Package 'desplot'

July 20, 2020

Title Plotting Field Plans for Agricultural Experiments

Version 1.7

Type Package

Description A function for plotting maps of agricultural field experiments that are laid out in grids.

Imports ggplot2, grid, lattice, reshape2,

Suggests agridat, knitr, rmarkdown, testthat

License GPL-3

LazyData yes

URL http://kwstat.github.io/desplot/

BugReports https://github.com/kwstat/desplot/issues

VignetteBuilder knitr

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

Author Kevin Wright [aut, cre] (<https://orcid.org/0000-0002-0617-8673>)

Maintainer Kevin Wright <kw.stat@gmail.com>

Repository CRAN

Date/Publication 2020-07-20 17:00:03 UTC

R topics documented:

desplot *Plot the layout/data of a field experiment.*

Description

Use this function to plot the layout of a rectangular lattice field experiment and also the observed data values.

Usage

```
desplot(
  data,
  form = formula(NULL ~ x + y),
  num = NULL,
  num.string = NULL,
  col = NULL,
  col.string = NULL,
  text = NULL,
  text.string = NULL,
  out1 = NULL,
  out1.string = NULL,
  out2 = NULL,
  out2.string = NULL,
  dq = NULL,
  dq.string = NULL,
  col.regions = RedGrayBlue,
  col.text = NULL,
  text.levels = NULL,
  out1.gpar = list(col = "black", lwd = 3),
  out2.gpar = list(col = "yellow", lwd = 1, lty = 1),
  at,
  midpoint = "median",
  ticks = FALSE,
  flip = FALSE,
  main = NULL,
  xlab,
  ylab,
  shorten = "abb",
  show.key = TRUE,
  key.cex,
  cex = 0.4,
  strip.cex = 0.75,
  subset = TRUE,
  gg = FALSE,
  ...
)
```

```
ggdesplot(
  data,
  form = formula(NULL ~ x + y),
  num = NULL,
  num.string = NULL,
  col = NULL,
  col.string = NULL,
  text = NULL,
  text.string = NULL,
  out1 = NULL,
  out1.string = NULL,
  out2 = NULL,
  out2.string = NULL,
  dq = NULL,
  dq.string = NULL,
  col.regions = RedGrayBlue,
  col.text = NULL,
  text.levels = NULL,
  out1.gpar = list(col = "black", lwd = 3),
  out2.gpar = list(col = "yellow", lwd = 1, lty = 1),
  at,
  midpoint = "median",
  ticks = FALSE,
  flip = FALSE,
  main = NULL,
  xlab,
  ylab,
  shorten = "abb",
  show.key = TRUE,
  key.cex,
  cex = 0.4,
  strip.cex = 0.75,
  subset = TRUE,
  gg = FALSE,
  ...
)
```

Arguments

`data`	A data frame.
`form`	A formula like `yield~x*y|location`. Note x,y are numeric.
`num`	Bare name (no quotes) of the column of the data to use as a factor for number-coding the text in each cell.
`num.string`	String name of the column of the data to use as a factor for number-coding the text in each cell.
`col`	Bare name (no quotes) of the column of the data to use for color-coding the text shown in each cell.

col.string	String name of the column of the data to use for color-coding the text shown in each cell.
text	Bare name (no quotes) of the column of the data to use for the actual text shown in each cell.
text.string	String name of the column of the data to use for the actual text shown in each cell.
out1	Bare name (no quotes) of the column of the data to use for first-level outlining around blocks of cells.
out1.string	String name of the column of the data to use for first-level outlining around blocks of cells.
out2	Bare name (no quotes) of the column of the data to use for second-level outlining around blocks of cells.
out2.string	String name of the column of the data to use for second-level outlining around blocks of cells.
dq	Bare name (no quotes) of the column of the data to use for indicating bad data quality with diagonal lines. This can either be a numeric vector or a factor/text. Cells with 1/"Q"/"Questionable" have one diagonal line. Cells with 2/"B"/"Bad","S","Suppressed" have crossed diagonal lines.
dq.string	String name of the column of the data to use for indicating bad data quality with diagonal lines.
col.regions	Colors for the fill color of cells.
col.text	Vector of colors for text strings.
text.levels	Character strings to use instead of default 'levels'.
out1.gpar	A list of graphics parameters for first-level outlining. Can either be an ordinary `list()` or a call to `gpar()` from the `grid` package.
out2.gpar	Graphics parameters for second-level of outlining.
at	Breakpoints for the color ribbon. Use this instead of 'zlim'. Note: using 'at' causes 'midpoint' to be set to NULL.
midpoint	Method to find midpoint of the color ribbon. One of 'midrange', 'median, or a numeric value.
ticks	If TRUE, show tick marks along the bottom and left sides.
flip	If TRUE, vertically flip the image.
main	Main title.
xlab	Label for x axis.
ylab	Label for y axis.
shorten	Method for shortening text in the key, either 'abb', 'sub', 'no', or FALSE.
show.key	If TRUE, show the key on the left side. (This is not the ribbon.)
key.cex	Left legend cex.
cex	Expansion factor for text/number in each cell.
strip.cex	Strip cex.
subset	An expression that evaluates to logical index vector for subsetting the data.
gg	If TRUE, desplot() switches to ggdesplot().
...	Other.

Details

To create the plot using lattice graphics: 1. `desplot(...)`.

To create the plot using ggplot2 graphics, use one of the following: 1. `ggdesplot(...)`. 2. `desplot(...,gg=TRUE)`. 3. `options(desplot.gg=TRUE); desplot(...)`. Method 3 is useful to modify all results from existing scripts.

The lattice version is complete, mature, and robust. The ggplot2 version is incomplete. The legend can only show colors, and some function arguments are ignored. In general, lattice graphics are about 4-5 times faster than ggplot2 graphics. Not all lattice parameters are passed down to `xyplot`, but it is possible to make almost any change to the plot by assigning the desplot object to a variable and then edit the object by hand or use `update` to modify the object. Then print it manually. See the first example below.

Use `col.regions` to specify fill colors. This can either be a vector of colors or a function that produces a vector of colors. If the response variable is a factor and `col.regions` is a *function*, it will be ignored and the cells are filled with default light-colored backgrounds and a key is placed on the left. If the response variable is *numeric*, the cells are colored according to `col.regions`, and a ribbon key is placed on the right.

Use `shorten='abb'` (this is default) to shorten the cell text using the `abbreviate` function. Use `shorten='sub'` to use a 3-character substring. Use `shorten='no'` or `shorten=FALSE` for no shortening.

Note that two sub-plots with identical levels of the split-plot factor can be adjacent to each other by virtue of appearing in different whole-plots. To correctly outline the split-plot factor, simply concatenate the whole-plot factor and sub-plot factor together.

To get a map of a field with a true aspect ratio (lattice version only), include 'aspect=ylen/xlen' in the call, where 'ylen' is the vertical length of the field and 'xlen' is the horizontal length of the field.

To call this function inside another function, you can hack like this: vr <- "yield"; vx <- "x"; vy <- "y"; eval(parse(text=paste("desplot(", vr, "~", vx, "*", vy, ", data=yates.oats)")))

Value

A lattice or ggplot2 object

Author(s)

Kevin Wright

References

K. Ryder (1981). Field plans: why the biometrician finds them useful. *Experimental Agriculture*, 17, 243–256.

Examples

```
if(require(agridat)){

# Show how to customize any feature.  Here: make the strips bigger.
data(besag.met)
d1 <- desplot(besag.met,
```

```
              yield ~ col*row|county,
              main="besag.met",
              out1=rep, out2=block, out2.gpar=list(col="white"), strip.cex=2)
d1 <- update(d1, par.settings = list(layout.heights=list(strip=2)))
print(d1)

# Show experiment layout
data(yates.oats)
desplot(yates.oats,
        yield ~ col+row,
        out1=block, out2=gen)

desplot(yates.oats,
        block ~ col+row,
        col=nitro, text=gen, cex=1, out1=block,
        out2=gen, out2.gpar=list(col = "gray50", lwd = 1, lty = 1))

}
```

`geom_tileborder` *Borders between tiles*

Description

'geom_tileborder' draws a border between tiles of different classes. The required aesthetics are 'aes(x,y,grp)', where 'grp' is the grouping classification that separates tiles.

Usage

```
geom_tileborder(
  mapping = NULL,
  data = NULL,
  geom = "segment",
  position = "identity",
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

	A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x,10)`).
`geom`	The geometric object to use display the data
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

Details

Note, we cannot use 'aes(group)' because it groups the interaction of ALL discrete variables including facets. Since we do not want to draw a border between facets, we had to define a new aesthetic. See: # http://ggplot2.tidyverse.org/reference/aes_group_order.html

Also, we do not want to split the data into separate groups for each level of 'grp', so we need to include 'aes(group=1)'.

Examples

```
dd <- data.frame(
  x=c(1,2,1,2,3,1,2,1,2,3),
  y=c(2,2,2,2,2,1,1,1,1,1),
  loc=factor(c(1,1,2,2,2,1,1,2,2,2)),
  rep=factor(c(2,2,1,2,3,1,1,1,2,3)))
library(ggplot2)
ggplot(dd, aes(x=x, y=y)) +
  facet_wrap( ~ loc) +
  geom_tile(aes(fill=rep)) +
  geom_tileborder(aes(group=1, grp=rep), lwd=1.5)
# Compare to lattice version of desplot
# desplot::desplot(rep ~ x*y|loc, data=dd, out1=rep)
```

`panel.outlinelevelplot` *Panel Function for desplot*

Description

This is a panel function for `desplot` which fills cells with a background color and adds outlines around blocks of cells.

Usage

```
panel.outlinelevelplot(
  x,
  y,
  z,
  subscripts,
  at,
  ...,
  alpha.regions = 1,
  out1f,
  out1g,
  out2f,
  out2g,
  dq
)
```

Arguments

`x`	Coordinates
`y`	Coordinates
`z`	Value for filling each cell.
`subscripts`	For compatability.
`at`	Breakpoints for the colors.
`...`	Other
`alpha.regions`	Transparency for fill colors. Not well tested.
`out1f`	Factor to use for outlining (level 1).
`out1g`	Factor to use for outlining (level 2).
`out2f`	Graphics parameters to use for outlining.
`out2g`	Graphics parameters to use for outlining.
`dq`	Indicator of which cells should be flagged for data quality.

Details

It does not add the text labels, numbers, or colors.

The rule for determining where to draw outlines is to compare the levels of the factor used for outlining. If bordering cells have different levels of the factor, then a border is drawn. 'NA' values are ignored (otherwise, too many lines would be drawn).

The code works, but is probably overkill and has not been streamlined.

References

None

RedGrayBlue *Function to create a Red-Gray-Blue palette*

Description

A function to create a Red-Gray-Blue palette.

Usage

```
RedGrayBlue(n)
```

Arguments

n	Number of colors to create

Details

Using gray instead of white allows missing values to appear as white (actually, transparent).

Value

A vector of n colors.

Author(s)

Kevin Wright

Examples

```
pie(rep(1,11), col=RedGrayBlue(11))
title("RedGrayBlue(11)")
```

Index