The ggplot Package

July 9, 2007

Type Package

Title An implementation of the Grammar of Graphics in R

Version 0.4.2

Date 2007-05-05

Author Hadley Wickham <h.wickham@gmail.com>

Maintainer Hadley Wickham <h.wickham@gmail.com>

Description An implementation of the grammar of graphics in R. It combines the advantages of both base and lattice graphics: conditioning and shared axes are handled automatically, and you can still build up a plot step by step from multiple data sources. It also implements a more sophisticated multidimensional conditioning system and a consistent interface to map data to aesthetic attributes. See http://had.co.nz/ggplot/ for more information, documentation and examples.

Suggests butler, mapproj, quantreg, MASS

Depends R (&gt;= 2.3), grid, reshape (&gt;= 0.7.4), RColorBrewer

LazyData yes

License GPL

R topics documented:

"add<-" .............................. 5
add_defaults .................................. 5
add_position .................................. 6
add_scale ..................................... 7
aesdefaults ................................... 7
alpha ........................................ 8
axis_vp_path ................................. 9
best_scale ................................... 9
blueyellow2 .................................. 10
boxplot.weighted.formula .................... 11
R topics documented:

breaks .......................... 11
brewer_palettes .................. 12
brewer_type ....................... 13
build_df .......................... 13
.build_options ...................... 14
chop ................................ 15
chop_auto .......................... 16
chop.breaks ......................... 17
defaultaesthetics .................. 17
defaultgrob ........................ 18
dhist .............................. 19
expand_range ....................... 19
facet ................................ 20
get.names ........................... 21
gg_add ............................. 21
gg2density ......................... 22
ggableline ......................... 23
ggarea ............................. 24
ggaxis .............................. 25
ggaxis_labels ...................... 26
ggaxis_line ......................... 27
ggaxis_ticks ......................... 27
ggaxis_vp ........................... 28
ggbar ............................... 28
ggboxplot ........................... 30
ggcontour ........................... 31
ggdensity ........................... 32
ggdist .............................. 33
ggerrorbar ......................... 34
ggfluctuation ....................... 35
gggroup ............................ 36
gghistogram ......................... 37
gghline ............................. 38
ggjitter ............................ 39
ggline .............................. 40
ggmissing ......................... 42
ggorder ............................ 43
ggpath ............................. 43
ggpcp .............................. 44
ggplot.default ....................... 45
ggplot_plot ........................ 48"$.ggplot" ......................... 49
ggpoint ............................. 50
ggpolygon ......................... 51
ggpretty ............................ 52
ggquantile ......................... 53
ggrect ............................. 54
ggribbon ............................ 55
R topics documented:

print.ggplot ......................................................... 92
print.manual ....................................................... 93
print.options ....................................................... 94
print.scale ......................................................... 94
ps_equal ............................................................ 95
ps_map .............................................................. 96
ps_categorical ...................................................... 96
ps_continuous ...................................................... 97
pstime ............................................................... 98
qplot .............................................................. 99
range.scale ......................................................... 100
rescale ............................................................ 101
resolution .......................................................... 102
sc ................................................................. 102
scale_apply_combine_map ................................. 103
scale_apply_to_data ........................................... 104
scale_categorical ............................................... 104
scale_cont_colour ............................................... 105
scale_continuous .................................................. 106
scale_map ........................................................ 107
scale_mapping ...................................................... 108
scale_new ........................................................ 109
scale_quality ...................................................... 109
scales ............................................................. 110
sccolour .......................................................... 111
scfillbrewer ........................................................ 112
scgradient ........................................................ 112
schcl ............................................................... 113
schsv ............................................................... 114
sclinetype .......................................................... 115
scmanual ........................................................... 116
scrgb .............................................................. 117
scshape ............................................................ 118
scsize ............................................................. 119
setdata ............................................................ 119
setfacets ........................................................... 120
setup_viewports ................................................... 121
summary.ggplot ................................................... 121
tailcolor .......................................................... 122
tr ................................................................. 123
tryapply ............................................................ 123
uneval .............................................................. 124
"update<-.scales" ............................................... 125
"update<-" ......................................................... 125
viewport_default .................................................. 126
vp_name ........................................................... 127
vp_path ........................................................... 127

Index 129
"add<-"  Add new scale

Description
Add new scale to list.

Usage
"add<-"(x, value)

Arguments
x
value

Details
Will overwrite any existing scales that use the same output variables.

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

add_defaults  Add default scales.

Description
Add default scales to a plot.

Usage
add_defaults(p = .PLOT, aesthetics)

Arguments
p plot object, if not specified will use current plot
aesthetics list of unevaluated aesthetics
Details
You shouldn’t need to call this function yourself. If you want to add a scale to a plot, use `add_scale`.

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples
### add_scale

**Add scale**

**Description**

Add (one) scale to the plot

**Usage**

```r
add_scale(p = .PLOT, scale)
```

**Arguments**

- `p`: plot object, if not specified will use current plot
- `scale`: scale to add, see `scales` for possible options

**Details**

You shouldn’t need to call this function yourself as all scale objects provide a convenient method to do so automatically. These are the functions that start with `sc`.

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

```r
p <- ggplot(movies, aesthetics=list(x=length, y=rating))
add_scale(ggpoint(p), position_continuous('x', range=c(80,100)))
aesdefaults(x, y, ...)
```

### aesdefaults

**Aesthetic defaults**

**Description**

Convenience method for setting aesthetic defaults

**Usage**

```r
aesdefaults(x, y, ...)
```

**Arguments**

- `x`: values from aesthetic mappings
- `y`: defaults
- `...`: user specified values
Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

---

```r
alpha  alpha
```

Description

Give a colour an alpha level

Usage

```r
alpha(colour, alpha)
```

Arguments

<table>
<thead>
<tr>
<th>colour</th>
<th>colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>alpha</td>
</tr>
</tbody>
</table>

alpha level [0,1]

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**axis_vp_path**

*Axis viewport path*

---

**Description**

Compute viewport path for specified component of axis

**Usage**

```
axis_vp_path(position, name)
```

**Arguments**

- **position**: position of axis
- **name**: component name

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

**best_scale**

*Best scale*

---

**Description**

Using `possible_scales` and `scale_quality` find

**Usage**

```
best_scale(data, expand=0)
```

**Arguments**

- **data**: vector of data
- **expand**: multiplicative expansion factor (experimental)
Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

---

blueyellow2

Create blue-yellow colour scheme

Description

Create a colour scheme that varies continuously along one ray of a colour space

Usage

blueyellow2(n)

Arguments

n

number of colours to produce

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**boxplot.weighted.formula**

*Weighted box plot calculations*

**Description**

**Usage**

```r
boxplot.weighted.formula(formula, data = NULL, ..., weights=1, subset, na.action = NULL)
```

**Arguments**

- `formula`
- `data`
- `...`
- `weights`
- `subset`
- `na.action`

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

**breaks**

*Scale breaks*

**Description**

Compute breaks of scale object

**Usage**

```r
breaks(scale, ...)
```
Arguments

scale scale object

Details

Author(s)

 Hadley Wickham <h.wickham@gmail.com>

Examples

---

**brewer_palettes** Get Brewer colour palettes

Description

Convenience function to retrieve private RColorBrewer palettes.

Usage

`brewer_palettes(type)`

Arguments

 type type of palettes to retrieve

Details

Author(s)

 Hadley Wickham <h.wickham@gmail.com>

Examples
**brewer_type**

Description

Return the type of factor in Cynthia brewers scheme.

Usage

```r
brewer_type(x)
```

Arguments

- `x`

Details

Value

character string giving Brewer type

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

**build_df**

Description

Build data frame for a plot with given data and ... (dots) arguments

Usage

```r
build_df(plot, data = plot$data, aesthetics=NULL)
```

Arguments

- `plot` plot object
- `data` data frame to use
- `aesthetics` extra arguments supplied by user that should be used first
Details

Depending on the arguments supplied to `plot_add` we need to stitch together a data frame using the defaults from `plot$defaults` where the user hasn’t explicitly specified otherwise.

Arguments in dots are evaluated in the context of `data` so that column names can easily be references.

Also makes sure that it contains all the columns required to correctly place the output into the row+column structure defined by the formula, by using `expand.grid.df` to add in extra columns if needed.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
.build_options

Set ggplot options
```

Description

Set global options for ggplot.

Usage

`.build_options(opt)`

Arguments

opt

list of options to get/set

Details

These are aliased into every plot object, so that `p$grid.col` will return the default grid colour, unless it has been overriden for a particular plot object. You can change the global options using the function, or the options for a specific plot by setting the values directly on the object. See the examples for more details.

Colour settings:

- axis.colour: axis text and line colour ("black")
- background.colour: background text colour ("black"), used for title
- background.fill: background fill ("white")
- grid.colour: plot grid colour ("white")
- grid.fill: plot grid background fill ("grey90")
Strip settings

- `strip.text`: function with two arguments (variable, and value) used for generating strip labels
- `strip.gp`: graphic parameter settings for the strip
- `strip.text.gp`: graphic parameter settings for the strip text

Legend settings

- `legend.position`: position of legend: "none" to hide legend; "left", "right", "top", "bottom", for positioning outside of plot; c(x, y) for positioning on top of plot

Other settings:

- `aspect.ratio`: aspect ratio of facets. Set to `NULL` to allow to vary with device size

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
ggopt(background.fill = "black", background.color ="white") # all new plots will use this
p <- ggplot(ggplot(tips, smoker ~ sex,aesthetics = list(y = tip, x = total_bill)))
p
p$background.fill = "white"
p
p$strip.text.gp <- gpar(col="red", fontsize=8)
p$background.colour <- "pink"
p$grid.colour <- "green"
p$grid.fill <- "blue"
p # a very ugly plot!
ggopt(background.fill = "white", background.color ="black")
```

Description

Chop a continuous variable into a categorical variable.

Usage

```r
chop(x, n=5, method="quantiles", midpoint=0, digits=2)
```
Arguments

- \( x \)
- \( n \)
- method
- midpoint
- digits

Details

Chop provides a convenient interface to the main methods of converting a continuous variable into a categorical variable.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

---

chop_auto

**Automatic chop**

Description

Keep categorical variables as is, chop up continuous variable

Usage

\[
\text{chop\_auto}(x)
\]

Arguments

- \( x \)

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
chop.breaks  Chop breaks

Description
Calculate breakpoints for chop function

Usage
chop.breaks(x, n, method, midpoint=NA)

Arguments
x
n
method
midpoint

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

defaultaesthetics  Default aesthetics

Description
Modify the default aesthetics for a plot

Usage
defaultaesthetics(plot, aesthetics)

Arguments
plot  plot object
aesthetics  new list of aesthetics
Details

Value

modified plot object

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

| defaultgrob | Default grob |

Description

Return the default grob to use for creating a legend

Usage

defaultgrob(x)

Arguments

x scale object

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**dhist**

**Description**

An algorithm for computing histogram breaks. Produces irregular bins.

**Usage**

\[ \text{dhist}(x, a=5 \times \text{diff} \left( \text{quantile}(x, c(0.25, 0.75)) \right), \text{nbins}=10, \text{rx} = \text{range}(x)) \]

**Arguments**

- `x`
- `a`
- `nbins`
- `rx`

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

**expand_range**

**Expand range**

**Description**

Convenience function for expanding a range with a multiplicative

**Usage**

\[ \text{expand_range}(\text{range}, \text{mul}=0, \text{add}=0) \]

**Arguments**

- `range` range of data
- `mul` multiplicative construct
- `add` additive constant
Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

---

`facet`  
*Break up data into individual facets*

Description

Usage

```
facet(x, formula, margins)
```

Arguments

- `x`
- `formula`
- `margins`

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples
**get.names**

*Get grid names*

**Description**

**Usage**

```r
get.names(x, indent=0)
```

**Arguments**

- `x`
- `indent`

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

**gg_add**

*gg add*

**Description**

Convenience method to make writing gg_XXXX functions easier.

**Usage**

```r
gg_add(map, plot, aesthetics=list(), ..., data=NULL)
```

**Arguments**

- `map` type of grob mapping to add
- `plot` plot object
- `aesthetics` list of aesthetic mappings
- `...` parameters passed to grob function
- `data` data frame
Details

Will automatically add scales if needed.

Value

modified plot object

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
library(ggplot2)

# Example usage

ggplot(mpg, aes(x = displacement, y = hwy)) +
  geom_point() +
  gg2density(aesthetics = list(x = 'displacement', y = 'hwy'))
```

Description

Perform a 2D kernel density estimation using `kde2d` and

Usage

```
gg2density(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

Arguments

- **plot**: the plot object to modify
- **aesthetics**: named list of aesthetic mappings, see details for more information
- **...**: other options, see details for more information
- **data**: data source, if not specified the plot default will be used

Details

This is another function useful for dealing with overplotting.

Aesthetic mappings that this grob function understands:

- `x`: x position (required)
- `y`: y position (required)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`

Other options:

- passed to `ggcontour`, see it for details
**ggabline**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**See Also**

`ggcontour` for another way of dealing with overplotting

**Examples**

```r
m <- ggplot(movies, aes(y=length, x=rating))
dens <- MASS::kde2d(movies$rating, movies$length)
densdf <- data.frame(expand.grid(rating = dens$x, length = dens$y), z=as.vector(dens$z))
ggcontour(m, list(z=z), data=densdf)
gg2density(m)
# they don't look the same due to scaling effects on kde2d
```

---

**ggabline**

*Grob function: abline*

**Description**

Add line specified by slope and intercept to a plot

**Usage**

```r
ggabline(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

**Arguments**

- `plot` the plot object to modify
- `aesthetics` named list of aesthetic mappings, see details for more information
- `...` other options, see details for more information
- `data` data source, if not specified the plot default will be used

**Details**

Aesthetic mappings that this grob function understands:

- none

Other options:

- `intercept`: intercept(s) of line
- `slope`: slope(s) of line, set to Inf
- `colour`: line colour
- `size`: line thickness
- `linetype`: line type
- `range`: x (or y if slope infinite) range to draw the line. This is sometimes necessary because ggplot isn’t smart enough to calculate the entire range of the data
Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
p <- ggplot(mtcars, aesthetics=list(x = wt, y=mpg))
ggabline(ggpoint(p), intercept=30, slope=-5)
ggabline(ggpoint(p), intercept=c(30,40,50), slope=-5)
ggsmooth(ggpoint(p), method=lm, formula=y~x)
```

---

ggarea

*Grob function: area*

Description

Add an filled area to a plot.

Usage

```r
ggarea(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

Arguments

- `plot`: x positions
- `aesthetics`: y positions
- `...`: id variable used to separate observations into different areas
- `data`: colour
- `pattern`: ...

Details

Aesthetic mappings that this grob function understands:

- `x`: x position (required)
- `y`: y position (required)
- `id`: identifier variable used to break up into multiple paths
- `colour`: line colour (see `sccolour`)
- `fill`: fill colour (see `sccolour`)
- `linetype`: line style/type (see `sclinetype`)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, e.g. `pscontinuous` or `pscategorical`.

Other options:

- none
Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

```
huron <- data.frame(year = 1875:1972, level = as.vector(LakeHuron))
p <- ggplot(huron, aes=list(y=level, x=year))
ggarea(p)
ggareap, colour="black")
gline(ggareap)) # better
qplot(year, level, data=huron, type=c("area", "line"))
ggarea(p, fill=alpha("grey80", 0.5))
pscontinuous(ggarea(p), "y", range=c(0,NA))
```

**ggaxis**  
*Grob axis*

Description
Grob for axes

Usage
```
ggaxis(at, labels, position="right", scale=c(0,1))
```

Arguments
- **at**  
  position of ticks
- **labels**  
  labels at ticks
- **position**  
  position of axis (top, bottom, left or right)
- **scale**  
  range of data values

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples
Description

Grob for axis labels

Usage

`ggaxis_labels(at, labels, position)`

Arguments

- `at` position of ticks
- `labels` labels at ticks
- `position` position of axis (top, bottom, left or right)

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

Description

Grob for axis baseline

Usage

`ggaxis_line(at, position)`

Arguments

- `at` position of ticks
- `position` position of axis (top, bottom, left or right)
ggaxis_ticks

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

---

ggaxis_ticks  

Grob axis ticks

Description

Grob for axis ticks

Usage

ggaxis_ticks(at, position)

Arguments

at  position of ticks
position  position of axis (top, bottom, left or right)

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
ggaxis_vp  

*Grob axis viewport*

**Description**

Generate viewport for axis grobs

**Usage**

```r
ggaxis_vp(position, labels, scale=c(0,1))
```

**Arguments**

- **position**  
  position of axis (top, bottom, left or right)

- **labels**  
  labels at ticks

- **scale**  
  range of data values

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

**gbar**  

*Grob function: bars*

**Description**

Add bars to a plot

**Usage**

```r
gbar(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

**Arguments**

- **plot**  
  the plot object to modify

- **aesthetics**  
  named list of aesthetic mappings, see details for more information

- **...**  
  other options, see details for more information

- **data**  
  data source, if not specified the plot default will be used
Details

The bar grob produces bars from the y-position to the y=0.

Aesthetic mappings that this grob function understands:

- x: x position (required)
- y: y position (required)
- fill: fill colour (see sccolour)

These can be specified in the plot defaults (see ggplot) or in the aesthetics argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with ps, e.g. pscontinuous or pscategorical

Other options:

- avoid: how should overplotting be dealt with? "none" (default) = do nothing, "stack" = stack bars on top of one another, "dodge" = dodge bars from side to side
- sort: Should the values of the bars be sorted

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

ggrect

Examples

cyltab <- as.data.frame(table(cyl=mtcars$cyl))
p <- ggplot(cyltab, aes=list(y=Freq, x=cyl))
ggbar(p)
ggbar(p, fill="white", colour="red")
# Can also make a stacked bar chart
p <- ggplot(mtcars, aes=list(y=1, x=factor(cyl)))
ggbar(p, avoid="stack")
ggbar(p, avoid="stack", colour="red") # Made up of multiple small bars
p <- ggplot(mtcars, aes=list(y=mpg, x=factor(cyl)))
ggbar(p, avoid="stack")
ggbar(p, avoid="dodge", sort=TRUE)
ggbar(p, aes=list(fill=mpg), avoid="dodge", sort=TRUE)
ggbar(p, avoid="stack", sort=TRUE)
ggboxplot

Grob function: boxplot

Description

Add box and whiskers

Usage

ggboxplot(plot = .PLOT, aesthetics=list(), ..., data=NULL)

Arguments

plot the plot object to modify
aesthetics named list of aesthetic mappings, see details for more information
... other options, see details for more information
data data source, if not specified the plot default will be used

Details

Aesthetic mappings that this grob function understands:

- **x**: x position (required)
- **y**: y position (required)
- **weight**: observation weights

These can be specified in the plot defaults (see ggplot) or in the aesthetics argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with ps, eg. pscontinuous or pscategorical

Other options:

- **breaks**: how to break up the x axis (only used if not already a factor)
- **orientation**: whether boxplots should be horizontal or vertical. If missing will automatically decide based on which variable is a factor.
- other arguments passed to boxplot

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

ggquantile for a continuous analogue of the boxplot
Examples

```r
p <- ggplot(mtcars, aesthetics=list(y=mpg, x=factor(cyl)))
p2 <- ggplot(mtcars, aesthetics=list(x=mpg, y=factor(cyl)))
ggpoint(p)
ggboxplot(p)
ggboxplot(p2)
ggboxplot(p, fill="pink", colour="green")
ggpoint(ggboxplot(p))
ggboxplot(p)
```

---

**ggcontour**

*Grob function: contours*

**Description**

Create a grob to display contours of a 3D data set.

**Usage**

```r
ggcontour(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

**Arguments**

- `plot` the plot object to modify
- `aesthetics` named list of aesthetic mappings, see details for more information
- `...` other options, see details for more information
- `data` data source, if not specified the plot default will be used

**Details**

Aesthetic mappings that this grob function understands:

- `x`: x position (required)
- `y`: y position (required)
- `z`: z position (required)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`

Other options:

- `nlevels`: number of contours to draw
- `levels`: contour positions
- `...`: other aesthetic parameters passed to `grob_path`
Description

Display a smooth density estimate.

Usage

```r
ggdensity(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

Arguments

- `plot`: the plot object to modify
- `aesthetics`: named list of aesthetic mappings, see details for more information
- `...`: other options, see details for more information
- `data`: data source, if not specified the plot default will be used

Details

Aesthetic mappings that this grob function understands:

- `x`: x position (required)
- `weight`: observation weights

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`

Other options:

- `adjust`: see `density` for details
- **kernel**: kernel used for density estimation, see `density` for details
- **other aesthetic properties passed on to `ggline`**
- **weight**: observation weights

**Author(s)**

Hadley Wickham &lt;h.wickham@gmail.com&gt;

**See Also**

`gghistogram, density`

**Examples**

```r
m <- ggplot(movies, aesthetics=list(x=rating))
ggdensity(m)
qplot(length, data=movies, type="density")
qplot(length, data=movies, type="density", weight=rating)
qplot(length, data=movies, type="density", weight=rating/sum(rating))
qplot(log(length), data=movies, type="density", log="x")
m <- ggplot(movies, Action ~ Comedy, aesthetics=list(x=rating), margins=TRUE)
ggdensity(m)
ggdensity(m, scale="freq")
ggdensity(m, colour="darkgreen", size=5)
```

---

**ggdist**

*Distribution plot*

**Description**

Experimental template

**Usage**

```r
ggdist(data, vars=names(data), facets = . ~ .)
```

**Arguments**

- data
- vars
- facets

**Details**
Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
```

Description

Add error bars to a plot

Usage

```r
ggerrorbar(plot = .PLOT, aesthetics = list(), ..., data = NULL)
```

Arguments

- `plot`: the plot object to modify
- `aesthetics`: named list of aesthetic mappings, see details for more information
- `...`: other options, see details for more information
- `data`: data source, if not specified the plot default will be used

Details

The error bar grob adds error bars to a plot. Thanks to Timm Danker for supplying some initial code and the motivation to include it in ggplot.

Aesthetic mappings that this grob function understands:

- `x`: x position (required)
- `y`: y position (required)
- `plus`: length of error bar in positive direction (required)
- `minus`: length of error bar in negative direction (defaults to -plus)
- `colour`: line colour (see `sccolour`)
- `size`: size of the line, in mm (see `scsise`)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`

Other options:

- `avoid`: how should overplotting be dealt with? "none" (default) = do nothing, "stack" = stack bars on top of one another, "dodge" = dodge bars from side to side
ggfluctuation

Author(s)
Hadley Wickham <h.wickham@gmail.com>

See Also
ggbar

Examples

df <- data.frame(x = factor(c(1, 1, 2, 2)), y = c(1, 5, 3, 4), g = c(1, 2, 1, 2), bar = c(0.1, 0.3, 0.3, 0.2))
df2 <- df[c(1, 3), ]; df2

p <- ggbar(ggplot(data=df, aes=list(fill=g, y=y, x=x)))
ggerrorbar(p, aes=list(plus=bar))
qplot(x, y, df, types=list("bar","errorbar"), avoid="dodge", aes=list(fill=g, plus=bar))
qplot(x, y, df, types=list("bar","errorbar"), avoid="dodge", aes=list(fill=g, plus=bar, minus=-2*bar))
qplot(x, y, df2, types=list("point","errorbar"), aes=list(plus=bar), width=0.1)
qplot(x, y, df2, types=list("bar","line","point","errorbar"), aes=list(fill=g, plus=bar, barcolour=g))
qplot(x, y, df2, types=list("jitter","errorbar"), aes=list(plus=bar))
qplot(x, y, df, types=list("point","line","errorbar"), aes=list(plus=bar, id=g), width=0.1)

---

ggfluctuation

Fluctuation plot

Description
Create a fluctuation plot.

Usage

ggfluctuation(table, type="size", floor=0, ceiling=max(table$freq, na.rm=TRUE))

Arguments

- **table**: a table of values, or a data frame with three columns, the last column being frequency
- **type**: size, or colour to create traditional heatmap
- **floor**: don’t display cells smaller than this value
- **ceiling**:

Details
A fluctuation diagram is a graphical representation of a contingency table. This function currently only supports 2D contingency tables but extension to more should be relatively straightforward.

With the default size fluctuation diagram, area is proportional to the count (length of sides proportional to sqrt(count))
Description

Create multiple of grobs based on id aesthetic.

Usage

`gggroup(plot = .PLOT, aesthetics=list(), ..., data=NULL)`

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot</td>
<td>the plot object to modify</td>
</tr>
<tr>
<td>aesthetics</td>
<td>named list of aesthetic mappings, see details for more information</td>
</tr>
<tr>
<td>...</td>
<td>other options, see details for more information</td>
</tr>
<tr>
<td>data</td>
<td>data source, if not specified the plot default will be used</td>
</tr>
</tbody>
</table>

Details

This grob function provides a general means of creating multiple grobs based on groups in the data. This is useful if you want to fit a separate smoother for each group in the data.

You will need an id variable in your aesthetics list with determines how the data is broken down.

Aesthetic mappings that this grob function understands:

- x:x position (required)
- y:y position (required)
- id:
- any other grobs used by the grob function you choose

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, e.g. `pscontinuous` or `pscategorical`

Other options:

- `grob`: grob function to use for subgroups
- anything else used by the grob function you choose
Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
p <- ggplot(mtcars, aesthetics=list(y=wt, x=qsec, id=cyl, colour=cyl))
gggroup(p)
gggroup(p, grob="density")
gggroup(p, grob="histogram", aes=list(fill=cyl))
gggroup(ggpoint(p), grob="smooth", se=FALSE, span=1)
gggroup(ggpoint(p), aes=list(id=cyl, size=cyl), grob="smooth", span=1)
```

gghistogram  

**Grob function: histogram**

Description

Draw a histogram

Usage

```r
gghistogram(plot = .PLOT, aesthetics=list(), scale="prob", ..., data=NULL)
```

Arguments

- `plot`: the plot object to modify
- `aesthetics`: named list of aesthetic mappings, see details for more information
- `scale`: other options, see details for more information
- `...`: data source, if not specified the plot default will be used

Details

Aesthetic mappings that this grob function understands:

Conceptually, the histogram is one of the most complicated of the grob functions, because it takes a 1D data set and makes it two dimensional. This necessitates an extra step, the `pre_histogram` function which bins the data and returns the bins with their counts. This data is then used by `grob_histogram` to plot the points.

- `x`: position (required)
- `weight`: observation weights

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`
gghline

Grob function: hline

Description

Add horizontal line(s) to a plot

Usage

gghline(plot = .PLOT, aesthetics=list(), ..., data=NULL)

Arguments

plot the plot object to modify
aesthetics named list of aesthetic mappings, see details for more information
... other options, see details for more information
data data source, if not specified the plot default will be used

Examples

m <- ggplot(movies, aesthetics=list(x=rating))
gghistogram(m)
gghistogram(m, breaks=100)
qplot(length, data=movies, type="histogram")
qplot(log(length), data=movies, type="histogram")
m <- ggplot(movies, Action ~ Comedy, aesthetics=list(x=rating), margins=TRUE)
gghistogram(m)
gghistogram(m, scale="freq")
gghistogram(m, colour="darkgreen", fill="white")
qplot(rating, data=movies, type="histogram")
qplot(rating, weight=votes, data=movies, type="histogram")
qplot(rating, weight=votes, data=movies, type=c("histogram", "density"))

Author(s)

Hadley Wickham <h.wickham@gmail.com>
ggjitter

Details
Aesthetic mappings that this grob function understands:

• none

Other options:

• position: vertical position(s) to draw lines
• colour: line colour
• size: line thickness
• linetype: line type
• range: x (or y if slope infinite) range to draw the line. This is sometimes necessary because ggplot isn’t smart enough to calculate the entire range of the data

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

p <- ggplot(mtcars, aesthetics=list(x = wt, y=mpg))
gghline(ggpoint(p), position=mean(mtcars$mpg), size=2)
Details

This is useful when plotting points with a categorical axis so to avoid overplotting.

Aesthetic mappings that this grob function understands:

- x: x position (required)
- y: y position (required)
- size: size of the point, in mm (see scsize)
- shape: shape of the glyph used to draw the point (see scshape)
- colour: point colour (see sccolour)

These can be specified in the plot defaults (see ggplot) or in the aesthetics argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with ps, eg. pscategorical or pscontinuous

Other options:

- xjitter: degree of jitter in x direction, see jitter for details, defaults to 1 if the x variable is a factor, 0 otherwise
- yjitter: degree of jitter in y direction, see jitter for details, defaults to 1 if the y variable is a factor, 0 otherwise

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
p <- ggplot(movies, aes=list(x=mpaa, y=rating))
ggjitter(p)
ggjitter(ggboxplot(p))
ggjitter(ggboxplot(p), xjitter=2)
ggjitter(ggboxplot(p), yjitter=1)
p <- ggplot(movies, aes=list(x=mpaa, y=factor(Action)))
ggjitter(p)
```

---

**ggline**  
**Grob function: line**

Description

Add a line to the plot

Usage

```r
ggline(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```
Arguments

plot the plot object to modify
aesthetics named list of aesthetic mappings, see details for more information
... other options, see details for more information
data data source, if not specified the plot default will be used

Details

Aesthetic mappings that this grob function understands:

- x:x position (required)
- y:y position (required)
- id:identifier variable used to break up into multiple paths
- size:size of the line, in mm (see scsize)
- colour:line colour (see sccolour)
- linetype:line style/type (see sclinetype)

These can be specified in the plot defaults (see ggplot) or in the aesthetics argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with ps, eg. pscontinuous or pscategorical

Other options:

- none

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

mry <- do.call(rbind, by(movies, round(movies$rating), function(df) {
  nums <- tapply(df$length, df$year, length)
  data.frame(rating=round(df$rating[1]), year = as.numeric(names(nums)), number=as.vector(nums))
})
p <- ggplot(mry, aesthetics = list(x=year, y=number, id=rating))
ggline(p)
ggpath(p, list(size=rating))
ggpath(p, list(colour=rating))
**ggmissing**

*Missing values plot*

**Description**

Create a plot to illustrate patterns of missing values

**Usage**

```r
ggmissing(data, avoid="stack", order=TRUE, missing.only = TRUE)
```

**Arguments**

- `data` - data.frame
- `avoid` - whether missings should be stacked or dodged, see `ggbbar` for more details
- `order` - whether variable should be ordered by number of missings
- `missing.only` - whether only variables containing some missing values should be shown

**Details**

The missing values plot is a useful tool to get a rapid overview of the number of missings in a dataset. It's strength is much more apparent when used with interactive graphics, as you can see in Mondrian ([http://rosuda.org/mondrian](http://rosuda.org/mondrian)) where this plot was copied from.

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**See Also**

`ggstructure, ggorder`

**Examples**

```r
mmissing <- movies
mmissing[sample(nrow(movies), 1000), sample(ncol(movies), 5)] <- NA
ggmissing(mmissing)
```

```r
ggmissing(mmissing, order=FALSE, missing.only = FALSE)
```

```r
pscontinuous(ggmissing(mmissing, avoid="dodge"), "y", transform=trans_sqrt, range=c(0, NA))
```

```r
pscontinuous(ggmissing(mmissing), "y", transform=trans_log10, range=c(1, NA))
```
ggorder

*Order plot*

Description

A plot to investigate the order in which observations were recorded.

Usage

```r
ggorder(data, scale="rank")
```

Arguments

- `data`: data set to plot
- `scale`: type of scaling to use. See `rescaler` for options

Details

ar Need ggobi version as well that creates edge between consecutive observations (and adds row number to dataset)

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

---

ggpath

*Grob function: path*

Description

Add a path (a line between points in the order that they appear in the dataset) to the plot

Usage

```r
ggpath(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

Arguments

- `plot`: the plot object to modify
- `aesthetics`: named list of aesthetic mappings, see details for more information
- `...`: other options, see details for more information
- `data`: data source, if not specified the plot default will be used
Details

Aesthetic mappings that this grob function understands:

- x: x position (required)
- y: y position (required)
- id: identifier variable used to break up into multiple paths
- size: size of the line, in mm (see scsize)
- colour: line colour (see sccolour)
- linetype: line style/type (see sclinetype)

These can be specified in the plot defaults (see ggplot) or in the aesthetics argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with ps, eg. pscontinuous or pscategorical

Other options:

- none

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
myear <- do.call(rbind, by(movies, movies$year, function(df) data.frame(year=df$year[1], mean.length = mean(df$length), mean.rating=mean(df$rating))))
p <- ggplot(myear, aesthetics=list(x=mean.length, y=mean.rating))
ggpath(p)
ggpath(p, list(size=year))
ggpath(p, list(colour=year))
ggpath(scsize(p, c(0.5,1)), list(size=year))
ggpath(scsize(p, c(0.5,1)), list(size=year))
p <- ggplot(mtcars, aesthetics=list(x=drat, y=wt))
ggpath(p)
ggpath(p, list(id=cyl))
```

---

**ggpcp**  
*Parallel coordinates plot.*

Description

Generate a plot “template” for a parallel coordinates plot.

Usage

```r
ggpcp(data, vars=names(data), scale="range", ...)
```
Arguments

- **data**: data frame
- **vars**: variables to include in parallel coordinates plot
- **scale**: scaling function, one of "range", "var" or "I"
- **...**: other arguments passed on plot creation

Details

One way to think about a parallel coordinates plot, is as plotting the data after it has transformation been transformed to gain a new variable. This function does this using melt.

This gives us enormous flexibility as we have separated out the type of drawing (lines by tradition) and can now use any of the existing grob functions. In particular this makes it very easy to create parallel boxplots, as shown in the example.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
# Using melt
ggline(ggpcp(mtcars))
ggline(ggpcp(mtcars, scale="var"))
ggline(ggpcp(mtcars, vars=names(mtcars)[3:6], formula= . ~cyl, scale="I"))
ggboxplot(ggpcp(mtcars, scale="I"))
ggline(ggpcp(mtcars, vars=names(mtcars[2:6])))
p <- ggpcp(mtcars, vars=names(mtcars[2:6]), formula= . ~vs)
ggline(p)
ggline(p, aes=list(colour=mpg))
```

---

**ggplot.default**  
Create a new plot

Description

Create a new ggplot plot

Usage

```r
ggplot.default(data = NULL, formula = . ~ ., margins=FALSE, aesthetics=list(), ...)
```
Arguments

- **data**: default data frame
- **formula**: formula describing row and column layout, see `reshape` for more details
- **margins**: a vector of names giving which margins to display, can include grand_row and grand_col or use TRUE to display all margins
- **aesthetics**: default list of aesthetic mappings (these can be colour, size, shape, line type – see individual grob functions for more details)

Details

This function creates the basic ggplot object which you can then furnish with graphical objects. Here you will set up the default data frame, default aesthetics and the formula that will determine how the panels are broken apart. See `reshape` for more details on specifying the facetting formula and margin arguments. Note that ggplot creates a plot object without a "plot": you need to grobs (points, lines, bars, etc.) to create something that you can see.

To get started, read the introductory vignette: `vignette("introduction", "ggplot")`

Steps to create a plot:

1. Create a new plot: `p <- ggplot(mtcars, aesthetics=list(y=hp, x=mpg))`
2. Set scales (if necessary)
3. Add grobs to the plot (ggpoint(p))

or, use `qplot`

Simple grobs:

- **ggabline**: line with given slope and intercept
- **ggarea**: area (polygons with base on y=0)
- **ggbar**: bars (stocked and dodged)
- **ggjitter**: jittered points (useful for discrete data)
- **ggline**: lines (paths sorted by x-axis values)
- **ggpath**: paths
- **ggpoint**: points
- **ggribbon**: ribbon
- **ggtext**: text
- **ggtile**: tiles, like a levelplot

Complex grobs:

- **ggboxplot**: box plot
- **ggcontour**: contour lines
- **ggdensity**: 1d density plot (continuous analogue of histogram)
- **gg2density**: 2d density countours
• **gghistogram**: histogram
  • **ggquantile**: quantile lines from a quantile regression
  • **ggsmooth**: smooths from any model family

Look at the documentation of these objects to see many examples of ggplot in action.

You will also want to add scales to the basic plot to give finer control over how the data values are mapped to aesthetics attributes of the grobs. For scales that control position of the points see:

  • **pscontinuous**: continuous scales (with optional transformation)
  • **pscategorical**: categorical scales

For other scales, see:

  • **sccolour**: colour categorical variables using Brewer colour scales (see also scfill)
  • **scgradient**: colour continuous scales with a gradient (see also scfillgradient)
  • **schcl**: map continuous variable to hue, chroma or luminance components (see also scfillhcl)
  • **schsv**: map continuous variable to hue, saturation or value components (see also scfillhsv)
  • **scmanual**: no automatic conversion, uses raw values directly
  • **sclinetype**: line type (solid, dashed, dotted, etc.)
  • **scrgb**: map continuous variable to red, green or blue components (see also scfillrgb)
  • **scshape**: point shape (glyph)
  • **scsize**: point or line size

ggplot is different from base and lattice graphics in how you build up the plot. With ggplot you build up the plot object (rather than the plot on the screen as in base graphics, or all at once as in lattice graphics.)

Each of the grob and scale functions adds the grob to the plot and returns the modified plot object. This lets you quickly experiment with different versions of the plot, using different grobs or scales. You can see how this works in the examples

You can also use **summary** to give a quick description of a plot.

If you want to change the background colour, how the panel strips are displayed, or any other default graphical option, see **ggopt**.

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**See Also**

[http://had.co.nz/ggplot], [stamp], [reshape], [ggopt], vignette("introduction", "ggplot")
Examples

```r
p <- ggplot(tips)
summary(p)
ggpoint(p, aesthetic=list(y = tip, x=total_bill))
p <- ggplot(tips, aesthetic=list(y = tip, x=total_bill))
p$title <- "Tips"
summary(p)
ggpoint(p)
ggpoint(p, colour="darkgreen", size=3)
ggpoint(p, list(colour=sex))
ggpoint(ggplot(tips, . ~ sex,aesthetics = list(y = tip, x = total_bill)))
p <- ggplot(tips, smoker ~ sex,aesthetics = list(y = tip, x = total_bill))
ggpoint(p)
ggsafe(ggpoint(p))
ggsafe(ggpoint (p), method=lm, formula=y~x)
ggline(ggpoint(p), slope=c(0.1,0.15,0.2))
(p2 <- ggplot(ggpoint(p, aes=list(colour=tip/total_bill)), slope=c(0.1,0.15,0.2)))
summary(p2)
sccgradient(p2)
sccgradient(p2, midpoint=0.15, high="green", mid="yellow")
```

```r
p<-ggplot(tips, sex ~ smoker, aesthetics=list(x=tip/total_bill), margins=TRUE)
gghistogram(p)
gghistogram(p, scale="density", breaks=seq(0,1, length=20))
ggdensity(p)
```

Description

Creates a complete `ggplot` grob.

Usage

```r
ggplot_plot(plot, viewport=viewport_default(plot, guides, plot$scales), panels=panels)
```

Arguments

- **plot**: plot object
- **viewport**: viewports
- **panels**: panels
- **guides**: guides
- **pretty**: should the plot be wrapped up inside the pretty accoutrements (labels, legends, etc)
Details

Delegates almost everything to its arguments. Responsible for the transformation chain and for collecting everything into one grob with the appropriate viewports.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

"$\cdot\text{ggplot}$"  
Access ggplot options

Description

Alias default options to plot object

Usage

"$\cdot\text{ggplot}(x, i)$"

Arguments

x

i

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
ggpoint

Grob function: point

Description
Add points to a plot

Usage

ggpoint(plot = .PLOT, aesthetics=list(), ..., data=NULL)

Arguments

- `plot` the plot object to modify
- `aesthetics` named list of aesthetic mappings, see details for more information
- `...` other options, see details for more information
- `data` data source, if not specified the plot default will be used

Details

Aesthetic mappings that this grob function understands:

- `x`: x position (required)
- `y`: y position (required)
- `size`: size of the point, in mm (see `scsize`)
- `shape`: shape of the glyph used to draw the point (see `scshape`)
- `colour`: point colour (see `sccolour`)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`

Other options:

- `unique`: if TRUE, draw at most one point at each unique location

Author(s)

Hadley Wickham <h.wickham@gmail.com>
**Examples**

```r
p <- ggplot(mtcars, aes(x=wt, y=mpg))
ggpoint(p)
ggpoint(p, list(colour=cyl))
ggpoint(p, list(blahbalh=cyl))  # unknown aesthetics are ignored
ggpoint(p, list(shape=cyl))
ggpoint(p, list(size=mpg))
ggpoint(p, list(size=mpg/wt))
ggpoint(p, list(x=cyl, colour=cyl))
p <- ggplot(mtcars)
ggpoint(p, aes(x=wt, y=mpg))
```

---

**ggpolygon**

*Grob function: polygon*

**Description**

Add polygons to a plot

**Usage**

```r
ggpolygon(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

**Arguments**

- `plot` the plot object to modify
- `aesthetics` named list of aesthetic mappings, see details for more information
- `...` other options, see details for more information
- `data` data source, if not specified the plot default will be used

**Details**

Aesthetic mappings that this grob function understands:

- `x`: x position (required)
- `y`: y position (required)
- `id`: identifier variable used to break up into multiple polygons
- `size`: size of the outline, in mm (see `scsize`)
- `colour`: outline colour (see `sccolour`)
- `fill`: internal colour (see `sccolour`)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, e.g. `pscontinuous` or `pscategorical`

Other options:

- `none`
Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
ggpretty
```

**Description**

Pretty axis breaks

**Usage**

```r
ggpretty(x)
```

**Arguments**

`x` values to prettify

**Details**

Same as `grid.pretty` but contains minimum and maximum of data as well. Useful for legends.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
ggquantile

Grob function: quantiles

Description

Add quantile lines from a quantile regression

Usage

ggquantile(plot = .PLOT, aesthetics=list(), ..., data=NULL)

Arguments

plot                 the plot object to modify
aesthetics           named list of aesthetic mappings, see details for more information
...                  other options, see details for more information
data                 data source, if not specified the plot default will be used

Details

This can be used a continuous analogue of a boxplot (see grob_boxplot) Lines will be automatically sized to reflect their distance from the median.

Aesthetic mappings that this grob function understands:

• x: x position (required)
• y: y position (required)
• weight: observation weights

These can be specified in the plot defaults (see ggplot) or in the aesthetics argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with ps, eg. pscontinuous or pscategorical

Other options:

• quantiles: quantiles to display
• formula: formula to use in quantile regression

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

rq for the code used to fit the quantile regression
Examples

```r
## Not run:
m <- ggplot(movies, aesthetics=list(y=length, x=rating))
ggquantile(m)
## End(Not run)
```

---

ggrect  

**Grob function: rectangle**

Description

Add rectangles to a plot

Usage

```r
ggrect(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

Arguments

- **plot**: the plot object to modify
- **aesthetics**: named list of aesthetic mappings, see details for more information
- **...**: other options, see details for more information
- **data**: data source, if not specified the plot default will be used

Details

This grob provides the basic functionality required by `ggbar` and `ggtile`. You should probably not call it yourself

Aesthetic mappings that this grob function understands:

- **x**: x position (required)
- **y**: y position (required)
- **width**: width of the rectangle (required)
- **height**: height of the rectangle (required)
- **fill**: fill colour (see `sccolour`)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, e.g. `pscontinuous` or `pscategorical`

Other options:

- **justification**: justification of the bar relative to its (x, y) location, see `rectGrob` for more details
**ggribbon**

**Author(s)**
Hadley Wickham <h.wickham@gmail.com>

**See Also**
ggbar, ggtile

**Examples**

```r
ggribbon
```

---

**Description**
Add a ribbon to the plot

**Usage**

```r
ggribbon(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

**Arguments**

- **plot**: the plot object to modify
- **aesthetics**: named list of aesthetic mappings, see details for more information
- **...**: other options, see details for more information
- **data**: data source, if not specified the plot default will be used

**Details**
Aesthetic mappings that this grob function understands:

- **x**: x position (required)
- **upper**: position of upper edge of ribbon (required)
- **lower**: position of lower edge of ribbon (required)
- **id**: identifier variable used to break up into multiple paths
- **colour**: line colour (see `sccolour`)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`

Other options:

- **none**
Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

```r
mry <- do.call(rbind, by(movies, round(movies$rating), function(df) {
  nums <- tapply(df$length, df$year, length)
  data.frame(rating=round(df$rating[1]), year = as.numeric(names(nums)), number=as.vector(nums))
})
p <- ggplot(mry, aesthetics = list(x=year, y=number, id=rating))
ggribbon(p, aes=list(upper=number+5, lower=number-5), fill="white", colour=NA)
ggribbon(p, aes=list(y=number, plus=5, minus=-5), fill="white", colour=NA)
ggribbon(p, aes=list(upper=number*1.1, lower=number*0.9), fill="white", colour=NA)
ggribbon(p, aes=list(upper=number+5, lower=number-5), fill="pink")
ggribbon(p, aes=list(upper=number+5, lower=number-5, fill=rating), colour=NA)
scfillgradient(ggribbon(p, aes=list(upper=number+5, lower=number-5, fill=rating), colour=NA), midpoint=5, low="red", high="darkgreen")
```

Description
Save a ggplot with sensible defaults

Usage

```r
ggsave(plot = .PLOT, filename=default_name(plot), device=default_device(filename),

Arguments

- plot: plot to save
- filename: file name/path of plot
- device: device to use, automatically extract from file name extension
- scale: scaling factor
- width: width (in inches)
- height: height (in inches)
- grid: grid to use, normal for white on pale grey, print for pale grey on white
- dpi: dpi to use for raster graphics

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>
Examples

Description
Add a smooth line to a plot

Usage

```r
ggsmooth(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

Arguments

- `plot` the plot object to modify
- `aesthetics` named list of aesthetic mappings, see details for more information
- `...` other options, see details for more information
- `data` data source, if not specified the plot default will be used

Details

This grob adds a smoother to the graphic to aid the eye in seeing important patterns, especially when there is a lot of overplotting.

You can customise this very freely, firstly by choosing the function used to fit the smoother (e.g. `loess, lm, rlm, gam, glm`) and the formula used to related the y and x values (e.g. `y ~ x, y ~ poly(x, 3)`).

This smoother is automatically restricted to the range of the data. If you want to perform predictions (or fit more complicated variabels with covariates) then you should fit the model and plot the predicted results.

Aesthetic mappings that this grob function understands:

- `x`: x position (required)
- `y`: y position (required)
- `size`: size of the point, in mm (see `scsize`)
- `colour`: point colour (see `sccolour`)
- `weight`: observation weights

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, e.g. `pscontinuous` or `pscategorical`

Other options:
• **method**: smoothing method (function) to use
• **formula**: formula to use in smoothing function
• **se**: display one standard error on either side of fit? (true by default)
• other arguments are passed to smoothing function

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

```r
p <- ggpoint(ggplot(mtcars, aesthetics=list(y=wt, x=qsec)))
gsmooth(p)
gsmooth(p, span=0.9)
gsmooth(p, method=lm)
gsmooth(p, method=lm, formula = y~splines::ns(x,3))
gsmooth(p, method=MASS::rlm, formula = y~splines::ns(x,3))
```

---

**ggstrip**

**Grob strip**

**Description**

Grob for strip labels

**Usage**

```r
ggstrip(text, horizontal=TRUE, strip.gp=ggopt()$strip.gp, text.gp=ggopt()$strip.text.gp)
```

**Arguments**

- **text**: text to display
- **horizontal**: orientation, horizontal or vertical
- **strip.gp**
- **text.gp**

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**
**ggstructure**  

*Structure plot*

**Description**

A plot which aims to reveal gross structural anomalies in the data

**Usage**

```
ggstructure(data, scale = "rank")
```

**Arguments**

- `data` data set to plot
- `scale` type of scaling to use. See `rescaler` for options

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

```
ggstructure(mtcars)
```

---

**ggtext**  

*Grob function: text*

**Description**

Add text to a plot

**Usage**

```
ggtext(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

**Arguments**

- `plot` the plot object to modify
- `aesthetics` named list of aesthetic mappings, see details for more information
- `...` other options, see details for more information
- `data` data source, if not specified the plot default will be used
Details

Aesthetic mappings that this grob function understands:

- **x**: x position (required)
- **y**: y position (required)
- **label**: text label to display
- **size**: size of the text, as a multiple of the default size, (see `scsize`)
- **rotation**: angle, in degrees, of text label
- **colour**: text colour (see `sccolour`)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`.

Other options:

- **justification**: justification of the text relative to its (x, y) location, see `textGrob` for more details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
p <- ggplot(mtcars, aesthetics=list(x=wt, y=mpg, labels = rownames(mtcars)))
ggtext(p)
ggtext(p, list(size=wt))
scsize(ggtext(p, list(size=wt)), c(0.5, 1.5))
ggtext(p, list(colour=cyl))
```

---

**ggtheme**

*Set ggplot theme.*

Description

A theme is a list of options for `ggopt`.

Usage

`ggtheme(theme)`

Arguments

- **theme**: theme, a list of options for `ggopt`
**ggtile**

**Details**
Use `ggtheme(defaulttheme)` to reset back to the default theme.

**Author(s)**
Hadley Wickham <h.wickham@gmail.com>

**Examples**

```r
ggtile

**Description**
Add tiles to a plot

**Usage**
ggtile(plot = .PLOT, aesthetics=list(), ..., data=NULL)

**Arguments**
- `plot` the plot object to modify
- `aesthetics` named list of aesthetic mappings, see details for more information
- `...` other options, see details for more information
- `data` data source, if not specified the plot default will be used

**Details**
The tile grob will tile the plot surface as densely as possible, assuming that every tile is the same size. It is similar to `levelplot` or `image`.

Aesthetic mappings that this grob function understands:

- `x` x position (required)
- `y` y position (required)
- `width` width of the rectangle
- `height` height of the rectangle
- `fill` fill colour (see `sccolour`)

These can be specified in the plot defaults (see `ggplot`) or in the `aesthetics` argument. If you want to modify the position of the points or any axis options, you will need to add a position scale to the plot. These functions start with `ps`, eg. `pscontinuous` or `pscategorical`

**Other options**
- none
Author(s)
Hadley Wickham <h.wickham@gmail.com>

See Also

ggrect, resolution

Examples

```r
pp <- function (n,r=4) {
  x <- seq(-r*pi, r*pi, len=n)
  df <- expand.grid(x=x, y=x)
  df$r <- sqrt(df$x^2 + df$y^2)
  df$z <- cos(df$r^2) * exp(-df$r/6)

  p <- ggplot(pp(20), aes=list(x=x,y=y))
  ggtile(p) # pretty useless!
  ggtile(p, list(fill=z))
  ggtile(p, list(height=abs(z), width=abs(z)))
  ggtile(ggplot(pp(100), aes=list(x=x,y=y,fill=z)))
  ggtile(ggplot(pp(100, r=2), aes=list(x=x,y=y,fill=z)))

  p <- ggplot(pp(20)[sample(20*20, size=200),], aes=list(x=x,y=y,fill=z))
  ggtile(p)
}
```

---

**ggvline**

**Grob function: vline**

Description

Add vertical line(s) to a plot

Usage

```r
ggvline(plot = .PLOT, aesthetics=list(), ..., data=NULL)
```

Arguments

- `plot`: the plot object to modify
- `aesthetics`: named list of aesthetic mappings, see details for more information
- `...`: other options, see details for more information
- `data`: data source, if not specified the plot default will be used
Details

Aesthetic mappings that this grob function understands:

- none

Other options:

- position: vertical position(s) to draw lines
- colour: line colour
- size: line thickness
- linetype: line type
- range: x (or y if slope infinite) range to draw the line. This is sometimes necessary because ggplot isn’t smart enough to calculate the entire range of the data

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
p <- ggplot(mtcars, aesthetics=list(x = wt, y=mpg))
ggvline(ggpoint(p), position=mean(mtcars$wt), size=2)
```

Description

Usage

grob_XXX()

Arguments

Details

See vignette("writing-grob-functions") for more details.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**grob_grid**

*Grob grid*

**Description**

Build up a subtle background grid

**Usage**

```r
grob_grid(aesthetics, xbreaks, ybreaks, fill=ggopt()$grid.fill, colour=ggopt()$grid.colour, ...)
```

**Arguments**

- **aesthetics**: not used
- **xbreaks**: x axis lines
- **ybreaks**: y axis lines
- **fill**: not used
- **colour**: ...

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

**grobPromise**

*Create a new grob promise object*

**Description**

**Usage**

```r
grobPromise(name, data, params, aesthetics)
```
**guides_basic**

**Arguments**
- name
- data
- params
- aesthetics

**Details**

**Author(s)**
Hadley Wickham <h.wickham@gmail.com>

**Examples**

```
# Default guides

# Generate default guides (legends, axes, and labels).

guides_basic(plot, scales=scales_default(plot))
```

**Description**
Generate default guides (legends, axes, and labels).

**Usage**

```
guides_basic(plot, scales=scales_default(plot))
```

**Arguments**
- plot: plot object
- scales: plot scales

**Details**
The default guides built for a plot are:
- the background colour over the whole plotting area (white)
- within each a panel a gray background with white gridlines (see `ggopt` to change)
- vertical and horizontal axes (appearance control by options to the position scales)
- facetting labels (see `ggopt` to change default colours etc)

To decouple plot construction from the objects that are placed within it, each of the grobs produced by this function uses a `vpPath`. 
Value

background list of grobs to appear in background
grid grobs that form background grob
axes_v vertical axes
axes_h horizontal axes
labels row and column labels

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

guides.default  Default guides

Description

Construct a default guide (legend) for a scale

Usage

guides.default(scale, ...)

Arguments

scale scale
...

Details

This is used for automatic legends.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
guides

**Description**
Create guides for the given scale

**Usage**
guides(scale, ...)

**Arguments**
scale
...

**Details**

**Author(s)**
Hadley Wickham <h.wickham@gmail.com>

**Examples**

guides.position

**Description**
Create x or y axis depending on variable name.

**Usage**
guides.position(scale, ...)

**Arguments**
scale
...

currently used
input

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

______________________________
input            Input
______________________________

Description

Return what input variable this scale uses

Usage

input(scale)

Arguments

scale

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**labels_default**  
*Default labels*

**Description**
Generate default facet labels.

**Usage**
```r
labels_default(plot, strip = plot$striplabel)
```

**Arguments**
- `plot`  
  plot object
- `strip`  

**Details**
Facet labels are only displayed when there are facets in a particular direction. By default the labels consist of the variable name : value. You can’t currently change this display, but it will be an option in the near future.

**Value**
gList containg text grobs with appropriate viewports

**Author(s)**
Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

**legends**  
*Legends*

**Description**
Create and arrange legends for all scales.

**Usage**
```r
legends(scales, horizontal = FALSE)
```
Arguments

- **scales**: scales object
- **horizontal**: direction of scales, vertical by default

Details

This function gathers together all of the legends produced by the scales that make up the plot and organises them into a `frameGrob`. If there are no legends to create, this function will return `NULL`.

Value

frameGrob, or NULL if no legends

Author(s)

- Hadley Wickham <h.wickham@gmail.com>

Examples

```r
make_all_grobs(x, data)
```

Description

Usage

```r
make_all_grobs(x, data)
```

Arguments

- **x**: 
- **data**: 

Details

Author(s)

- Hadley Wickham <h.wickham@gmail.com>

Examples
**make_grobs**

### Description

### Usage

```r
make_grobs(x, data)
```

### Arguments

- `x`
- `data`

### Details

### Author(s)

Hadley Wickham &lt;h.wickham@gmail.com&gt;

### Examples

**map_aesthetic**

### Description

Map raw data to aesthetics using the provided scale

### Usage

```r
map_aesthetic(scale, data, ...)
```

### Arguments

- `scale`
- `data`
- `...`
Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

map_aesthetic.scales

Description

Applies scales to data to return a data frame

Usage

map_aesthetic.scales(scale, data, ...)

Arguments

scale scale
data data
... other arguments (unused)

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**map_all**  

**Map all**

**Description**

Map all grobs with scale

**Usage**

```r
map_all(scale, matrix)
```

**Arguments**

- `scale`: scale to map with
- `matrix`: matrix of grobs

**Details**

**Author(s)**

 Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

**map_colour_brewer**  

*Aesthetic mapping: Brewer colours*

**Description**

Map categorical variables to Brewer colour scales

**Usage**

```r
map_colour_brewer(x, palette=1)
```

**Arguments**

- `x`: data vector
- `palette`: palette number to use
Details

If x is not a factor, will be converted to one by `chop_auto`. Can display at most 9 different categories.

Unordered factors will use qualitative scales. Ordered factors will use sequential scales. Ordered factors with negative level will use diverging scales.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
map_colour_gradient

Aesthetic mapping: colour gradient
```

Description

Map values to a colour gradient

Usage

```r
map_colour_gradient(x, low="red", mid="white", high="black", midpoint = 0, from=range(x, na.rm=TRUE))
```

Arguments

- `x`: data vector
- `low`: colour to use at bottom of scale
- `mid`: colour to use at middle of scale
- `high`: colour to use at top of scale
- `midpoint`: where mid point colour should be used
- `from`: where mid point colour should be used

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
Description

Map variables to hue, chroma or luminance.

Usage

map_colour_hcl(h=0, c=80, l=50, a=1, h.to=c(0,360), c.to=c(0,200), l.to=c(0,100), a.to=c(0,1), h.from = range(h, na.rm=TRUE), c.from = range(c, na.rm=TRUE), l.from = range(l, na.rm=TRUE), a.from = range(a, na.rm=TRUE))

Arguments

- h: hue
- c: chroma
- l: luminance
- a: alpha
- h.to: hue to
- c.to: chroma to
- l.to: luminance to
- a.to: alpha to
- h.from: hue from
- c.from: chroma from
- l.from: luminance from
- a.from: alpha from

Details

Using hue is the best.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
map_colour_hsv

Description

Map variables to hue, saturation or value

Usage

map_colour_hsv(h=1, s=1, v=1, a=1, h.to=c(0,1), s.to=c(0,1), v.to=c(0,1), a.to=c(0,1))

Arguments

h    hue
s    saturation
v    value
a    alpha
h.to  hue range
s.to  saturation range
v.to  value range
a.to  alpha range
h.from
s.from
v.from
a.from

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
Description

Convenience function to power map_colour_hsv.

Usage

.map_colour(colours, tos, froms, colour_function)

Arguments

colours       list of colour vectors
    tos       list of colour tos in same order as colours
    froms     function to produce colours in #rrggbbaa form
    colour_function

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

map_colour_rgb Aesthetic mapping: rgb components of colour

Description

Map variables to red, green or blue components.

Usage

map_colour_rgb(r=0, g=0, b=0, a=1, r.to = c(0,1), g.to=c(0,1), b.to=c(0,1), a.to=c(0,1))
Arguments

r  red

Arguments:
g  green

Arguments:
b  blue

Arguments:
a  alpha

Arguments:
r.to  red to

Arguments:
g.to  green to

Arguments:
b.to  blue to

Arguments:
a.to  alpha to

Arguments:
r.from

Arguments:
g.from

Arguments:
b.from

Arguments:
a.from

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

map_linetype

Aesthetic mapping: line type

Description

Map values to line types

Usage

map_linetype(x)

Arguments

x  data vector

Details

If x is not a factor, will be converted to one by link(chop_auto). Can display at most 4 different categories.
map_shape

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

map_shape  Aesthetic mapping: glyph shape

Description
Map values to point shapes.

Usage
map_shape(x, solid=FALSE)

Arguments
x data vector
solid use solid points?

Details
If x is not a factor, will be converted to one by chop_auto. Can display at most 6 different categories.

Author(s)
Hadley Wickham <h.wickham@gmail.com>

See Also
http://www.public.iastate.edu/~dicook/scgn/v141.pdf

Examples
match.fun.null  Cleaner version of match.fun

Description

Version of match.fun that returns NULL on failure

Usage

match.fun.null(x)

Arguments

x    function name to find (character vector)

Details

Value

function if found, otherwise NULL

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples


mdf  Mosaic plot

Description

Usage

mdf(formula=~., df, direction, offset, xrange=c(0,1), yrange=c(0,1))
mosaicdata

Arguments

formula
df
direction
offset
xrange
yrange

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

library(grid)
tt <- as.data.frame(Titanic)
mdf(~ Class, tt)  # counts
mdf(Freq ~ Class, tt)  # with a weighting variable
mdf(Freq ~ Class + Sex + Age, tt)
mdf(Freq ~ Class + Sex + Age, tt, direction=c("v","h","v"))

mosaicdata

Mosaic data

Description

Construct a data frame for producing a mosaic plot

Usage

mosaicdata(df, direction, offset, xrange=c(0,1), yrange=c(0,1))

Arguments

df data frame, with values in last column
direction character vector of direction ("v" or "h") to split in
offset vector of offsets to use
xrange xrange
yrange yrange
Details

Take a data frame, with last column value and then recursively create a data frame that reflects the position of each cell.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

IMDB movies data Movie information and user ratings from IMDB.com

Description

The internet movie database, imdb.com, is a website devoted to collecting movie data supplied by studios and fans. It claims to be the biggest movie database on the web and is run by amazon. More about information imdb.com can be found online, including information about the data collection process.

IMDB makes their raw data available. Unfortunately, the data is divided into many text files and the format of each file differs slightly. To create one data file containing all the desired information, I wrote a script in the ruby to extract the relevant information and store it in a database. This data was then exported into csv for easy import into many programs.

The following text files were downloaded and used:

- business.list. Total budget
- genres.list. Genres that a movie belongs to (eg. comedy and action)
- movies.list. Master list of all movie titles with year of production.
- mpaa-ratings-reasons.list. MPAA ratings.
- ratings.list. IMDB fan ratings.
- running-times.list. Movie length in minutes.

Movies were selected for inclusion if they had a known length and had been rated by at least one imdb user. The csv file contains the following fields:

- title. Title of the movie.
- year. Year of release.
- budget. Total budget (if known) in US dollars
- length. Length in minutes.
- rating. Average IMDB user rating.
- votes. Number of IMDB users who rated this movie.
• r1-10. Multiplying by ten gives percentile (to nearest 10%) of users who rated this movie a 1.
• mpaa. MPAA rating.
• action, animation, comedy, drama, documentary, romance, short. Binary variables representing if movie was classified as belonging to that genre.

Usage

data(movies)

Format

A data frame with 28819 rows and 24 variables

References

http://had.co.nz/data/movies/

Description

Return what output variables this scale produces

Usage

output(scale)

Arguments

scale

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
panels_default  Default panels function.

Description
Place all grobs in the [x,y] position in panel_x_y viewport

Usage
panels_default(plot, grobs)

Arguments
plot plot object
grobs

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

plot_add_grobs  Add grobs to plot

Description
Add grobs to plot grob list

Usage
plot_add_grobs(plot, grob_matrix)

Arguments
plot plot object
grob_matrix matrix of grobs
plot_add

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

plot_add

Plot add.

Description

Add graphical objects using specific mapping.

Usage

plot_add(plot=.PLOT, data=NULL, map="point", aesthetics=list(), ...)

Arguments

plot: plot object
data: data to use
map: how to map data into graphics object
aesthetics: arguments passed down to mapping object specifying (eg.) aesthetics to use
...

Details

This is the powerhouse function that you use to actually display stuff on your plot. You should really keep track of the new plot object that is created by this function, but if you’re lazy and don’t want to, it automatically stores the result in the "global" variable .PLOT.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
plot_grob_matrix  

**Plot grob matrix**

Description

Take a matrix of grobs and edit them so that their viewport name

Usage

```
plot_grob_matrix(gm, type=deparse(substitute(gm)))
```

Arguments

- `gm`: matrix of grobs to position
- `type`: viewport type to position them in

Details

This provides a convenient way of converting a matrix of grobs (as produced by `stamp`) into the equivalent visual representation. Assumes that there are viewports named `type_1_1`, `type_1_2`, ..., `type_nrow_ncol`.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

---

plot_layout  

**Plot layout**

Description

Create `grid.layout` for plot

Usage

```
plot_layout(gm, rows, cols, row.labels, col.labels, axes_h, axes_v, aspect_ratio)
```
Arguments

- **gm**: grob matrix
- **rows**: total number of rows
- **cols**: total number of columns
- **row.labels**: data frame of row labels
- **col.labels**: data frame of column labels
- **axes_h**: matrix of horizontal axis grobs
- **axes_v**: matrix of vertical axis grobs
- **aspect_ratio**: aspect ratio of cells (defaults to not preserved)

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples


<table>
<thead>
<tr>
<th>position_adjust</th>
<th>Position adjust</th>
</tr>
</thead>
</table>

Description

This function extracts out some of the code necessary to perform

Usage

```
position_adjust(aesthetics, avoid, direction, adjust=1)
```

Arguments

- **aesthetics**: list of aesthetics
- **avoid**: avoid method to use
- **direction**: direction (currently only works with vertical)
- **adjust**: adjustment factor if multiple rows correspond to same grob (awful hack)

Details
position_apply

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

Description
Apply a function to x and y position scales.

Usage

position_apply(scales, f, ...)

Arguments

scales scales
f function to apply
... other arguments to pass to f

Details
This is a convenience method because position scales can be made up of two separate scales, or one
scale that provides both x and y position mappings.

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples
possible_scales

**Description**

For a given data set, create all possible scales from the

**Usage**

```
possible_scales(data, expand=0)
```

**Arguments**

- `data` vector of data
- `expand` multiplicative expansion factor (experimental)

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

preprocess

**Description**

Precprocess panel given data set using information in grobPromise

**Usage**

```
preprocess(gp, data)
```

**Arguments**

- `gp` grobPromise to preprocess
- `data`

**Details**
Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

---

```r
preprocess_all
```

Preprocess all panels in grob promise

Description

Usage

```r
preprocess_all(gp, plot)
```

Arguments

- `gp`
- `plot`

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
Description

Build a plot with all the usual bits and pieces.

Usage

```r
prettyplot(plot, plotgrob)
```

Arguments

- `plot` : plot
- `plotgrob` : plot grob

Details

As well as the plotting area, a plot need:

- main title
- x and y axis labels
- space for legends (currently on the right hand side)
  These are stored as options in the plot object.

This function sets up the appropriate viewports and packs the various components in. The viewport is set up so that each component will only take up the amount of space that it requires.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
print.categorical  *Print categorical details*

**Description**

Print moderately useful details of this categorical scale.

**Usage**

```r
print.categorical(x, ...)```

**Arguments**

- `x`  
  scale object

- `...`  
  not used

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

print.ggplot  *Print ggplot*

**Description**

Print generic for ggplot. Plot on current graphics device.

**Usage**

```r
print.ggplot(x, newpage = is.null(vp), vp = NULL, save=ggopt()$save, ...)```

**Arguments**

- `x`  
  plot to display

- `newpage`  
  draw new (empty) page first?

- `vp`  
  viewport to draw plot in

- `save`  
  other arguments passed on to `ggplot_plot`

- `...`  
  not used
print.manual

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

print.manual

Print manual details

Description

Print moderately useful details of this manual scale.

Usage

print.manual(x, ...)

Arguments

x scale object

... not used

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
print.options  

Print options

Description

Usage

print.options(x, ...)

Arguments

x

...

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

print.scale  

Print scale

Description

Print a moderately useful description of a scale object

Usage

print.scale(x, ...)

Arguments

x  scale object

...  unused

Details
ps_equal

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

---

ps_equal  Equal scales

Description
Create a scale for axes with equal length on each

Usage

ps_equal()

Arguments

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples
ps_map  

*Map projection scale*

**Description**

Map projections

**Usage**

```r
ps_map(projection="mercator", params=NULL)
```

**Arguments**

- `projection`: projection to use, see `mapproject` for possible values
- `params`: list of parameters passed to `mapproject`

**Details**

This allows you to use map type projection.

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

pscategorical  

*Position: categorical*

**Description**

Add a categorical position scale to the plot

**Usage**

```r
pscategorical(plot = .PLOT, variable="x", name="", expand=c(0.01, 0.6))
```

**Arguments**

- `plot`: ggplot object
- `variable`: axis ("x" or "y")
- `name`: name of the scale (used in the legend)
- `expand`: expansion vector (numeric vector, multiplicative and additive expansion). Defaults to adding 0.6 on either end of the scale.
Details

A categorical scale converts a factor into a numerical representation very simply: by using `as.numeric`. This means that levels will be placed a integer locations in the same order that they appear in the levels of the factor (see `levels`).

If you want to reorder (or combine) categories, currently the best way to do this is to modify the original factors. In a future version of ggplot I will probably expand the categorical scale so that you can do that here.

This scale is added to the plot automatically when you use a categorical variable in the x or y aesthetics. You shouldn’t need to to call this function unless (for some reason) you want to change the expansion factor.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
p <- ggpoint(ggplot(mtcars, aesthetics=list(x=cyl, y=mpg)))
pscategorical(p, "x") # no change, because already categorical
pscategorical(p, "y") # chops into discrete segments
```

---

**pscontinuous**

*Position: continuous*

Description

Add a continuous position scale to the plot

Usage

```r
pscontinuous(plot = .PLOT, variable="x", name="", transform=trans_none, range=c(NA, NA), expand=c(0.05, 0), breaks=NULL)
```

Arguments

- `plot` : plot
- `variable` : variable ("x" or "y")
- `name` : name of the scale (used in the legend)
- `transform` : transform function and it’s inverse in a vector
- `range` : range, or leave missing to automatically determine
- `expand` : expansion vector (numeric vector, multiplicative and additive expansion)
- `breaks` : set breaks manually
Details

There are a few useful things that you can do with `pscontinuous`:

- set plot limits explicitly (with `range`)
- transform the scale (with `transform`)
- explicitly set where the axis labels (and grid lines) should appear (with `breaks`)

Note, that if you explicitly set the axis range, you may want to use `expand_range` to add a little extra room on each side.

When transforming an axes, you need to supply the transforming function and it's inverse (used to create pretty axis labels). I have created a few common ones for you:

- `trans_log10`: log base 10
- `trans_log2`: log base 2
- `trans_inverse`: inverse
- `trans_sqrt`: square root

Value

modified plot object

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
p <- ggpoint(ggplot(mtcars, aesthetics=list(x=mpg, y=disp)))
pscontinuous(p, "x", range=c(20,30))
pscontinuous(p, "y", breaks=seq(100, 400, 50))
pscontinuous(p, "y", transform=trans_inverse)
pscontinuous(p, "x", transform=trans_sqrt)
pscontinuous(p, "x", transform=trans_log10)
pscontinuous(p, "x", transform=trans_log10, breaks=seq(10,30, 5))
```

---

**pstime**

*Time scale takes standard numeric scale (in seconds since epoch)*

Description

and displays nice time intervals instead

Usage

```r
pstime(plot = .PLOT, variable="x", name="", transform=trans_none, range=c(NA,NA), e
```
qplot

Arguments

- plot
- variable
- name
- transform
- range
- expand
- by

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
qplot
```

Quick plot.

Description

Quick plot is a convenient wrapper function for creating simple ggplot plot objects.

Usage

```r
qplot(x, y = NULL, data, facets = . ~ ., margins=FALSE, types = "point", colour = NULL, size = NULL, shape = NULL, ... ylim = c(NA, NA), log = "", main = NULL, xlab = deparse(substitute(x)), ylab = deparse(substitute(y)), add=NULL, ...)
```

Arguments

- `x` x values
- `y` y values
- `data` data frame to use (optional)
- `facets` facetting formula to use
- `margins` grob type(s) to draw (can be a vector of multiple names)
- `types` vector to use for colours
- `colour` vector to use for sizes
- `size` vector to use for shapes
- `shape` vector to use for line type
linetype vector to use for fill colour
fill vector to use for ids
id vector to use for weights
weight limits for x axis (defaults to range of data)
xlim limits for y axis (defaults to range of data)
ylim which variables to log transform ("x", "y", or "xy")
log character vector or expression for plot title
main character vector or expression for x axis label
xlab character vector or expression for y axis label
ylab if specified, build on top of this gplot, rather than creating a new one
add other arguments passed on to the grob functions
...

Details

FIXME: describe how to get more information
FIXME: add more examples

qplot provides a quick way to create simple plots.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

qplot(LETTERS[1:5], 1:5, type="rect", main="Blah", xlab="Hi")
qplot(LETTERS[1:5], 1:5, type=c("tile", "point"), main="Blah", xlab="Hi", ylim=c(0,10), col=
qplot(wt, mpg, data=mtcars, col=cyl, shape=cyl, size=wt)
Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
rescale(x, to=c(0,1), from=range(x, na.rm=TRUE))
```

Description

Rescale numeric vector to have specified minimum and maximum.

Usage

```
rescale(x, to=c(0,1), from=range(x, na.rm=TRUE))
```

Arguments

- `x`
- `to`
- `from`

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
resolution  Resolution

Description
Compute the "resolution" of a data vector, ie. what is the smallest non-zero

Usage
resolution(x)

Arguments
x    numeric vector

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

See Also
ggtitle

Examples

---

sc  Scale methods

Description

Usage
sc()

Arguments
scale_apply_combine_map

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

```r
scale_apply_combine_map
    Scale apply combine map
```

Description

Usage

```r
scale_apply_combine_map(scale, rdmatrix)
```

Arguments

- `scale`
- `rdmatrix`

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples

```r
scale_new(c(size="3 * height"), "size", range=c(0, 5), class="size")
scale_new(c(x="weight", y="height"), c("x", "y"), class=c("equal", "position"))
scale_new(c(group="age"), c("colour", "glyph"), class="group")
scale_new(c(h="age"), c("colour"), class="hsv", range=list(l=c(4,4), h=c(30,40)))
scale_new(c())
```
scale_apply_to_data

Given a scale and a data.frame, this runs the expressions

Description

in the input and returns a list

Usage

scale_apply_to_data(scale, data=data.frame())

Arguments

scale
data

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

scale_categorical  Scale: general categorical

Description

Create a categorical scale for the specified variable

Usage

scale_categorical(variable="x", name="", expand=c(0,0), transform="as.numeric", ...)

Arguments

variable  variable that this scale is for
name  name of the scale (used in the legend)
expand  expansion factor for scale
transform  transformation function
...
**Details**

A categorical scale is a simple mapping from the levels of the categorical factor to values of the aesthetic attribute. These mappings are created by the aesthetic mapping functions `map_colour` and `map_linetype`. You will want to refer to those to see the possible options that can be used to control the mapping.

You should not call this function yourself. Instead use:

- `pscategorical`
- `sccolour`
- `sclinetype`
- `scshape`

If you use a continuous variable with this scale, it will automatically be converted to a categorical variable using `chop_auto`. If you want more control over the conversion you will want to use `chop` yourself. However, be careful to do all the chopping in one place, otherwise you may end up with different scales in different grobs.

This categorical scale places evenly spaces the levels of the factor along the intergers. If you want to change the order of the levels you will need to change the levels in the original factor.

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

```r
scale_cont_colour(inputs, name='', variable="colour", to=list())
```

### Description

Scale colour along a continuous path

### Usage

```r
scale_cont_colour(inputs, name='', variable="colour", to=list())
```

### Arguments

- `inputs`
- `name`
- `variable`
- `to`
Details

This scale class is the workhorse behind:

- `scrgb`
- `schsv`
- `schcl`

See those function for more details.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
scale_continuous
```

Scale: general continuous (incl. transformations)

Description

Transform scale with a monotone function

Usage

```r
scale_continuous(variable="x", name="", transform=trans_none, range=c(NA,NA), expand=c(0,0), breaks=NULL, to=NULL, ...)
```

Arguments

- `variable`: variable name
- `name`: name of the scale (used in the legend)
- `transform`: vector of length two, first element the transforming function and the second its inverse
- `range`: range of values to display on guides
- `expand`: expansion factor for guides
- `breaks`: manually specified breaks to use
- `to`: if non-null, scale variable to this range after transformation
- `...`:
Details

You shouldn’t call this function yourself. Please use one of:

scsize
scgradient
pscontinuous

The continuous scale is the most complicated of the scale functions as it accepts so many options. Making pretty axis labels for transformed data isn’t a trivial problem. Here I use a simple heuristic, and compute pretty breaks on the transformed range and then back transform to the original scale.

There are a few common transformation vectors defined: trans_log10, trans_log2, trans_sqrt, trans_inverse

Value

modified plot object

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
scale_map(scale, data, domain, range=scale$range)
```

Description

@arguments data.frame to be mapped

Usage

scale_map(scale, data, domain, range=scale$range)

Arguments

scale scale object
data data.frame to be mapped
domain named list of domains (one for each) input variable, each domain being a list of length two, the first element containing the range of the numeric component, the second the text labels of the categorical component
range
scale_mapping

Description
Text string describing how the mapping of the scale works

Usage
scale_mapping(x)

Arguments
x scale object

Details

Author(s)
Hadley Wickham <h.wickham@gmail.com>

Examples
Description

Usage

scale_new(input, output=names(input), name = paste(input, collapse="/"), ..., class=)

Arguments

input
output
name
...
class

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

description

scale_quality    Scale quality

Description

Compute the how good a scale is for a given data set

Usage

scale_quality(s, q, data)
Arguments

- **s**: scale values
- **q**: unit of Q used to create scale
- **data**: vector of data

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
scales
```

Scaleslist object

Description

Create a list of scales objects

Usage

```r
scales(...)
```

Arguments

- **...**: scales objects

Details

The scales output maintains a list of scale objects.

- input and output variables
- maps a data frames using those scales
- generates ready to use scales

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**sccolour**  
*Scale: categorical colour*

**Description**

Create a scale for categorical colours.

**Usage**

```
sccolour(plot = .PLOT, name="", h=c(0,360), l=65, c=100, alpha=1)
```

**Arguments**

- `plot`  plot to add scale to
- `name` name of the scale (used in the legend)
- `h` range of hues to use
- `l` luminance value
- `c` chroma value
- `alpha` alpha value

**Details**

Continuous variables will automatically be converted to categorical using `chop_auto`. You may want to use `chop` to convert the values yourself for finer control.

This scale is automatically added when you have colour in your list of aesthetics. For finer control, you will need to set the scale yourself. See the example for some ideas.

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**See Also**

`scale_categorical`, `map_colour`

**Examples**

```
p <- ggplot(movies, aes=list(x=mpaa, y=rating))
ggjitter(p, list(colour=rating))
ggjitter(p, list(colour=length))
ggjitter(p, list(colour=chop(length)))
ggjitter(p, list(colour=chop(length,3)))
sccolour(ggjitter(p, list(colour=chop(length,3))), 2)
```
**scfillbrewer**  
*Scale: Brewer colours*

**Description**

Use Brewer colour scheme for colour fill.

**Usage**

```r
scfillbrewer(plot = .PLOT, name="", palette=1)
```

**Arguments**

- `plot`: plot to add scale to
- `name`: name of the scale (used in the legend)
- `palette`: Color Brewer palette to use, see `brewer.pal` for details. Note that palette type is chosen automatically.

**Examples**

**scgradient**  
*Scale: colour gradient*

**Description**

Scale a continuous variable along a colour gradient.

**Usage**

```r
scgradient(plot = .PLOT, name="", low='red', mid='white', high='black', midpoint=0,
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot</td>
<td>plot object to add scale to</td>
</tr>
<tr>
<td>name</td>
<td>name of the scale (used in the legend)</td>
</tr>
<tr>
<td>low</td>
<td>colour at low end of scale</td>
</tr>
<tr>
<td>mid</td>
<td>colour at middle of scale</td>
</tr>
<tr>
<td>high</td>
<td>colour at top of scale</td>
</tr>
<tr>
<td>midpoint</td>
<td>definition of midpoint</td>
</tr>
<tr>
<td>range</td>
<td>range to scale data to</td>
</tr>
</tbody>
</table>

Details

This scale creates a continuous colour gradient from the low colour to the mid colour to high colour, as defined in the arguments.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

sccolour

Examples

```r
p <- scgradient(ggplot(movies, aes=list(x=mpaa, y=rating)))
ggjitter(p, list(colour=rating))
ggjitter(p, list(colour=length))
p <- ggjitter(p, list(colour=rating))
sccolour(p, low="yellow")
sccolour(p, high="green", midpoint=5)
```

schcl

Scale: colour (hcl)

Description

Scale continuous variables to hue, chroma and luminance components of colour

Usage

```r
schcl(plot = .PLOT, name="", to=list())
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
</tr>
</tbody>
</table>
Details
This colour map is the most perceptually uniform. However, use multiple mappings with care. It is often a good idea to restrict the range of the hue, as shown in the example.

Note: alpha mappings only work with the Quartz and PDF devices.

Author(s)
Hadley Wickham <h.wickham@gmail.com>

See Also
map_colour_hcl, hcl

Examples
```r
p <- schsv(ggplot(movies, aes=list(y=rating, x=year)))
ggpoint(p, list(h=year))
schsv(ggpoint(p, list(h=year)), list(h.to=c(45,60)))
ggpoint(p, list(c=rating))
ggpoint(p, list(l=length))
ggpoint(p, list(h=rating, l=year))
ggpoint(p, list(h=rating, c=year, l=year))
```

schsv

Scale: colour (hsv)

Description
Scale continuous variables to hue, saturation and value components of colour.

Usage
```
schsv(plot = .PLOT, name="", to=list())
```

Arguments
- `plot`
- `name`
- `to`

Details
Use multiple mappings with care

Note: alpha mappings only work with the Quartz and PDF devices.
sclinetype

Description
Create a scale for categorical line types.

Usage
sclinetype(plot = .PLOT, name = "")

Arguments
- plot: plot to add scale to
- name: name of the scale (used in the legend)

Details
This scale is automatically added to the plot when you use the linetype aesthetic. As there are no options to this scale, you shouldn’t ever need to add it yourself.

Author(s)
Hadley Wickham <h.wickham@gmail.com>

See Also
scale_categorical, map_linetype

Examples
p <- ggplot(mtcars, aes=list(x=mpg, y=wt, linetype=cyl))
ggline(p)
ggline(sclinetype(p))
Description

Create a manual scale

Usage

scmanual(plot = .PLOT, variable="x", name="", breaks=NULL, labels=as.character(breaks), grob=function(x) grob_point(x, unique=FALSE))

Arguments

- **plot**: plot object to add scale to
- **variable**: variable to scale
- **name**: name of the scale (used in the legend)
- **breaks**: numeric vector of break points
- **labels**: character vector of break labels
- **grob**: grob function to use when drawing legend

Details

This scale function allows you complete control over the scale.
Supply labels and breaks to produce a legend.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

- [ggfluctuation](#) for a use

Examples
scrgb

Scale: colour (rgb)

Description

Scale continuous variables to red, green and blue components of colour.

Usage

scrgb(plot = .PLOT, name="", to=list())

Arguments

plot  plot to add scale to
name  name of the scale (used in the legend)
to    named list of target ranges (r.to, g.to, b.to, a.to)

Details

The RGB colour space is NOT perceptually uniform. Use this scale with care. It is extremely ill-advised to map variables to more than one of r, g, b, or a.

Note: alpha mappings only work with the Quartz and PDF devices.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

map_colour_rgb, rgb

Examples

p <- scrgb(ggplot(movies, aes=list(y=rating, x=year)))
ggpoint(p, list(r=year))
ggpoint(p, list(b=rating))
ggpoint(p, list(b=rating, r=1))
scrgb(ggpoint(p, list(b=rating, r=1)), list(b.to=c(0.25,0.75)))
ggpoint(p, list(b=rating, r=year))
ggpoint(p, list(b=rating, r=year, g=year))
scshape

Scale: shape

Description

Create a scale for categorical shapes.

Usage

scshape(plot = .PLOT, name="", solid=TRUE)

Arguments

- plot: plot to add scale to
- name: name of the scale (used in the legend)
- solid: should points be solid or hollow?

Details

This scale is automatically added when you use the shape aesthetic mapping. By using this scale you can explicitly decide whether the points used should be hollow or solid.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

scale_categorical, map_shape

Examples

p <- ggplot(mtcars, aes=list(x=mpg, y=wt, shape=cyl))
ggpoint(p)
ggpoint(scshape(p, FALSE))
**scsize**  

*Scale: size*

**Description**

Linearly map size to a variable.

**Usage**

```r
csize(plot = .PLOT, name="", to=c(0.8, 5))
```

**Arguments**

- `plot`: plot to add scale to.
- `name`: name of the scale (used in the legend)
- `to`: size range in mm (numeric vector, length 2)

**Details**

The mapping between size and the original variable value is not linear, but square rooted. This is because the human brain tends to perceive area rather than radius.

You can manipulate the range of the result by modifying the `to` argument.

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

```r
p <- ggplot(mtcars, aes=list(x=mpg, y=hp))
ggpoint(p)
ggpoint(p, list(size=wt))
csize(ggpoint(p, list(size=wt)), c(1,10))
csize(ggpoint(p, list(size=sqrt(wt))), c(1,5))
```

---

**setdata**  

*Set default dataset for a plot*

**Description**

Set the default data set for a plot object

**Usage**

```r
setdata(p = .PLOT, data)
```
Arguments

- **p** plot object, if not specified will use current plot
- **data** new data set

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
setfacets
```

**Description**

Set the function that controls how the plot is facetted into multiple panels.

**Usage**

```
setfacets(p = .PLOT, formula = . ~ ., margins = FALSE)
```

**Arguments**

- **p** plot object, if not specified will use current plot
- **formula** formula describing row and column layout, see `reshape` for more details
- **margins** a vector of names giving which margins to display, can include `grand_row` and `grand_col` or use `TRUE` to display all margins

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**setup_viewports**

Setup viewports

Description

Setup matrix of viewports for a layout with given parameters

Usage

```r
setup_viewports(type, rows=nrow(data), cols=ncol(data), data, offset=c(0,0), range, angle)
```

Arguments

- `type` : viewport type
- `rows` : number of rows
- `cols` : number of columns
- `data` : optional data to compute rows and columns from
- `offset` : offset from top and left
- `range` : list containing x and y ranges
- `angle` :

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
summary.ggplot
```

**summary.ggplot**

Summarise ggplot object

Description

Displays a useful description of a ggplot object

Usage

```r
summary.ggplot(object, ...)
```
Arguments

- object

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```r
tailcolor
```

Tail colour

Description

Colour scheme that emphasizes differences in tails.

Usage

```r
tailcolor(n=100, q1=0.10, q2=0.25, q3 = 1-q2, q4 = 1-q1)
```

Arguments

- **n**: number of colours to produce
- **q1**: quantile for first colour
- **q2**: quantile for second colour
- **q3**: quantile for third colour, defaults to symmetric
- **q4**: quantile for fourth colour, defaults to symmetric

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**tr**

*Traceback*

**Description**
Redefine trace back to work better with *do.call*.

**Usage**

```r
tr(x = NULL)
```

**Arguments**

- `x`

**Details**

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

```r
tryapply
```

*Apply with built in try*

**Description**

**Usage**

```r
tryapply(list, fun, ...)
```

**Arguments**

- `list`
- `fun`
- `...`

**Details**
uneval

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

---

uneval Uneval

Description

Convert an unevaluated list to a list of unevaluated objects

Usage

uneval(x)

Arguments

x unevaluated list (create with substitute)

Details

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
Description

This function updates an entire set of scales with data.

Usage

"update<-".scales"(x, value)

Arguments

x scales object
value data

Details

Update needs to be able to deal with the multiple possible data formats it could receive:
* a single data frame (representing one panel from one grob function)
* a matrix of data frames (all panels from a grob function)
* a list of matrix of data frames (all panels from all grob functions)

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
Details

This is used to teach each scale about the full range of the data so that all panels share a common scale.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

viewport_default  Default viewports

Description

Set up named viewports that the other components use.

Usage

viewport_default(plot, guides=guides_basic(plot), scales=scales_default(plot))

Arguments

plot    plot object
guides  guides grobs
scales  scales grobs

Details

This function sets up a vpTree in which all of the components of a plot will be placed. This allows for a clean separation between the generation of plot objects, and their placement, and means none of the components have to know anything about the others.

This function is responsible for the overall layout of the plot, ie where the panels, labels and axes go. In future, I will add more viewport layout functions so that you can have the same layout as, e.g., the trellis default.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
**vp_name**

*Viewport name*

**Description**

Compute viewport name

**Usage**

\[
\text{vp\_name}(\text{row}, \text{col}, \text{type})
\]

**Arguments**

- **row**: row index
- **col**: column index
- **type**: viewport type

**Details**

This helps ensure a common naming scheme throughout ggplot.

**Author(s)**

Hadley Wickham <h.wickham@gmail.com>

**Examples**

---

**vp_path**

*Viewport path*

**Description**

Calculate viewport path.

**Usage**

\[
\text{vp\_path}(\text{row}, \text{col}, \text{type})
\]

**Arguments**

- **row**: row index
- **col**: column index
- **type**: viewport type
Details

Convience method for calculating the viewport path to a particular entry in a matrix viewport. This helps ensure a common naming scheme throughout ggplot/

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples
Index

*Topic datasets
  IMDB movies data, 79

*Topic file
  ggsave, 53

*Topic hplot
  .map_colour, 74
  add_defaults, 2
  add_position, 3
  add_scale, 4
  axis_vp_path, 6
  blueyellow2, 7
  build_df, 10
  defaultaesthetics, 14
  defaultgrob, 15
  gg2density, 19
  gg_add, 18
  ggabline, 20
  ggarea, 21
  ggaxis, 22
  ggaxis_labels, 23
  ggaxis_line, 23
  ggaxis_ticks, 24
  ggaxis_vp, 25
  ggbar, 25
  ggbboxplot, 27
  ggcontour, 28
  ggdensity, 29
  ggerrorbar, 31
  ggfluctuation, 32
  gggg group, 33
  gghistogram, 34
  gghline, 35
  ggjitter, 36
  ggline, 37
  ggmssing, 39
  ggorder, 40
  ggpath, 40
  ggppc, 41
  ggplot.default, 42
  ggplot_plot, 45
  ggpoint, 47
  ggpolygon, 48
  ggquantile, 50
  ggrect, 51
  ggribbon, 52
  ggsmooth, 54
  ggstrip, 55
  ggstructure, 56
  gtext, 56
  ggtile, 58
  ggvline, 59
  grob_grid, 61
  guides, 64
  guides.default, 63
  guides.position, 64
  guides_basic, 62
  input, 65
  labels_default, 66
  legends, 66
  map_aesthetic, 68
  map_aesthetic.scales, 69
  map_all, 70
  map_colour_gradient, 71
  map_colour_hcl, 72
  map_colour_hsv, 73
  map_colour_rgb, 74
  mosaicdata, 78
  output, 80
  panels_default, 81
  plot_add, 82
  plot_add_grobs, 81
  plot_grob_matrix, 83
  plot_layout, 83
  position_apply, 85
  prettyplot, 88
  print.ggplot, 89
  ps_equal, 92
  ps_map, 93
pscategorical, 93
pscontinuous, 94
qplot, 96
resolution, 99
scale_categorical, 101
scale_cont_colour, 102
scale_continuous, 103
scales, 107
sccolour, 108
scfillbrewer, 109
scgradient, 109
scho1, 110
schov, 111
sclinetype, 112
scmanual, 113
scrgb, 114
scshape, 115
scsize, 116
setdata, 116
setfacets, 117
setup_viewports, 118
tailcolor, 119
viewport_default, 123
vp_name, 124
vp_path, 124
*Topic internal
.map_colour, 74
add_defaults, 2
add_position, 3
add_scale, 4
aesdefaults, 4
alpha, 5
axis_vp_path, 6
best_scale, 6
blueyellow2, 7
boxplot.weighted.formula, 8
breaks, 8
brewer_palettes, 9
brewer_type, 10
build_df, 10
defaultgrob, 15
dhist, 16
facet, 17
get.names, 18
gg_add, 18
ggarea, 21
ggaxis, 22
ggaxis_labels, 23
ggaxis_line, 23
ggaxis_ticks, 24
ggaxis_vp, 25
gdist, 30
ggplot_plot, 45
ggpretty, 49
ggstrip, 55
grob_grid, 61
grob_XXX, 60
grobPromise, 61
guides, 64
guides_basic, 62
input, 65
labels_default, 66
legends, 66
make_all_grobs, 67
make_grobs, 68
map_aesthetic, 68
map_aesthetic.scales, 69
map_all, 70
match.fun.null, 77
mdf, 77
mosaicdata, 78
output, 80
panels_default, 81
plot_add, 82
plot_add_grobs, 81
plot_grob_matrix, 83
plot_layout, 83
position_adjust, 84
position_apply, 85
possible_scales, 86
preprocess, 86
preprocess_all, 87
print.categorical, 89
print.ggplot, 89
print.manual, 90
print.options, 91
print.scale, 91
pstime, 95
range.scale, 97
resolution, 99
sc, 99
scale_apply_combine_map, 100
scale_apply_to_data, 101
scale_categorical, 101
scale_cont_colour, 102
scale_continuous, 103
INDEX

scale_map, 104
scale_mapping, 105
scale_new, 106
scale_quality, 106
scales, 107
setup_viewports, 118
summary.ggplot, 118
tailcolor, 119
trailers, 120
tryapply, 120
uneval, 121
viewport_default, 123
vp_name, 124
vp_path, 124

*Topic manip

.. build_options, 11
breaks, 8
chop, 12
chop.breaks, 14
chop_auto, 13
expand_range, 16
facet, 17
ggtheme, 57
grobPromise, 61
make_all_grobs, 67
make_grobs, 68
map_colour_brewer, 70
map_linetype, 75
map_shape, 76
preprocess, 86
preprocess_all, 87
print.categorical, 89
print.manual, 90
print.scale, 91
range.scale, 97
rescale, 98
scale_mapping, 105
trailers, 120
uneval, 121
.. build_options, 11
.. map_colour, 74

add_defaults, 2
add_position, 3
add_scale, 3, 4
aesdefaults, 4
alpha, 5
axis_vp_path, 6
best_scale, 6
blueyellow2, 7
boxplot, 27
boxplot.weighted
  (boxplot.weighted.formula), 8
boxplot.weighted.formula, 8
boxplot_stats_weighted
  (boxplot.weighted.formula), 8
breaks, 8
breaks.time (pstime), 95
brewer.pal, 109
brewer_palettes, 9
brewer_type, 10
build_df, 10

chop, 12, 102, 108
chop.breaks, 14
chop_auto, 13, 71, 76, 102, 108
deafultgrob.scale (defaultgrob), 15
defaultaesthetics, 14
defaultgrob, 15
defaultgrob.manual (sc), 99
density, 29, 30
dhist, 16
do.call, 120

expand.grid.df, 11
expand_range, 16, 95

facet, 17
frameGrob, 67
gam, 54
get.names, 18
gg2density, 19, 29, 43
gg_add, 18
ggabline, 20, 43
ggarea, 21, 43
ggaxis, 22
ggaxis_labels, 23
ggaxis_line, 23
ggaxis_ticks, 24
ggaxis_vp, 25
ggbar, 25, 32, 39, 43, 51, 52
ggboxplot, 27, 43
ggcontour, 19, 20, 28, 43
ggdensity, 29, 43
ggdist, 30
ggerrorbar, 31
ggfluctuation, 32, 113
gggroup, 33
gghistogram, 30, 34, 44
gghline, 35
ggjitter, 36, 43
gglines, 30, 37, 43
ggmissing, 39
ggopt, 44, 57, 62
ggopt (.build_options), 11
ggorder, 39, 40
ggpath, 40, 43
ggpcp, 41
ggplot, 19, 21, 26–29, 31, 33, 34, 37, 38, 41, 47, 48, 50–52, 54, 57, 58
ggplot (ggplot.default), 42
ggplot.default, 42
ggplot.plot, 3, 45, 89
ggpoint, 43, 47
ggpolygon, 48
ggpretty, 49
ggquantile, 27, 44, 50
ggrect, 26, 35, 51, 59
ggribbon, 43, 52
ggsave, 53
ggsmooth, 44, 54
ggstrip, 55
ggstructure, 39, 56
ggtex, 43, 56
ggtheme, 57
ggtile, 43, 51, 52, 58, 99
ggvline, 59
glm, 54
ggrid.layout, 83
ggrid.pretty, 49
grob_2density (grob_XXX), 60
grob_abline (grob_XXX), 60
grob_area (grob_XXX), 60
grob_bar (grob_XXX), 60
grob_boxplot, 50
grob_boxplot (grob_XXX), 60
grob_contour (grob_XXX), 60
grob_density (grob_XXX), 60
grob_errorbar (grob_XXX), 60
grob_grid, 61
grob_group (grob_XXX), 60
grob_histogram (grob_XXX), 60
grob_hline (grob_XXX), 60
grob_jitter (grob_XXX), 60
grob_line (grob_XXX), 60
grob_path, 28
grob_path (grob_XXX), 60
grob_point (grob_XXX), 60
grob_polygon (grob_XXX), 60
grob_quantile (grob_XXX), 60
grob_rect (grob_XXX), 60
grob_ribbon (grob_XXX), 60
grob_smooth (grob_XXX), 60
grob_text (grob_XXX), 60
grob_tile (grob_XXX), 60
grob_vline (grob_XXX), 60
grob_XXX, 60
grobPromise, 61
gguides, 64
gguides.default, 63
gguides.manual (sc), 99
gguides.position, 64
gguides.scales (scales), 107
gguides_basic, 62
hcl, 111
hist, 35
hsv, 112
image, 58
IMDB movies data, 79
input, 65
input.scales (scales), 107
jitter, 37
kde2d, 19
labels.categorical (sc), 99
labels.continuous (sc), 99
labels.gradient (sc), 99
labels_manual (sc), 99
labels.ps_double (sc), 99
labels.ps_map (sc), 99
labels.time (pstime), 95
labels_default, 66
legends, 66
levelplot, 58
levels, 94
INDEX

lm, 54
loess, 54

make_all_grobs, 67
make_grobs, 68
map_aesthetic, 68
map_aesthetic.scales, 69
map_all, 70
map_color(map_colour_brewer), 70
map_color_brewer
  (map_colour_brewer), 70
map_color_gradient
  (map_colour_gradient), 71
map_color_hcl(map_colour_hcl), 72
map_color_hsv(map_colour_hsv), 73
map_color_rgb(map_colour_rgb), 74
map_colour, 102, 108
map_colour(map_colour_brewer), 70
map_colour_brewer, 70
map_colour_gradient, 71
map_colour_hcl, 72, 111
map_colour_hsv, 73, 74, 112
map_colour_rgb, 74, 114
map_linetype, 75, 102, 112
map_shape, 76, 115
mapproject, 93
match.fun, 77
match.fun.null, 77
mdf, 77
melt, 42
mosaicdata, 78
movies(IMDB movies data), 79

output, 80
output.scales(scales), 107

package-ggplot(ggplot.default), 42
panels_default, 81
plot_add, 11, 82
plot_add_grobs, 81
plot_grob_matrix, 83
plot_layout, 83
plotmosaicdata(mosaicdata), 78
position_adjust, 84
position_apply, 85
position_categorical(sc), 99
position_continuous(sc), 99
position_time(pstime), 95
possible_scales, 6, 86
pre_bar(grob XXX), 60
pre_density(grob XXX), 60
pre_errorbar(grob XXX), 60
pre_group(grob XXX), 60
pre_histogram(grob XXX), 60
preprocess, 86
preprocess_all, 87
prettyplot, 88
print.categorical, 89
print.continuous(sc), 99
print.ggplot, 89
print.gradient(sc), 99
print.manual, 90
print.options, 91
print.scale, 91
print.scales(scales), 107
ps_equal, 92
ps_map, 93
pscategorical, 19, 21, 26–29, 31, 33, 34,
  37, 38, 41, 44, 47, 48, 50–52, 54, 57,
  58, 93, 102
pscontinuous, 19, 21, 26–29, 31, 33, 34,
  37, 38, 41, 44, 47, 48, 50–52, 54, 57,
  58, 94, 104
pstime, 95

qplot, 43, 96

range.categorical(sc), 99
range.continuous(sc), 99
range.ps_double(sc), 99
range.scale, 97
range.scales(scales), 107
range.time(pstime), 95
range2(scale_new), 106
rectGrob, 51
rescale, 98
rescaler, 40, 56
reshape, 43, 44, 117
resolution, 59, 99
rgb, 114
rlm, 54
rg, 50

sc, 99
scale_apply_combine_map, 100
scale_apply_to_data, 101
scale_categorical, 101, 108, 112, 115
scale_colour(sc), 99
scale_cont_colour, 102
scale_continuous, 103
scale_desired_size(scale_new), 106
scale_domain(scale_new), 106
scale_fill(sc), 99
scale_fill_brewer(sc), 99
scale_fill_hcl(sc), 99
scale_fill_hsv(sc), 99
scale_fill_rgb(sc), 99
scale_fill_gradient(sc), 99
tailcolor, 119
textGrob, 57
theme_bw(.build_options), 11
theme_default(.build_options), 11
time_scale(pstime), 95
tr, 120
trans_inverse(scale_continuous), 103
trans_log10(scale_continuous), 103
trans_log2(scale_continuous), 103
trans_none(scale_continuous), 103
trans_sqrt(scale_continuous), 103
tryapply, 120
tryNULL(tryapply), 120
uneval, 121
viewport_default, 123
vp_name, 124
vp_path, 124
vpPath, 62
vpTree, 123

scfill, 44
scfill(sccolour), 108
scfillbrewer, 109
cscfillgradient, 44
cscfillgradient(scgradient), 109
cscfillhcl, 44
cscfillhcl(schcl), 110
scfillhsv, 44
cscfillhsv(schsv), 111
cscfillrgb, 44
cscfillrgb(scrrgb), 114
cscgradient, 44, 104, 109
cschcl, 44, 103, 110
cschsv, 44, 103, 111
cscstinlinetype, 21, 38, 41, 44, 102, 112
scsmanual, 44, 113
scrrgb, 44, 103, 114
cscshape, 37, 44, 47, 102, 115
cscsize, 31, 37, 38, 41, 44, 47, 48, 54, 57,