
width	rectangle width as a fraction of 1
height	rectangle height as a fraction of 1
group...	list of arguments passed to group_2d_grid (or NULL)
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the grid package: arrow_2d_grid(), axes_2d_grid(), density_2d_grid(), group_2d_grid(), label_2d_grid(), points_2d_grid(), qq_2d_grid()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_loon()

---

rect_2d_loon *Rectangle plot in 2d using the interactive loon package*

Description

Rectangle plot in 2d using the interactive loon package

Usage

```
rect_2d_loon(
  zargs,
  loc.x = NULL,
  loc.y = NULL,
  color = NULL,
  fill = NULL,
  lwd = 1,
  linkingGroup = NULL,
  showLabels = FALSE,
  showScales = FALSE,
  showGuides = FALSE,
  baseplot = NULL,
  parent = NULL,
  group... = NULL,
  ...
)
```

Arguments

zargs	The argument list as passed from zenplot()
loc.x	x-location of rectangle
loc.y	y-location of rectangle
color	Colour of the rectangle outline
fill	Colour of the rectangle interior
lwd	line width for rectangle outline
linkingGroup	The initial linking group (ignored)
showLabels	Logical determining whether axis labels are displayed
showScales	Logical determining whether scales are displayed
showGuides	Logical determining whether the background guidelines are displayed
baseplot	If non-null the base plot on which the plot should be layered
parent	The tk parent for this loon plot widget
group...	A list of arguments passed to group_2d_loon (or NULL)
...	Additional parameters passed to loon::l_layer_text(...)

Value

The base loon::l_plot with the added text layer

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 2d plot functions using the interactive loon package: arrow_2d_loon(), axes_2d_loon(), density_2d_loon(), group_2d_loon(), label_2d_loon(), points_2d_loon()

Other default 2d plot functions: arrow_2d_graphics(), arrow_2d_grid(), arrow_2d_loon(), axes_2d_graphics(), axes_2d_grid(), axes_2d_loon(), density_2d_graphics(), density_2d_grid(), density_2d_loon(), extract_2d(), group_2d_graphics(), group_2d_grid(), group_2d_loon(), label_2d_graphics(), label_2d_grid(), label_2d_loon(), points_2d_graphics(), points_2d_grid(), points_2d_loon(), qq_2d_graphics(), qq_2d_grid(), rect_2d_graphics(), rect_2d_grid()

rug_1d_graphics *Rug plot in 1d using R's base graphics*

Description

Rug plot in 1d using R's base graphics

Usage

```
rug_1d_graphics(
  zargs,
  loc = 0.5,
  length = 0.5,
  width = 1,
  col = par("fg"),
  add = FALSE,
  plot... = NULL,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	location in [0,1]; 0 corresponds to left, 1 to right (in the direction of the path)
length	length of the rugs
width	line width of the rugs
col	color of the rugs
add	logical indicating whether this plot should be added to the last one
plot...	additional arguments passed to plot_region()
...	additional arguments passed to segments()

Value

invisible()

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using R's base graphics: arrow_1d_graphics(), boxplot_1d_graphics(), density_1d_graphics(), hist_1d_graphics(), jitter_1d_graphics(), label_1d_graphics(), lines_1d_graphics(), points_1d_graphics(), rect_1d_graphics()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_grid(), rug_1d_loon()

---

rug_1d_grid *Rug plot in 1d using the grid package*

Description

Rug plot in 1d using the grid package

Usage

```
rug_1d_grid(
  zargs,
  loc = 0.5,
  length = 0.5,
  width = 0.001,
  col = par("fg"),
  draw = FALSE,
  ...
)
```

Arguments

zargs	argument list as passed from zenplot()
loc	location in [0,1]; 0 corresponds to left, 1 to right (in the direction of the path)
length	length of the rugs
width	line width of the rugs
col	default color of the rectangles/rugs
draw	logical indicating whether drawing should take place
...	additional arguments passed to gpar()

Value

grob (invisibly)

Note

The choice of width and height is to leave the rugs enough space to not touch points (so to avoid points and rugs overplotting).

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the grid package: arrow_1d_grid(), boxplot_1d_grid(), density_1d_grid(), hist_1d_grid(), jitter_1d_grid(), label_1d_grid(), lines_1d_grid(), points_1d_grid(), rect_1d_grid()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_loon()

---

rug_1d_loon *Rug plot in 1d using the interactive loon package*

Description

Rug plot in 1d using the interactive loon package

Usage

```
rug_1d_loon(zargs, ...)
```

Arguments

zargs	The argument list as passed from zenplot()
...	Additional parameters passed to loon::l_plot()

Value

A loon loon::l_plot(...)

Note

Just calls points_1d_loon with glyph = "osquare" to preserve linking

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other default 1d plot functions using the interactive loon package: arrow_1d_loon(), boxplot_1d_loon(), density_1d_loon(), hist_1d_loon(), jitter_1d_loon(), label_1d_loon(), lines_1d_loon(), points_1d_loon(), rect_1d_loon()

Other default 1d plot functions: arrow_1d_graphics(), arrow_1d_grid(), arrow_1d_loon(), boxplot_1d_graphics(), boxplot_1d_grid(), boxplot_1d_loon(), density_1d_graphics(), density_1d_grid(), density_1d_loon(), extract_1d(), hist_1d_graphics(), hist_1d_grid(), hist_1d_loon(), jitter_1d_graphics(), jitter_1d_grid(), jitter_1d_loon(), label_1d_graphics(), label_1d_grid(), label_1d_loon(), lines_1d_graphics(), lines_1d_grid(), lines_1d_loon(), points_1d_graphics(), points_1d_grid(), points_1d_loon(), rect_1d_graphics(), rect_1d_grid(), rect_1d_loon(), rug_1d_graphics(), rug_1d_grid()

---

turn_checker *Check the Turns (Number/Type)*

Description

Check the Turns (Number/Type)

Usage

```
turn_checker(turns, n2dplots, first1d, last1d)
```

Arguments

turns	The turns
n2dplots	The number of 2d plots
first1d	A logical indicating whether the first 1d plot should be plotted
last1d	A logical indicating whether the last 1d plot should be plotted

Value

TRUE (unless it fails)

Author(s)

Marius Hofert

See Also

Other zenplot technical tools: convert_occupancy(), is.standard(), n2dcols_aux(), num_cols()

`unfold` *Unfold the hypercube and produce all information concerning the zenpath and zenplot layout*

Description

The `unfold()` function imagines each pair of variables/dimensions as a "face" of a high dimensional cube. These faces are "unfolded" from one 2d space or "face" to the next about the 1d face or "edge" they share. The `unfold()` function takes, as first argument, `nfaces`, the number of 2d plots/spaces to be "unfolded" and produces the zenpath and zenplot layout required for the function zenplot(). Laying out these pairs with a zenplot is what is alluded to as an "unfolding" of (at least a part of) the high dimensional space.

Usage

```
unfold(nfaces, turns = NULL,
       n2dcols = c("letter", "square", "A4", "golden", "legal"),
       method = c("tidy", "double.zigzag", "single.zigzag", "rectangular"),
       first1d = TRUE, last1d = TRUE, width1d = 1, width2d = 10)
```

Arguments

`nfaces`	The number of faces of the hypercube to unfold
`turns`	A `character` vector (of length two times the number of variables to be plotted minus 1) consisting of `"d"`, `"u"`, `"r"` or `"l"` indicating the turns out of the current plot position; if `NULL`, the `turns` are constructed.
`n2dcols`	number of columns of 2d plots (≥ 1) or one of `"letter"`, `"square"`, `"A4"`, `"golden"` or `"legal"` in which case a similar layout is constructed. Note that `n2dcols` is ignored if `!is.null(turns)`.
`method`	The type of zigzag plot (a `character`). Available are: `tidy`: more tidied-up `double.zigzag` (slightly more compact placement of plots towards the end). `double.zigzag`: zigzag plot in the form of a flipped "S". Along this path, the plots are placed in the form of an "S" which is rotated counterclockwise by 90 degrees. `single.zigzag`: zigzag plot in the form of a flipped "S". `rectangular`: plots that fill the page from left to right and top to bottom. This is useful (and most compact) for plots that do not share an axis. Note that `method` is ignored if `turns` are provided.
`first1d`	A `logical` indicating whether the first one-dimensional (1d) plot should be plotted.
`last1d`	A `logical` indicating whether the last one-dimensional (1d) plot should be plotted
`width1d`	A graphical parameter > 0 giving the width of 1d plots.
`width2d`	A graphical parameter > 0 giving the width of 2d plots.

Value

A list describing the unfolded path and its layout as a list of named components:

path: the path of the unfolding, itself given as a structured list having components

- turns: the sequence of turns – each being one of "l" (for left), "r" (for right), "d" (for down), and "u" (for up) – required to move from the current plot location in the display to the next along the unfolded path.
- positions: the path as a matrix of (x, y) positions giving the indices in the occupancy matrix of each plot in the path.
- occupancy: A rectangular array whose cells indicate the positions of the plots on the page.

layout: the details of the visual layout of the plots and given as a structured list having components

- orientations: a vector indicating the orientation of each of the displays in order – "h" for horizontal, "v" for vertical, and "s" for square.
- dimensions: a vector giving the dimensionality of each plot in order.
- vars: A matrix of the variable indices to be used in each plot – x being the horizontal variable and y the vertical.
- layoutWidth: A positive integer giving the display width of a 2d plot.
- layoutHeight: A positive integer giving the display height of a 2d plot.
- boundingBoxes: A matrix of 4 columns giving locations (left, right, bottom, and top) of the box which bound each of the plots in order.

Note

Although unfold() is probably rather rarely used directly by a user, it provides insight into how zenplots are constructed.

Author(s)

Marius Hofert and Wayne Oldford

See Also

Other creating zenplots: zenplot()

Examples

```
dim <- 20
unfolding <- unfold(nfaces = dim -1)
names(unfolding)
```

vport *Viewport Constructing Function for Grid Functions*

Description

Auxiliary function for constructing viewports for 1d and 2d (default) plots.

Usage

```
vport(ispace, xlim = NULL, ylim = NULL, x = NULL, y = NULL, ...)
```

Arguments

ispace	inner space (in $[0, 1]$))
xlim	x-axis limits; if NULL, the data limits are used.
ylim	y-axis limits; if NULL, the data limits are used.
x	x data (only used if is.null(xlim)); if NULL, 0:1 is used.
y	y data (only used if is.null(ylim)); if NULL, 0:1 is used.
...	additional arguments passed to the underlying viewport().

Details

This is an auxiliary function used by the provided **grid**-related 1d and 2d plots.

Value

The viewport.

Note

Ideas from dataViewport() and extendrange() Omitted check: if(length(ispace) != 4) ispace <- rep(ispace, length.out = 4) stopifnot(0 <= ispace, ispace <= 1)

Author(s)

Marius Hofert

See Also

Other graphical tools: l_ispace_config(), na_omit_loon(), plot_region(), zenarrow()

wine *Wine Data Set*

Description

Data set consisting of 178 rows and 27 columns containing data about wine from the Piedmont region of Italy.

Usage

```
data("wine")
```

Format

`data.frame()` with 27 columns:

`wine`: wine name (categorical variable with levels `Barbera`, `Barolo`, `Grignolino`).
`alcohol`: alcohol percentage (numeric).
`sugar`: sugar-free extract (numeric).
`acidity`: fixed acidity (numeric).
`tartaric`: tartaric acid (numeric).
`malic`: malic acid (numeric).
`uronic`: uronic acids (numeric).
`pH`: pH (numeric).
`ash`: ash (numeric).
`alcal_ash`: alcalinity of ash (numeric).
`potassium`: potassium (numeric).
`calcium`: calcium (numeric).
`magnesium`: magnesium (numeric).
`phosphate`: phosphate (numeric).
`cloride`: chloride (numeric).
`phenols`: total phenols (numeric).
`flavanoids`: flavanoids (numeric).
`nonflavanoids`: nonflavanoid phenols (numeric).
`proanthocyanins`: proanthocyanins (numeric).
`colour`: colour intensity (numeric).
`hue`: hue (numeric).
`OD_dw`: OD_{280}/OD_{315} of diluted wines (numeric).
`OD_fl`: OD_{280}/OD_{315} of flavanoids (numeric).
`glycerol`: glycerol (numeric).
`butanediol`: 2,3-butanediol (numeric).
`nitrogen`: total nitrogen (numeric).
`proline`: proline (numeric).
`methanol`: methanol (numeric).

Source

The data set was obtained from the R\ package **sn** (for convenience). It represent chemical measurements on each of 178 wine specimens belonging to three types of wine produced in the Piedmont region of Italy. The data set includes all variables listed by Forina *et al.* (1986) with the exception of 'Sulphate'. The first variable is categorial, all others are numeric.

Forina, M., Lanteri, S. Armanino, C., Casolino, C., Casale, M. and Oliveri, P. V-PARVUS 2008: an extendible package of programs for esplorative data analysis, classification and regression analysis. Dip. Chimica e Tecnologie Farmaceutiche ed Alimentari, Università di Genova, Italia. Web-site (not accessible as of 2014): 'http://www.parvus.unige.it'

References

Forina M., Armanino C., Castino M. and Ubigli M. (1986). Multivariate data analysis as a discriminating method of the origin of wines. *Vitis* **25**, 189–201.

Examples

```
data("wine")
```

---

zenarrow *Defining an arrow*

Description

Defining an arrow

Usage

```
zenarrow(turn, angle = 80, length = 1, coord.scale = 1)
```

Arguments

turn	The direction in which the arrow will point ("l", "r", "d", "u")
angle	The angle
length	The length of the arrow in [0,1] from tip to base
coord.scale	Scale the coordinates of the arrow

Value

A 3-column matrix containing the (x,y) coordinates of the left edge end point, the arrow head and the right edge end point

Author(s)

Marius Hofert

See Also

Other graphical tools: l_ispace_config(), na_omit_loon(), plot_region(), vport()

---

zenpath *Construct a Path of Indices to Order Variables*

Description

Constructing zenpaths and tools for extracting, connecting and displaying pairs, as well as grouping and indexing data structures.

Usage

```
zenpath(x, pairs = NULL,
        method = c("front.loaded", "back.loaded",
                   "balanced", "eulerian.cross",
                   "greedy.weighted", "strictly.weighted"),
        decreasing = TRUE)
```

Arguments

x	for method single integer >= 1. "front.loaded"/"back.loaded": as for method = "front.loaded". "balanced": as for method = "front.loaded". "eulerian.cross": two integers >= 1 representing the group sizes. "greedy.weighted": numeric weight vector (or matrix or distance matrix). "strictly.weighted": as for method = "greedy.weighted".
pairs	a two-column matrix containing (row-wise) the pairs of connected variables to be sorted according to the weights. Note that the resulting graph must be connected (i.e. any variable can be reached from any other variable following the connections given by pairs). The pairs argument is only used for the methods greedy.weighted and strictly.weighted and can be NULL (in which case a default is constructed in lexicographical order).
method	character string indicating the sorting method to be used. Available methods are: "front.loaded": Sort all pairs such that the first variables appear the most frequently early in the sequence; an Eulerian path; note that it might be slightly longer than the number of pairs because, first, an even graph has to be made. "back.loaded": Sort all pairs such that the later variables appear the most frequently later in the sequence; an Eulerian path (+ see front.loaded concerning length)

"balanced": Sort all pairs such that all variables appear in balanced blocks throughout the sequence (a Hamiltonian Decomposition; Eulerian, too).

"eulerian.cross": Generate a sequence of pairs such that each is formed with one variable from each group.

"greedy.weighted": Sort all pairs according to a greedy (heuristic) Euler path with x as weights visiting each edge precisely once.

"strictly.weighted": Strictly respect the order of the weights - so the first, second, third, and so on, adjacent pair of numbers of the output of zenpath() corresponds to the pair with largest, second-largest, third-largest, and so on, weight.

decreasing A logical indicating whether the sorting is done according to increasing or decreasing weights.

Details

Most methods do not require any non-CRAN packages. However, method = "greedy.weighted" constructs an Eulerian path via a graphNEL object and therefore requires Bioconductor package **graph** at runtime. The "strictly.weighted" method does *not* require **graph**.

Value

Returns a sequence of variables (indices or names, possibly a list of such), which can then be used to index the data (via groupData() for plotting via zenplot().

Note

For method = "greedy.weighted" only, **graph** must be installed. If **graph** is unavailable, an informative error should be raised by the function.

Author(s)

Marius Hofert and Wayne Oldford

See Also

zenplot() which provides the zenplot.

Other tools related to constructing zenpaths: connect_pairs(), extract_pairs(), graph_pairs(), groupData(), indexData()

Examples

```
## Methods that do not need Bioconductor:
zenpath(5, method = "front.loaded")
zenpath(5, method = "balanced")

## Greedy weighted (requires Bioconductor 'graph' at runtime):
## Not run:
## if (requireNamespace("graph", quietly = TRUE)) {
##   w <- runif(choose(5, 2))
```

```
##   zp <- zenpath(w, method = "greedy.weighted")
## }
```

---

zenplot *Main function to create a zenplot*

Description

Constructs and draws a zigzag expanded navigation plot for a graphical exploratory analysis of a path of variables. The result is an alternating sequence of one-dimensional (1d) and two-dimensional (2d) plots laid out in a zigzag-like structure so that each consecutive pair of 2d plots has one of its variates (or coordinates) in common with that of the 1d plot appearing between them.

Usage

```
zenplot(x, turns = NULL,
        first1d = TRUE, last1d = TRUE,
        n2dcols = c("letter", "square", "A4", "golden", "legal"),
        n2dplots = NULL,
        plot1d = c("label", "points", "jitter", "density", "boxplot", "hist",
                   "rug", "arrow", "rect", "lines", "layout"),
     plot2d = c("points", "density", "axes", "label", "arrow", "rect", "layout"),
        zargs = c(x = TRUE, turns = TRUE, orientations = TRUE,
                  vars = TRUE, num = TRUE, lim = TRUE, labs = TRUE,
                  width1d = TRUE, width2d = TRUE,
                  ispace = match.arg(pkg) != "graphics"),
        lim = c("individual", "groupwise", "global"),
        labs = list(group = "G", var = "V", sep = ", ", group2d = FALSE),
        pkg = c("graphics", "grid", "loon"),
        method = c("tidy", "double.zigzag", "single.zigzag", "rectangular"),
        width1d = if(is.null(plot1d)) 0.5 else 1,
        width2d = 10,
        ospace = if(pkg == "loon") 0 else 0.02,
        ispace = if(pkg == "graphics") 0 else 0.037,
        draw = TRUE,
        ...)
```

Arguments

x	A data object of "standard forms", being a vector, or a matrix, or a data.frame, or a list of any of these. In the case of a list, the components of x are interpreted as groups of data which are visually separated by a two-dimensional (group) plot.
turns	A character vector (of length two times the number of variables to be plotted minus 1) consisting of "d", "u", "r" or "l" indicating the turns out of the current plot position; if NULL, the turns are constructed (if x is of the "standard form" described above).

first1d — A logical indicating whether the first one-dimensional plot is included.

last1d — A logical indicating whether the last one-dimensional plot is included.

n2dcols — number of columns of 2d plots (≥ 1) or one of "letter", "square", "A4", "golden" or "legal" in which case a similar layout is constructed. Note that n2dcols is ignored if !is.null(turns).

n2dplots — The number of 2d plots.

plot1d — A function to use to return a one-dimensional plot constructed with package pkg. Alternatively, a character string of an existing function. For the defaults provided, the corresponding functions are obtained when appending _1d_graphics, _1d_grid or _1d_loon depending on which pkg is used.

If plot1d = NULL, then no 1d plot is produced in the zenplot.

plot2d — A function returning a two-dimensional plot constructed with package pkg. Alternatively, a character string of an existing function. For the defaults provided, the corresponding functions are obtained when appending _2d_graphics, _2d_grid or _2d_loon depending on which pkg is used.

As for plot1d, plot2d omits 2d plots if plot2d = NULL.

zargs — A fully named logical vector indicating whether the respective arguments are (possibly) passed to plot1d() and plot2d() (if the latter contain the formal argument zargs, which they typically do/should, but see below for an example in which they do not).

zargs can maximally contain all variables as given in the default. If one of those variables does not appear in zargs, it is treated as TRUE and the corresponding arguments are passed on to plot1d and plot2d. If one of them is set to FALSE, the argument is not passed on.

lim — (x-/y-)axis limits. This can be a character string or a numeric(2).

If lim = "groupwise" and x does not contain groups, the behaviour is equivalent to lim = "global".

labs — The plot labels to be used; see the argument labs of burst() for the exact specification. labs can, in general, be anything as long as plot1d and plot2d know how to deal with it.

pkg — The R package used for plotting (depends on how the functions plot1d and plot2d were constructed; the user is responsible for choosing the appropriate package among the supported ones).

method — The type of zigzag plot (a character).

Available are:

tidy: more tidied-up double.zigzag (slightly more compact placement of plots towards the end).

double.zigzag: zigzag plot in the form of a flipped "S". Along this path, the plots are placed in the form of an "S" which is rotated counterclockwise by 90 degrees.

single.zigzag: zigzag plot in the form of a flipped "S".

rectangular: plots that fill the page from left to right and top to bottom. This is useful (and most compact) for plots that do not share an axis.

Note that method is ignored if turns are provided.

width1d	A graphical parameter > 0 giving the width of 1d plots.
width2d	A graphical parameter > 0 giving the height of 2d plots.
ospace	The outer space around the zenplot. A vector of length four (bottom, left, top, right), or one whose values are repeated to be of length four, which gives the outer space between the device region and the inner plot region around the zenplot. Values should be in $[0, 1]$ when pkg is "graphics" or "grid", and as number of pixels whenpkg is "loon".
ispace	The inner space in $[0, 1]$ between the each figure region and the region of the (1d/2d) plot it contains. Again, a vector of length four (bottom, left, top, right) or a shorter one whose values are repeated to produce a vector of length four.
draw	A logical indicating whether a the zenplot is immediately displayed (the default) or not.
...	arguments passed to the drawing functions for both plot1d and plot2d. If you need to pass certain arguments only to one of them, say, plot2d, consider providing your own plot2d; see the examples below.

Value

(besides plotting) invisibly returns a list having additional classnames marking it as a zenplot and a zenPkg object (with Pkg being one of Graphics, Grid, or Loon, so as to identify the package used to construct the plot).

As a list it contains at least the path and layout (see unfold for details).

Depending on the graphics package pkg used, the returned list includes additional components. For pkg = "grid", this will be the whole plot as a grob (grid object). For pkg = "loon", this will be the whole plot as a loon plot object as well as the toplevel tk object in which the plot appears.

Author(s)

Marius Hofert and Wayne Oldford

See Also

All provided default plot1d and plot2d functions.

extract_1d() and extract_2d() for how zargs can be split up into a list of columns and corresponding group and variable information.

burst() for how x can be split up into all sorts of information useful for plotting (see our default plot1d and plot2d). vport() for how to construct a viewport for (our default) **grid** (plot1d and plot2d) functions.

extract_pairs(), connect_pairs(), group() and zenpath() for (zen)path-related functions.

The various vignettes for additional examples.

Other creating zenplots: unfold()

Examples

```
### Basics ######################################################################

## Generate some data
n <- 1000 # sample size
d <- 20 # dimension
set.seed(271) # set seed (for reproducibility)
x <- matrix(rnorm(n * d), ncol = d) # i.i.d. N(0,1) data

## A basic zenplot
res <- zenplot(x)
uf <- unfold(nfaces = d - 1)
## `res` and `uf` is not identical as `res` has specific
## class attributes.
for(name in names(uf)) {
  stopifnot(identical(res[[name]], uf[[name]]))
}

## => The return value of zenplot() is the underlying unfold()

## Some missing data
z <- x
z[seq_len(n-10), 5] <- NA # all NA except 10 points
zenplot(z)

## Another column with fully missing data (use arrows)
## Note: This could be more 'compactified', but is technically
##       more involved
z[, 6] <- NA # all NA
zenplot(z)

## Lists of vectors, matrices and data frames as arguments (=> groups of data)
## Only two vectors
z <- list(x[,1], x[,2])
zenplot(z)

## A matrix and a vector
z <- list(x[,1:2], x[,3])
zenplot(z)

## A matrix, NA column and a vector
z <- list(x[,1:2], NA, x[,3])
zenplot(z)
z <- list(x[,1:2], cbind(NA, NA), x[,3])
zenplot(z)
z <- list(x[,1:2], 1:10, x[,3])
zenplot(z)

## Without labels or with different labels
z <- list(A = x[,1:2], B = cbind(NA, NA), C = x[,3])
zenplot(z, labs = NULL) # without any labels
zenplot(z, labs = list(group = NULL, group2d = TRUE)) # without group labels
```

```
zenplot(z, labs = list(group = NULL)) # without group labels unless groups change
zenplot(z, labs = list(var = NULL)) # without variable labels
zenplot(z, labs = list(var = "Variable ", sep = " - ")) # change default labels

## Example with a factor
zenplot(iris)
zenplot(iris, lim = "global") # global scaling of axis
zenplot(iris, lim = "groupwise") # acts as 'global' here (no groups in the data)


### More sophisticated examples ################################################

## Note: The third component (data.frame) naturally has default labels.
##       zenplot() uses these labels and prepends a default group label.
z <- list(x[,1:5], x[1:10, 6:7], NA,
          data.frame(x[seq_len(round(n/5)), 8:19]), cbind(NA, NA), x[1:10, 20])
zenplot(z, labs = list(group = "Group ")) # change the group label (var and sep are defaults)
## Alternatively, give z labels
names(z) <- paste("Group", LETTERS[seq_len(length(z))]) # give group names
zenplot(z) # uses given group names
## Now let's change the variable labels
z. <- lapply(z, function(z.) {
                    if(!is.matrix(z.)) z. <- as.matrix(z.)
                    colnames(z.) <- paste("Var.", seq_len(ncol(z.)))
                    z.
                    }
            )
zenplot(z.)


### A dynamic plot based on 'loon' (if installed and R compiled with tcl support)

## Not run:
    if(requireNamespace("loon", quietly = TRUE))
        zenplot(x, pkg = "loon")

## End(Not run)


### Providing your own turns ###################################################

## A basic example
turns <- c("l","d","d","r","r","d","d","r","r","u","u","r","r","u","u","l","l",
           "u","u","l","l","u","u","l","l","d","d","l","l","d","d","l","l",
           "d","d","r","r","d","d")
zenplot(x, plot1d = "layout", plot2d = "layout", turns = turns) # layout of plot regions
## => The tiles stick together as ispace = 0.
zenplot(x, plot1d = "layout", plot2d = "layout", turns = turns,
        pkg = "grid") # layout of plot regions with grid
## => Here the tiles show the small (default) ispace

## Another example (with own turns and groups)
zenplot(list(x[,1:3], x[,4:7]), plot1d = "arrow", plot2d = "rect",
```

```
        turns = c("d", "r", "r", "r", "r", "d",
                  "d", "l", "l", "l", "l", "l"), last1d = FALSE)


### Providing your own plot1d() or plot2d() ###################################

## Creating a box
zenplot(x, plot1d = "label", plot2d = function(zargs)
    density_2d_graphics(zargs, box = TRUE))

## With grid

    zenplot(x, plot1d = "label", plot2d = function(zargs)
        density_2d_grid(zargs, box = TRUE), pkg = "grid")


## An example with width1d = width2d and where no zargs are passed on.
## Note: This could have also been done with 'rect_2d_graphics(zargs, col = ...)'
##       as plot1d and plot2d.
myrect <- function(...) {
    plot(NA, type = "n", ann = FALSE, axes = FALSE, xlim = 0:1, ylim = 0:1)
    rect(xleft = 0, ybottom = 0, xright = 1, ytop = 1, ...)
}
zenplot(matrix(0, ncol = 15),
        n2dcol = "square", width1d = 10, width2d = 10,
        plot1d = function(...) myrect(col = "royalblue3"),
        plot2d = function(...) myrect(col = "maroon3"))

## Colorized rugs as plot1d()
basecol <- c("royalblue3", "darkorange2", "maroon3")
palette <- colorRampPalette(basecol, space = "Lab")
cols <- palette(d) # different color for each 1d plot
zenplot(x, plot1d = function(zargs) {
              rug_1d_graphics(zargs, col = cols[(zargs$num+1)/2])
              }
       )

## With grid
library(grid) # for gTree() and gList()

   zenplot(x, pkg = "grid", # you are responsible for choosing the right pkg (cannot be tested!)
             plot1d = function(zargs)
                 rug_1d_grid(zargs, col = cols[(zargs$num+1)/2]))


## Rectangles with labels as plot2d() (shows how to overlay plots)
## With graphics
## Note: myplot2d() could be written directly in a simpler way, but is
##       based on the two functions here to show how they can be combined.
zenplot(x, plot1d = "arrow", plot2d = function(zargs) {
    rect_2d_graphics(zargs)
    label_2d_graphics(zargs, add = TRUE)
})
```

```
## With grid

    zenplot(x, pkg = "grid", plot1d = "arrow", plot2d = function(zargs)
        gTree(children = gList(rect_2d_grid(zargs),
                               label_2d_grid(zargs))))


## Rectangles with labels outside the 2d plotting region as plot2d()
## With graphics
zenplot(x, plot1d = "arrow", plot2d = function(zargs) {
    rect_2d_graphics(zargs)
    label_2d_graphics(zargs, add = TRUE, xpd = NA, srt = 90,
                      loc = c(1.04, 0), adj = c(0,1), cex = 0.7)
})

## With grid

    zenplot(x, pkg = "grid", plot1d = "arrow", plot2d = function(zargs)
        gTree(children = gList(rect_2d_grid(zargs),
                               label_2d_grid(zargs, loc = c(1.04, 0),
                                             just = c("left", "top"),
                                             rot = 90, cex = 0.45))))


## 2d density with points, 1d arrows and labels
zenplot(x, plot1d = function(zargs) {
    rect_1d_graphics(zargs)
    arrow_1d_graphics(zargs, add = TRUE, loc = c(0.2, 0.5))
    label_1d_graphics(zargs, add = TRUE, loc = c(0.8, 0.5))
}, plot2d = function(zargs) {
    points_2d_graphics(zargs, col = adjustcolor("black", alpha.f = 0.4))
    density_2d_graphics(zargs, add = TRUE)
})

## 2d density with labels, 1d histogram with density and label
## Note: The 1d plots are *improper* overlays here as the density
##       plot does not know the heights of the histogram. In other
##       words, both histograms and densities use the whole 1d plot
##       region but are not correct relative to each other in the
##       sense of covering the same are. For a *proper* overlay
##       see below.
zenplot(x,
    plot1d = function(zargs) {
                    hist_1d_graphics(zargs)
                    density_1d_graphics(zargs, add = TRUE,
                                        border = "royalblue3",
                                        lwd = 1.4)
                    label_1d_graphics(zargs, add = TRUE,
                                      loc = c(0.2, 0.8),
                                      cex = 0.6, font = 2,
                                      col = "darkorange2")
                    },
```

```
    plot2d = function(zargs) {
                    density_2d_graphics(zargs)
                    points_2d_graphics(zargs, add = TRUE,
                                       col = adjustcolor("black", alpha.f = 0.3))
                                }
            )


### More sophisticated examples ################################################

### Example: Overlaying histograms with densities (the *proper* way)

## Define proper 1d plot for overlaying histograms with densities
hist_with_density_1d <- function(zargs)
{
## Extract information and data
num <- zargs$num # plot number (among all 1d and 2d plots)
turn.out <- zargs$turns[num] # turn out of current position
horizontal <- turn.out == "d" || turn.out == "u"
# the indices of the 'x' variable to be displayed in the current plot
ii <- plot_indices(zargs)
label <- paste0("V", ii[1]) # label
srt <- if(horizontal) 0 else if(turn.out == "r") -90 else 90 # label rotation
x <- zargs$x[,ii[1]] # data
lim <- range(x) # data limits
## Compute histogram information
breaks <- seq(from = lim[1], to = lim[2], length.out = 21)
binInfo <- hist(x, breaks = breaks, plot = FALSE)
binBoundaries <- binInfo$breaks
widths <- diff(binBoundaries)
heights <- binInfo$density
## Compute density information
dens <- density(x)
xvals <- dens$x
keepers <- (min(x) <= xvals) & (xvals <= max(x)) # keep those within the range of the data
x. <- xvals[keepers]
y. <- dens$y[keepers]
## Determine plot limits and data
if(turn.out == "d" || turn.out == "l") { # flip density/histogram
    heights <- -heights
    y. <- -y.
}
if(horizontal) {
    xlim <- lim
    xlim.bp <- xlim - xlim[1] # special for barplot(); need to shift the bars
    ylim <- range(0, heights, y.)
    ylim.bp <- ylim
    x <- c(xlim[1], x., xlim[2]) - xlim[1] # shift due to plot region set up by barplot()
    y <- c(0, y., 0)
} else {
    xlim <- range(0, heights, y.)
    xlim.bp <- xlim
    ylim <- lim
```

```
    ylim.bp <- ylim - ylim[1] # special for barplot(); need to shift the bars
    x <-  c(0, y., 0)
    y <- c(xlim[1], x., xlim[2]) - ylim[1] # shift due to plot region set up by barplot()
}
## Determining label position relative to the zenpath
loc <- c(0.1, 0.6)

# when walking downwards, change both left/right and up/down
if(turn.out == "d") loc <- 1-loc

# when walking to the right, coordinates change and 2nd is flipped
if(turn.out == "r") {
    loc <- rev(loc)
    loc[2] <- 1-loc[2]
}

# when walking to the left, coordinates change and 1st is flipped
if(turn.out == "l") {
    loc <- rev(loc)
    loc[1] <- 1-loc[1]
}
## Plotting
barplot(heights, width = widths, xlim = xlim.bp, ylim = ylim.bp,
        space = 0, horiz = !horizontal, main = "", xlab = "", axes = FALSE) # histogram
polygon(x = x, y = y, border = "royalblue3", lwd = 1.4) # density
opar <- par(usr = c(0, 1, 0, 1)) # switch to relative coordinates for text
on.exit(par(opar))
text(x = loc[1], y = loc[2], labels = label, cex = 0.7, srt = srt, font = 2,
     col = "darkorange2") # label
    }

## Zenplot
zenplot(x,
plot1d = "hist_with_density_1d",
plot2d = function(zargs) {
       density_2d_graphics(zargs)
       points_2d_graphics(zargs,
                          add = TRUE,
                          col = adjustcolor("black", alpha.f = 0.3))
}
)


### Example: A path through pairs of a grouped t copula sample


## 1) Build a random sample from a 17-dimensional grouped t copula
d. <- c(8, 5, 4) # sector dimensions
d <- sum(d.) # total dimension
nu <- rep(c(12, 1, 0.25), times = d.) # d.o.f. for each dimension
n <- 500 # sample size
set.seed(271)
Z <- matrix(rnorm(n * d), ncol = n) # (d,n)-matrix
```

```
P <- matrix(0.5, nrow = d, ncol = d)
diag(P) <- 1
L <- t(chol(P))   # L: LL^T = P
Y <- t(L %*% Z) # (n,d)-matrix containing n d-vectors following N(0,P)
U. <- runif(n)
W <- sapply(nu, function(nu.) 1/qgamma(U., shape = nu./2, rate = nu./2)) # (n,d)-matrix
X <- sqrt(W) * Y # (n,d)-matrix
U <- sapply(1:d, function(j) pt(X[,j], df = nu[j])) # (n,d)-matrix

## 2) Plot the data with a pairs plot, colorizing the groups
cols <- matrix("black", nrow = d, ncol = d) # colors
start <- c(1, cumsum(head(d., n = -1))+1) # block start indices
end <- cumsum(d.) # block end indices
for(j in seq_along(d.)) cols[start[j]:end[j], start[j]:end[j]] <- basecol[j] # colors
diag(cols) <- NA # remove colors corresponding to diagonal entries
cols <- as.vector(cols) # convert to a vector
cols <- cols[!is.na(cols)] # remove NA entries corresponding to diagonal
count <- 0 # panel number
my_panel <- function(x, y, ...) # panel function for colorizing groups
             { count <<- count + 1; points(x, y, pch = ".", col = cols[count]) }
pairs(U, panel = my_panel, gap = 0,
     labels = as.expression( sapply(1:d, function(j) bquote(italic(U[.(j)]))) ))

## 3) Zenplot of a random path through all pairs, colorizing the respective group
## Define our own points_2d_grid() for colorizing the groups
my_points_2d_grid <- function(zargs, basecol, d.) {
       r <- extract_2d(zargs) # extract information from zargs
       x <- r$x
       y <- r$y
       xlim <- r$xlim
       ylim <- r$ylim
       num2d <- zargs$num/2
       vars <- as.numeric(r$vlabs[num2d:(num2d+1)]) # two variables to be plotted
       ## Alternatively, we could have used ord[r$vars[num2d:(num2d+1)]] with
       ## the order 'ord' (see below) being passed to my_points_2d_grid()
       col <- if(all(1 <= vars & vars <= d.[1])) { basecol[1] } else {
           if(all(d.[1]+1 <= vars & vars <= d.[1]+d.[2])) { basecol[2] } else {
               if(all(d.[1]+d.[2]+1 <= vars & vars <= d)) basecol[3] else "black"
           }
       } # determine the colors
       vp <- vport(zargs$ispace, xlim = xlim, ylim = ylim, x = x, y = y) # viewport
       pointsGrob(x = x[[1]], y = y[[1]], pch = 21, size = unit(0.02, units = "npc"),
                  name = "points_2d", gp = gpar(col = col), vp = vp)
   }
## Plot a random permutation of columns via a zenplot
## Note: We set column labels here, as otherwise the labels can only
##       show *indices* of the variables to be plotted, i.e., the column
##       number in U[,ord], and not the original column number in U (which
##       is what we want to see in order to see how our 'path' through
##       the pairs of variables looks like).
colnames(U) <- 1:d
set.seed(1)
(ord <- sample(1:d, size = d)) # path; 1:d would walk parallel to the secondary diagonal
```

```
zenplot(U[,ord], plot1d = "layout", plot2d = "layout", pkg = "grid") # layout
zenplot(U[,ord], # has correct variable names as column names
        pkg = "grid",
        plot1d = function(zargs) arrow_1d_grid(zargs, col = "grey50"),
        plot2d = function(zargs)
               gTree(children = gList(
                   my_points_2d_grid(zargs, basecol = basecol, d. = d.),
                   rect_2d_grid(zargs, width = 1.05, height = 1.05,
                                col = "grey50", lty = 3),
                   label_2d_grid(zargs, loc = c(1.06, -0.03),
                                 just = c("left", "top"), rot = 90, cex = 0.45,
                                 fontface = "bold") )))
## => The points are colorized correctly (compare with the pairs plot).



### Using ggplot2 ##################################################################

## Although not thoroughly tested, in principle ggplot2 can also be used via
## pkg = "grid" as follows.

library(ggplot2)

## Define our own 2d plot
my_points_2d_ggplot <- function(zargs, extract2d = TRUE)
   {
       if(extract2d) {
           r <- extract_2d(zargs) # extract results from zargs
           df <- data.frame(r$x, r$y) # data frame
           names(df) <- c("x", "y")
           cols <- zargs$x[,"Species"]
       } else {
           ii <- plot_indices(zargs) # the indices of the variables to be plotted
           irs <- zargs$x # iris data
           df <- data.frame(x = irs[,ii[1]], y = irs[,ii[2]]) # data frame
           cols <- irs[,"Species"]
       }
       num2d <- zargs$num/2 # plot number among all 2d plots
       p <- ggplot() + geom_point(data = df, aes(x = x, y = y, colour = cols),
                                  show.legend = num2d == 3) +
           labs(x = "", y = "") # 2d plot
      if(num2d == 3) p <- p + theme(legend.position = "bottom", # legend for last 2d plot
                                    legend.title = element_blank())
       ggplot_gtable(ggplot_build(p)) # 2d plot as grob
   }

## Plotting
iris. <- iris
colnames(iris.) <- gsub("\\\\.", " ", x = colnames(iris)) # => nicer 1d labels
zenplot(iris., n2dplots = 3, plot2d = "my_points_2d_ggplot", pkg = "grid")
zenplot(iris., n2dplots = 3,
        plot2d = function(zargs) my_points_2d_ggplot(zargs, extract2d = FALSE),
        pkg = "grid")
```

```
### Providing your own data structure ##########################################


## Danger zone: An example with a new data structure (here: a list of *lists*)
## Note: - In this case, we most likely need to provide both plot1d and plot2d
##         (but not in this case here since arrow_1d_graphics() does not depend
##         on the data structure)
##       - Note that we still make use of zargs here.
##       - Also note that the variables are not correctly aligned anymore:
##         In the ggplot2 examples we guaranteed this by plot_indices(),
##         but here we don't. This then still produces our layout but the
##         x/y axis of adjacent plots might not be the same anymore. This is
##         fine if only a certain order of the plots is of interest, but
##         not a comparison between adjacent plots.
z <- list(list(1:5, 2:1, 1:3), list(1:5, 1:2))
zenplot(z, n2dplots = 4, plot1d = "arrow", last1d = FALSE,
        plot2d = function(zargs, ...) {
            r <- unlist(zargs$x, recursive = FALSE)
            num2d <- zargs$num/2 # plot number among 2d plots
            x <- r[[num2d]]
            y <- r[[num2d + 1]]
            if(length(x) < length(y)) x <- rep(x, length.out = length(y))
            else if(length(y) < length(x)) y <- rep(y, length.out = length(x))
            plot(x, y, type = "b", xlab = "", ylab = "")
        }, ispace = c(0.2, 0.2, 0.1, 0.1))



### Zenplots based on 3d lattice plots #########################################


library(lattice)
library(grid)
library(gridExtra)

## Build a list of cloud() plots (trellis objects)
## Note:
## - 'grid' problem: Without print(), the below zenplot() may fail (e.g.,
##   in fresh R sessions) with: 'Error in UseMethod("depth") :
##   no applicable method for 'depth' applied to an object of class "NULL"'
## - col = "black" inside scales is needed to make the ticks show
mycloud <- function(x, num) {
       lim <- extendrange(0:1, f = 0.04)
       print(cloud(x[, 3] ~ x[, 1] * x[, 2], xlim = lim, ylim = lim, zlim = lim,
                   xlab = substitute(U[i.], list(i. = num)),
                   ylab = substitute(U[i.], list(i. = num + 1)),
                   zlab = substitute(U[i.], list(i. = num + 2)),
                   zoom = 1, scales = list(arrows = FALSE, col = "black"),
                   col = "black",
                   par.settings = list(standard.theme(color = FALSE),
```

```
                                        axis.line = list(col = "transparent"),
                                        clip = list(panel = "off"))))
   }
plst.3d <- lapply(1:4, function(i)
       mycloud(x[,i:(i+2)], num = i)) # list of trellis objects

## Preparing the zenplot
num <- length(plst.3d)
ncols <- 2
turns <- c(rep("r", 2*(ncols-1)), "d", "d",
           rep("l", 2*(ncols-1)), "d")
plot2d <- function(zargs) {
       num2d <- (zargs$num+1)/2
       vp <- vport(zargs$ispace, xlim = 0:1, ylim = 0:1)
      grob(p = zargs$x[[num2d]], vp = vp, cl = "lattice") # convert trellis to grid object
       ## Note: For further plots, Work with
       ##       gTree(children = gList(grob(zargs$x[[num2d]], vp = vp,
       ##                                   cl = "lattice")))
   }

## Zenplot
## Note: We use a list of *plots* here already (not data)
zenplot(plst.3d, turns = turns, n2dplots = num, pkg = "grid", first1d = FALSE,
        last1d = FALSE, plot1d = "arrow_1d_grid", plot2d = plot2d)
```

Index

∗ **creating zenplots**
unfold, 102
zenplot, 109
∗ **data extraction functions to build plots**
extract_1d, 35
extract_2d, 37
∗ **datagen**
burst, 20
plot_indices, 79
∗ **datasets**
de_elect, 33
happiness, 48
olive, 78
wine, 105
∗ **default 1d plot functions using R's base graphics**
arrow_1d_graphics, 4
boxplot_1d_graphics, 16
density_1d_graphics, 25
hist_1d_graphics, 49
jitter_1d_graphics, 55
label_1d_graphics, 58
lines_1d_graphics, 70
points_1d_graphics, 81
rect_1d_graphics, 90
rug_1d_graphics, 98
∗ **default 1d plot functions using the grid package**
arrow_1d_grid, 5
boxplot_1d_grid, 17
density_1d_grid, 26
hist_1d_grid, 50
jitter_1d_grid, 56
label_1d_grid, 59
lines_1d_grid, 71
points_1d_grid, 82
rect_1d_grid, 91
rug_1d_grid, 99
∗ **default 1d plot functions using the interactive loon package**
arrow_1d_loon, 7
boxplot_1d_loon, 18
density_1d_loon, 27
hist_1d_loon, 52
jitter_1d_loon, 57
label_1d_loon, 60
lines_1d_loon, 72
points_1d_loon, 83
rect_1d_loon, 93
rug_1d_loon, 100
∗ **default 1d plot functions**
arrow_1d_graphics, 4
arrow_1d_grid, 5
arrow_1d_loon, 7
boxplot_1d_graphics, 16
boxplot_1d_grid, 17
boxplot_1d_loon, 18
density_1d_graphics, 25
density_1d_grid, 26
density_1d_loon, 27
extract_1d, 35
hist_1d_graphics, 49
hist_1d_grid, 50
hist_1d_loon, 52
jitter_1d_graphics, 55
jitter_1d_grid, 56
jitter_1d_loon, 57
label_1d_graphics, 58
label_1d_grid, 59
label_1d_loon, 60
lines_1d_graphics, 70
lines_1d_grid, 71
lines_1d_loon, 72
points_1d_graphics, 81
points_1d_grid, 82
points_1d_loon, 83
rect_1d_graphics, 90
rect_1d_grid, 91

rect_1d_loon, 93
rug_1d_graphics, 98
rug_1d_grid, 99
rug_1d_loon, 100
∗ **default 2d plot functions using R's base graphics**
arrow_2d_graphics, 8
axes_2d_graphics, 12
density_2d_graphics, 28
group_2d_graphics, 44
label_2d_graphics, 62
points_2d_graphics, 84
qq_2d_graphics, 88
rect_2d_graphics, 94
∗ **default 2d plot functions using the grid package**
arrow_2d_grid, 9
axes_2d_grid, 13
density_2d_grid, 30
group_2d_grid, 46
label_2d_grid, 63
points_2d_grid, 85
qq_2d_grid, 89
rect_2d_grid, 95
∗ **default 2d plot functions using the interactive loon package**
arrow_2d_loon, 10
axes_2d_loon, 14
density_2d_loon, 31
group_2d_loon, 47
label_2d_loon, 64
points_2d_loon, 86
rect_2d_loon, 96
∗ **default 2d plot functions**
arrow_2d_graphics, 8
arrow_2d_grid, 9
arrow_2d_loon, 10
axes_2d_graphics, 12
axes_2d_grid, 13
axes_2d_loon, 14
density_2d_graphics, 28
density_2d_grid, 30
density_2d_loon, 31
extract_2d, 37
group_2d_graphics, 44
group_2d_grid, 46
group_2d_loon, 47
label_2d_graphics, 62
label_2d_grid, 63
label_2d_loon, 64
points_2d_graphics, 84
points_2d_grid, 85
points_2d_loon, 86
qq_2d_graphics, 88
qq_2d_grid, 89
rect_2d_graphics, 94
rect_2d_grid, 95
rect_2d_loon, 96
∗ **dplot**
plot_region, 80
vport, 104
∗ **graphical tools**
l_ispace_config, 73
na_omit_loon, 75
plot_region, 80
vport, 104
zenarrow, 106
∗ **hplot**
zenplot, 109
∗ **tools for constructing your own plot1d and plot2d functions**
burst, 20
burst_aux, 21
check_zargs, 22
extract_1d, 35
extract_2d, 37
plot_indices, 79
∗ **tools related to constructing zenpaths**
connect_pairs, 23
extract_pairs, 39
graph_pairs, 42
groupData, 43
indexData, 53
zenpath, 107
∗ **utilities**
convert_occupancy, 24
∗ **zenplot technical tools**
convert_occupancy, 24
is.standard, 54
n2dcols_aux, 75
num_cols, 77
turn_checker, 101

adjust_bb, 4
arrow_1d_graphics, 4, *6*, *8*, *17–19*, *26–28*, *36*, *50*, *51*, *53*, *55*, *57–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *98–101*

arrow_1d_grid, *5*, 5, *8*, *17–19*, *26–28*, *36*, *50*, *51*, *53*, *55*, *57–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *99–101*
arrow_1d_loon, *5*, *6*, 7, *17–19*, *26–28*, *36*, *50*, *51*, *53*, *55*, *57–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *99–101*
arrow_2d_graphics, 8, *10*, *11*, *13*, *14*, *16*, *29*, *31*, *32*, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
arrow_2d_grid, *9*, 9, *11*, *13*, *14*, *16*, *29*, *31*, *32*, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
arrow_2d_loon, *9*, *10*, 10, *13*, *14*, *16*, *29*, *31*, *32*, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
as.matrix, *44*
as_numeric, 11
axes_2d_graphics, *9–11*, 12, *14*, *16*, *29*, *31*, *32*, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
axes_2d_grid, *9–11*, *13*, 13, *16*, *29*, *31*, *32*, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
axes_2d_loon, *9–11*, *13*, *14*, 14, *29*, *31*, *32*, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*

boxplot_1d_graphics, *5*, *6*, *8*, 16, *18*, *19*, *26–28*, *36*, *50*, *51*, *53*, *55*, *57–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *98–101*
boxplot_1d_grid, *5*, *6*, *8*, *17*, 17, *19*, *26–28*, *36*, *50*, *51*, *53*, *55*, *57–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *99–101*
boxplot_1d_loon, *5*, *6*, *8*, *17*, *18*, 18, *26–28*, *36*, *50*, *51*, *53*, *55*, *57–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *99–101*
burst, 20, *22*, *23*, *36*, *38*, *80*, *110*, *111*
burst_aux, *21*, 21, *23*, *36*, *38*, *80*

character, *24*, *102*, *107*, *109*, *110*
check_zargs, *21*, *22*, 22, *36*, *38*, *80*
connect_pairs, 23, *40*, *43*, *44*, *54*, *108*, *111*
convert_occupancy, 24, *54*, *75*, *77*, *101*

data.frame, *20*, *23*, *33*, *48*, *53*, *78*, *105*, *109*
de_elect, 33
density_1d_graphics, *5*, *6*, *8*, *17–19*, 25, *27*, *28*, *35–37*, *50*, *51*, *53*, *55*, *57–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *98–101*
density_1d_grid, *5*, *6*, *8*, *17–19*, *26*, 26, *28*, *36*, *50*, *51*, *53*, *56–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *99–101*
density_1d_loon, *5*, *6*, *8*, *17–19*, *26*, *27*, 27, *36*, *50*, *51*, *53*, *56–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *99–101*
density_2d_graphics, *9–11*, *13*, *14*, *16*, 28, *31*, *32*, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
density_2d_grid, *9–11*, *13*, *14*, *16*, *29*, 30, *32*, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
density_2d_loon, *9–11*, *13*, *14*, *16*, *29*, *31*, 31, *38*, *45*, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*

extract_1d, *5*, *6*, *8*, *17–19*, *21–23*, *26–28*, 35, *38*, *50*, *51*, *53*, *56–60*, *62*, *70*, *72*, *73*, *80–82*, *84*, *91*, *92*, *94*, *99–101*, *111*
extract_2d, *9–11*, *13*, *14*, *16*, *21–23*, *29*, *31*, *32*, *36*, 37, *45*, *47*, *48*, *63*, *64*, *66*, *80*, *85–87*, *89*, *90*, *95–97*, *111*
extract_pairs, *24*, 39, *43*, *44*, *54*, *108*, *111*

function, *110*

get_layout, 40
get_path, 41
get_zigzag_turns, 42
graph_pairs, *24*, *40*, 42, *44*, *54*, *108*
graphNEL, *43*, *108*
grob, *111*
group, *111*
group_2d_graphics, *9–11*, *13*, *14*, *16*, *29*, *31*, *32*, *38*, 44, *47*, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
group_2d_grid, *9–11*, *13*, *14*, *16*, *29*, *31*, *32*, *38*, *45*, 46, *48*, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
group_2d_loon, *9–11*, *13*, *14*, *16*, *29*, *31*, *32*, *38*, *45*, *47*, 47, *63*, *64*, *66*, *85–87*, *89*, *90*, *95–97*
groupData, *24*, *40*, *43*, 43, *54*, *108*

happiness, 48
hist_1d_graphics, *5*, *6*, *8*, *17–19*, *26–28*, *36*, 49, *51*, *53*, *55–60*, *62*, *70*, *72*, *73*, *81*, *82*, *84*, *91*, *92*, *94*, *98–101*

hist_1d_grid, *5, 6*, *8*, *17–19*, *26–28*, *35–37*, *50*, 50, *53*, *56–60*, *62*, *70*, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *99–101*
hist_1d_loon, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, 52, *56–60*, *62*, *70*, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *99–101*

indexData, *24*, *40*, *43, 44*, 53, *108*
integer, *107*
invisible, *80*
is.standard, *24*, 54, *75*, *77*, *101*

jitter_1d_graphics, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, 55, *57–60*, *62*, *70*, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *98–101*
jitter_1d_grid, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56*, 56, *58–60*, *62*, *70*, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *99–101*
jitter_1d_loon, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56, 57*, 57, *59, 60*, *62*, *70*, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *99–101*

l_ispace_config, 73, *76*, *80*, *104*, *107*
label_1d_graphics, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *55–58*, 58, *60*, *62*, *70*, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *98–101*
label_1d_grid, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–59*, 59, *62*, *70*, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *99–101*
label_1d_loon, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–60*, 60, *70*, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *99–101*
label_2d_graphics, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, 62, *64*, *66*, *85–87*, *89, 90*, *95–97*
label_2d_grid, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63*, 63, *66*, *85–87*, *89, 90*, *95–97*
label_2d_loon, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, 64, *85–87*, *89, 90*, *95–97*
layout_1d_graphics, 66
layout_1d_grid, 66
layout_1d_loon, 67
layout_2d_graphics, 68
layout_2d_grid, 68
layout_2d_loon, 69
lines_1d_graphics, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *55–60*, *62*, 70, *72, 73*, *81, 82*, *84*, *91, 92*, *94*, *98–101*
lines_1d_grid, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–60*, *62*, *70*, 71, *73*, *81, 82*, *84*, *91, 92*, *94*, *99–101*
lines_1d_loon, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–60*, *62*, *70*, *72*, 72, *81, 82*, *84*, *91, 92*, *94*, *99–101*
list, *20*, *23*, *36*, *38, 39*, *43, 44*, *103*, *109*
logical, *20*, *23*, *36*, *38*, *44*, *61*, *65*, *102*, *108*, *110, 111*

matrix, *20*, *23, 24*, *43, 44*, *53*, *107*, *109*
move, 74

n2dcols_aux, *24*, *54*, 75, *77*, *101*
na_omit_loon, *74*, 75, *80*, *104*, *107*
next_move_tidy, 76
NULL, *20*
num_cols, *24*, *54*, *75*, 77, *101*
numeric, *20*, *107*

olive, 78

plot, *42, 43*, *80*
plot_exists, 78
plot_indices, *21–23*, *36*, *38*, 79
plot_region, *74*, *76*, 80, *104*, *107*
points_1d_graphics, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *55–60*, *62*, *70*, *72, 73*, 81, *82*, *84*, *91, 92*, *94*, *98–101*
points_1d_grid, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–60*, *62*, *71–73*, *81*, 82, *84*, *91, 92*, *94*, *99–101*
points_1d_loon, *5, 6*, *8*, *17–19*, *26–28*, *35–37*, *50, 51*, *53*, *56–60*, *62*, *71–73*, *81, 82*, 83, *91, 92*, *94*, *99–101*
points_2d_graphics, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, *66*, 84, *86, 87*, *89, 90*, *95–97*
points_2d_grid, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, *66*, *85*, 85, *87*, *89, 90*, *95–97*
points_2d_loon, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, *66*, *85, 86*, 86, *89, 90*, *95–97*

qq_2d_graphics, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, *66*, *85–87*, 88, *90*, *95–97*

qq_2d_grid, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, *66*, *85–87*, *89*, 89, *95–97*

rect_1d_graphics, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *55–60*, *62*, *70–73*, *81, 82*, *84*, 90, *92*, *94*, *98–101*
rect_1d_grid, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–60*, *62*, *71–73*, *81, 82*, *84*, *91*, 91, *94*, *99–101*
rect_1d_loon, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–60*, *62*, *71–73*, *81, 82*, *84*, *91, 92*, 93, *99–101*
rect_2d_graphics, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, *66*, *85–87*, *89, 90*, 94, *96, 97*
rect_2d_grid, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, *66*, *85–87*, *89, 90*, *95*, 95, *97*
rect_2d_loon, *9–11*, *13, 14*, *16*, *29*, *31, 32*, *38*, *45*, *47, 48*, *63, 64*, *66*, *85–87*, *89, 90*, *95, 96*, 96
rug_1d_graphics, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *55–60*, *62*, *70–73*, *81, 82*, *84*, *91, 92*, *94*, 98, *100, 101*
rug_1d_grid, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–60*, *62*, *71–73*, *81, 82*, *84*, *91, 92*, *94*, *99*, 99, *101*
rug_1d_loon, *5, 6*, *8*, *17–19*, *26–28*, *36*, *50, 51*, *53*, *56–60*, *62*, *71–73*, *81, 82*, *84*, *91, 92*, *94*, *99, 100*, 100

turn_checker, *24*, *54*, *75*, *77*, 101

unfold, 102, *111*

vector, *20*, *24*, *39*, *107*, *109, 110*
viewport, *104*
vport, *74*, *76*, *80*, 104, *107*, *111*

wine, 105

zenarrow, *74*, *76*, *80*, *104*, 106
zenpath, *24*, *40*, *43, 44*, *53, 54*, 107, *111*
zenplot, *5–9*, *11, 12*, *14–17*, *19, 20*, *24–27*, *29, 30*, *32*, *35*, *37*, *40*, *43–47*, *50–52*, *54–58*, *60–62*, *64–72*, *79*, *81–85*, *87–89*, *91–95*, *97–100*, *103*, *108*, 109