Package 'ggplot2'

January 8, 2011

Type Package

Title An implementation of the Grammar of Graphics

Version 0.8.9

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 sophisticated multidimensional conditioning system and a consistent interface to map data to aesthetic attributes. See the ggplot2 website for more information, documentation and examples.

Depends reshape (>= 0.8.0), grid, proto

Imports plyr (>= 1.0), splines, MASS, RColorBrewer, digest, colorspace

Suggests quantreg, Hmisc, mapproj, maps, hexbin, gpclib, maptools

Extends sp

License GPL-2

URL http://had.co.nz/ggplot2/

LazyLoad false

LazyData true

Collate aaa-.r aaa-compare.r aaa-constants.r aaa-examples.r aaa-html.r aaa-rdoc.r aes.r annotation.r coord-.r coord-cartesian-.r coord-cartesian-equal.r coord-cartesian-flipped.r coord-map.r coord-polar.r coord-transform.r date-time-breaks.r date-time.r facet-.r facet-grid-.r facet-labels.r facet-viewports.r facet-wrap.r formats.r fortify-lm.r fortify-map.r fortify-spatial.r fortify.r geom-.r geom-abline.r geom-bar-.r

geom-bar-histogram.r geom-bin2d.r geom-blank.r geom-boxplot.r
geom-crossbar.r geom-defaults.r geom-error.r geom-errorh.r
geom-freqpoly.r geom-hex.r geom-hline.r geom-linerange.r
geom-path-.r geom-path-contour.r geom-path-density2d.r
geom-path-line.r geom-path-step.r geom-point-.r
geom-point-jitter.r geom-pointrange.r geom-polygon.r
geom-quantile.r geom-rect.r geom-ribbon-.r
geom-ribbon-density.r geom-rug.r geom-segment.r geom-smooth.r
geom-text.r geom-tile.r geom-vline.r grob-absolute.r
grob-background.r grob-grid.r grob-null.r guides-axis.r
guides-grid.r guides-legend.r labels.r layer.r matrix.r
plot-build.r plot-construction.r plot-last.r plot-render.r
plot-surrounds.r plot.r position-.r position-collide.r
position-dodge.r position-fill.r position-identity.r
position-jitter.r position-stack.r quick-plot.r save.r scale-.r
scale-continuous-.r scale-continuous-alpha.r
scale-continuous-colour.r scale-convenience.r scale-date.r
scale-datetime.r scale-defaults.r scale-discrete-.r scale-discrete-colour.r scale-discrete-grey.r
scale-discrete-position.r scale-identity.r scale-linetype.r
scale-manual.r scale-shape.r scale-size.r scales-.r stat-.r
stat-bin.r stat-bin2d.r stat-binhex.r stat-boxplot.r
stat-contour.r stat-density-2d.r stat-density.r stat-function.r
stat-identity.r stat-qq.r stat-quantile.r stat-smooth-methods.r
stat-smooth.r stat-spoke.r stat-sum.r stat-summary.r
stat-unique.r stat-vline.r summary.r templates.r
theme-defaults.r theme-elements.r theme.r trans-.r trans-scales.r utilities-break.r utilities-colour.r
utilities-discrete.r utilities-facet.r utilities-grid.r
utilities-layer.r utilities-matrix.r utilities-position.r
utilities-resolution.r utilities.r xxx-codegen.r xxx-digest.r xxx.r zxx.r coord-munch.r

Repository CRAN

Date/Publication 2010-12-23 18:17:13

R topics documented:

`aes` *Generate aesthetic mappings*

Description

Aesthetic mappings describe how variables in the data are mapped to visual properties (aesthetics) of geoms.

Usage

```
aes(x, y, ...)
```

Arguments

x	x value
y	y value
...	List of name value pairs

Details

aes creates a list of unevaluated expressions. This function also performs partial name matching, converts color to colour, and old style R names to new ggplot names (eg. pch to shape, cex to size)

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

aes_string

Examples

```
aes(x = mpg, y = wt)
aes(x = mpg ^ 2, y = wt / cyl)
```

---

borders *Map borders.*

Description

Create a layer of map borders

Usage

```
borders(database = "world", regions = ".", fill = NA, colour = "grey50", ...)
```

Arguments

database	map data, see map for details
regions	map region
fill	fill colour
colour	border colour
...	other arguments passed on to geom_polygon

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
if (require(maps)) {
ia <- map_data("county", "iowa")
mid_range <- function(x) mean(range(x))
seats <- ddply(ia, .(subregion), colwise(mid_range, .(lat, long)))
ggplot(ia, aes(long, lat)) +
geom_polygon(aes(group = group), fill = NA, colour = "grey60") +
geom_text(aes(label = subregion), data = seats, size = 2, angle = 45)

data(us.cities)
capitals <- subset(us.cities, capital == 2)
ggplot(capitals, aes(long, lat)) +
borders("state") +
geom_point(aes(size = pop)) +
scale_area()
}
```

`comma` *Comma formatter*

Description

Format number with commas separating thousands

Usage

```
comma(x, ...)
```

Arguments

`x`	numeric vector to format
`...`	other arguments passed on to `format`

Author(s)

Hadley Wickham <h.wickham@gmail.com>

coord_cartesian *coord_cartesian*

Description

Cartesian coordinates

Usage

```
coord_cartesian(xlim = NULL, ylim = NULL, wise = FALSE, ...)
```

Arguments

xlim	x limits
ylim	y limits
wise	NULL
...	ignored

Details

The Cartesian coordinate system is the most familiar, and common, type of coordinate system. There are no options to modify, and it is used by default, so you shouldn't need to call it explicitly

This page describes coord_cartesian, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/coord_cartesian.html

Examples

```
## Not run:
# There are two ways of zooming the plot display: with scales or
# with coordinate systems.  They work in two rather different ways.

(p <- qplot(disp, wt, data=mtcars) + geom_smooth())

# Setting the limits on a scale will throw away all data that's not
# inside these limits.  This is equivalent to plotting a subset of
# the original data
```

```
p + scale_x_continuous(limits = c(325, 500))

# Setting the limits on the coordinate system performs a visual zoom
# the data is unchanged, and we just view a small portion of the original
# plot.  See how the axis labels are the same as the original data, and
# the smooth continue past the points visible on this plot.
p + coord_cartesian(xlim = c(325, 500))

# You can see the same thing with this 2d histogram
(d <- ggplot(diamonds, aes(carat, price)) +
  stat_bin2d(bins = 25, colour="grey50"))

# When zooming the scale, the we get 25 new bins that are the same
# size on the plot, but represent smaller regions of the data space
d + scale_x_continuous(limits = c(0, 2))

# When zooming the coordinate system, we see a subset of original 50 bins,
# displayed bigger
d + coord_cartesian(xlim = c(0, 2))


## End(Not run)
```

`coord_fixed` *coord_fixed*

Description

Cartesian coordinates with fixed relationship between x and y scales.

Usage

```
coord_fixed(ratio = 1, ...)
```

Arguments

`ratio`	NULL
`...`	ignored

Details

A fixed scale coordinate system forces a specified ratio between the physical representation of data units on the axes. The ratio represents the number of units on the y-axis equivalent to one unit on the x-axis. The default, ratio = 1, ensures that one unit on the x-axis is the same length as one unit on the y-axis. Ratios higher than one make units on the y axis longer than units on the x-axis, and vice versa. This is similar to ?eqscplot in MASS, but it works for all types of graphics

This page describes coord_fixed, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/coord_fixed.html

Examples

```
## Not run:
# ensures that the ranges of axes are equal to the specified ratio by
# adjusting the plot aspect ratio

qplot(mpg, wt, data = mtcars) + coord_equal(ratio = 1)
qplot(mpg, wt, data = mtcars) + coord_equal(ratio = 5)
qplot(mpg, wt, data = mtcars) + coord_equal(ratio = 1/5)

# Resize the plot to see that the specified aspect ratio is mantained

## End(Not run)
```

---

coord_flip *coord_flip*

Description

Flipped cartesian coordinates

Usage

```
coord_flip(xlim = NULL, ylim = NULL, wise = FALSE, ...)
```

Arguments

xlim	x limits
ylim	y limits
wise	NULL
...	ignored

Details

Flipped cartesian coordinates so that horizontal becomes vertical, and vertical, horizontal. This is primarily useful for converting geoms and statistics which display y conditional on x, to x conditional on y

This page describes coord_flip, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/coord_flip.html

Examples

```
## Not run:
# Very useful for creating boxplots, and other interval
# geoms in the horizontal instead of vertical position.
qplot(cut, price, data=diamonds, geom="boxplot")
last_plot() + coord_flip()

qplot(cut, data=diamonds, geom="bar")
last_plot() + coord_flip()

qplot(carat, data=diamonds, geom="histogram")
last_plot() + coord_flip()

# You can also use it to flip lines and area plots:
qplot(1:5, (1:5)^2, geom="line")
last_plot() + coord_flip()

## End(Not run)
```

---

coord_map *coord_map*

Description

Map projections

Usage

```
coord_map(projection = "mercator", orientation = NULL, xlim = NULL,
    ylim = NULL, fast = TRUE, ...)
```

Arguments

projection	projection to use, see ?mapproject for complete list
orientation	orientation, which defaults to c(90, 0, mean(range(x))). This is not optimal for many projections, so you will have to supply your own.
xlim	x limits
ylim	y limits
fast	NULL
...	other arguments passed on to mapproject

Details

This coordinate system provides the full range of map projections available in the mapproj package.

This is still experimental, and if you have any advice to offer regarding a better (or more correct) way to do this, please let me know

This page describes coord_map, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/coord_map.html

Examples

```
## Not run:
try_require("maps")
# Create a lat-long dataframe from the maps package
nz <- data.frame(map("nz", plot=FALSE)[c("x","y")])
(nzmap <- qplot(x, y, data=nz, geom="path"))

nzmap + coord_map()
nzmap + coord_map(project="cylindrical")
nzmap + coord_map(project='azequalarea',orientation=c(-36.92,174.6,0))

states <- data.frame(map("state", plot=FALSE)[c("x","y")])
(usamap <- qplot(x, y, data=states, geom="path"))
```

```
usamap + coord_map()
# See ?mapproject for coordinate systems and their parameters
usamap + coord_map(project="gilbert")
usamap + coord_map(project="lagrange")

# For most projections, you'll need to set the orientation yourself
# as the automatic selection done by mapproject is not available to
# ggplot
usamap + coord_map(project="orthographic")
usamap + coord_map(project="stereographic")
usamap + coord_map(project="conic", lat0 = 30)
usamap + coord_map(project="bonne", lat0 = 50)

## End(Not run)
```

`coord_polar` *coord_polar*

Description

Polar coordinates

Usage

```
coord_polar(theta = "x", start = 0, direction = 1, expand = FALSE,
    ...)
```

Arguments

`theta`	variable to map angle to ('x' or 'y')
`start`	offset from 12 o'clock in radians
`direction`	1, clockwise; -1, anticlockwise
`expand`	should axes be expanded to slightly outside the range of the data? (default: FALSE)
`...`	other arguments

Details

The polar coordinate system is most commonly used for pie charts, which are a stacked bar chart in polar coordinates.

This coordinate system has one argument, `theta`, which determines which variable is mapped to angle and which to radius. Valid values are "x" and "y".

This page describes coord_polar, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/coord_polar.html

Examples

```
## Not run:
# NOTE: Use these plots with caution - polar coordinates has
# major perceptual problems.  The main point of these examples is
# to demonstrate how these common plots can be described in the
# grammar.  Use with EXTREME caution.

# A coxcomb plot = bar chart + polar coordinates
cxc <- ggplot(mtcars, aes(x = factor(cyl))) +
  geom_bar(width = 1, colour = "black")
cxc + coord_polar()
# A new type of plot?
cxc + coord_polar(theta = "y")

# A pie chart = stacked bar chart + polar coordinates
pie <- ggplot(mtcars, aes(x = factor(1), fill = factor(cyl))) +
 geom_bar(width = 1)
pie + coord_polar(theta = "y")

# The bullseye chart
pie + coord_polar()

# Hadley's favourite pie chart
df <- data.frame(
  variable = c("resembles", "does not resemble"),
  value = c(80, 20)
)
ggplot(df, aes(x = "", y = value, fill = variable)) +
  geom_bar(width = 1) +
  scale_fill_manual(values = c("red", "yellow")) +
  coord_polar("y", start=pi / 3) +
  opts(title = "Pac man")

# Windrose + doughnut plot
movies$rrating <- cut_interval(movies$rating, length = 1)
movies$budgetq <- cut_number(movies$budget, 4)

doh <- ggplot(movies, aes(x = rrating, fill = budgetq))

# Wind rose
```

```
doh + geom_bar(width = 1) + coord_polar()
# Race track plot
doh + geom_bar(width = 0.9, position = "fill") + coord_polar(theta = "y")

## End(Not run)
```

---

coord_trans *coord_trans*

Description

Transformed cartesian coordinate system

Usage

```
coord_trans(xtrans = "identity", ytrans = "identity", ...)
```

Arguments

xtrans	NULL
ytrans	NULL
...	ignored

Details

This page describes coord_trans, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/coord_trans.html

Examples

```
## Not run:
# See ?geom_boxplot for other examples

# Three ways of doing transformating in ggplot:
#  * by transforming the data
qplot(log10(carat), log10(price), data=diamonds)
#  * by transforming the scales
qplot(carat, price, data=diamonds, log="xy")
qplot(carat, price, data=diamonds) + scale_x_log10() + scale_y_log10()
#  * by transforming the coordinate system:
qplot(carat, price, data=diamonds) + coord_trans(x = "log10", y = "log10")

# The difference between transforming the scales and
# transforming the coordinate system is that scale
# transformation occurs BEFORE statistics, and coordinate
# transformation afterwards.  Coordinate transformation also
# changes the shape of geoms:

d <- subset(diamonds, carat > 0.5)
qplot(carat, price, data = d, log="xy") +
  geom_smooth(method="lm")
qplot(carat, price, data = d) +
  geom_smooth(method="lm") +
  coord_trans(x = "log10", y = "log10")

# Here I used a subset of diamonds so that the smoothed line didn't
# drop below zero, which obviously causes problems on the log-transformed
# scale

# With a combination of scale and coordinate transformation, it's
# possible to do back-transformations:
qplot(carat, price, data=diamonds, log="xy") +
  geom_smooth(method="lm") +
  coord_trans(x="pow10", y="pow10")
# cf.
qplot(carat, price, data=diamonds) + geom_smooth(method = "lm")


## End(Not run)
```

---

cut_interval *Discretise continuous variable, equal interval length.*

Description

Cut numeric vector into intervals of equal length.

Usage

```
cut_interval(x, n = NULL, length = NULL, ...)
```

Arguments

`x`	numeric vector
`n`	number of intervals to create, OR
`length`	length of each interval
`...`	other arguments passed on to `cut`

Details

@arguments numeric vector @arguments number of intervals to create, OR @arguments length of each interval @arguments other arguments passed on to `cut` @keyword manip @seealso `cut_number`

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

`cut_number`

Examples

```
table(cut_interval(1:100, n = 10))
table(cut_interval(1:100, n = 11))
table(cut_interval(1:100, length = 10))
```

`cut_number` *Discretise continuous variable, equal number of points.*

Description

Cut numeric vector into intervals containing equal number of points.

Usage

```
cut_number(x, n = NULL, ...)
```

Arguments

`x`	numeric vector
`n`	number of intervals to create, OR
`...`	length of each interval
	other arguments passed on to `cut`

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

cut_interval

Examples

```
table(cut_number(runif(1000), n = 10))
```

Diamond prices *Prices of 50,000 round cut diamonds*

Description

A dataset containing the prices and other attributes of almost 54,000 diamonds. The variables are as follows:

- price. price in US dollars (\$326–\$18,823)
- carat. weight of the diamond (0.2–5.01)
- cut. quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- colour. diamond colour, from J (worst) to D (best)
- clarity. a measurement of how clear the diamond is (I1 (worst), SI1, SI2, VS1, VS2, VVS1, VVS2, IF (best))
- x. length in mm (0–10.74)
- y. width in mm (0–58.9)
- z. depth in mm (0–31.8)
- depth. total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43–79)
- table. width of top of diamond relative to widest point (43–95)

Usage

```
data(diamonds)
```

Format

A data frame with 53940 rows and 10 variables

dist_central_angle *Compute central angle between two points.*

Description

Multiple by radius of sphere to get great circle distance

Usage

```
dist_central_angle(lon, lat)
```

Arguments

lon	longitude
lat	latitude

Author(s)

Hadley Wickham <h.wickham@gmail.com>

dollar *Currency formatter*

Description

Round to nearest cent and display dollar sign

Usage

```
dollar(x, ...)
```

Arguments

x	numeric vector to format
...	other arguments passed on to format

Author(s)

Hadley Wickham <h.wickham@gmail.com>

expand_limits *Expand the plot limits with data.*

Description

Some times you may want to ensure limits include a single value, for all panels or all plots. This function is a thin wrapper around geom_blank that makes it easy to add such values.

Usage

```
expand_limits(...)
```

Arguments

...	named list of aesthetics specifying the value (or values that should be included.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
p <- qplot(mpg, wt, data = mtcars)
p + expand_limits(x = 0)
p + expand_limits(y = c(1, 9))
p + expand_limits(x = 0, y = 0)

qplot(mpg, wt, data = mtcars, colour = cyl) +
expand_limits(colour = seq(2, 10, by = 2))
qplot(mpg, wt, data = mtcars, colour = factor(cyl)) +
expand_limits(colour = factor(seq(2, 10, by = 2)))
```

expand_range *Expand range*

Description

Convenience function for expanding a range with a multiplicative or additive constant.

Usage

```
expand_range(range, mul = 0, add = 0, zero = 0.5)
```

Arguments

range	range of data
mul	multiplicative constract
add	additive constant
zero	distance to use if range has zero width

Author(s)

Hadley Wickham <h.wickham@gmail.com>

---

facet_grid *facet_grid*

Description

Lay out panels in a rectangular/tabular manner.

Usage

```
facet_grid(facets = . ~ ., margins = FALSE, scales = "fixed",
    space = "fixed", labeller = "label_value", as.table = TRUE,
    widths = NULL, heights = NULL, ...)
```

Arguments

facets	a formula with the rows (of the tabular display) on the LHS and the columns (of the tabular display) on the RHS; the dot in the formula is used to indicate there should be no faceting on this dimension (either row or column); the formula can also be entered as a string instead of a classical formula object
margins	logical value, should marginal rows and columns be displayed
scales	NULL
space	NULL
labeller	NULL
as.table	NULL
widths	NULL
heights	NULL
...	other arguments

Details

This page describes facet_grid, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/facet_grid.html

Examples

```
## Not run:
# faceting displays subsets of the data in different panels
p <- ggplot(diamonds, aes(carat, ..density..)) +
 geom_histogram(binwidth = 1)

# With one variable
p + facet_grid(. ~ cut)
p + facet_grid(cut ~ .)

# With two variables
p + facet_grid(clarity ~ cut)
p + facet_grid(cut ~ clarity)
# p + facet_grid(cut ~ clarity, margins=TRUE)

qplot(mpg, wt, data=mtcars, facets = . ~ vs + am)
qplot(mpg, wt, data=mtcars, facets = vs + am ~ . )

# You can also use strings, which makes it a little easier
# when writing functions that generate faceting specifications
# p + facet_grid("cut ~ .")

# see also ?plotmatrix for the scatterplot matrix

# If there isn't any data for a given combination, that panel
# will be empty
qplot(mpg, wt, data=mtcars) + facet_grid(cyl ~ vs)

# If you combine a facetted dataset with a dataset that lacks those
# facetting variables, the data will be repeated across the missing
# combinations:
p <- qplot(mpg, wt, data=mtcars, facets = vs ~ cyl)

df <- data.frame(mpg = 22, wt = 3)
p + geom_point(data = df, colour="red", size = 2)

df2 <- data.frame(mpg = c(19, 22), wt = c(2,4), vs = c(0, 1))
p + geom_point(data = df2, colour="red", size = 2)

df3 <- data.frame(mpg = c(19, 22), wt = c(2,4), vs = c(1, 1))
```

```
p + geom_point(data = df3, colour="red", size = 2)


# You can also choose whether the scales should be constant
# across all panels (the default), or whether they should be allowed
# to vary
mt <- ggplot(mtcars, aes(mpg, wt, colour = factor(cyl))) + geom_point()

mt + facet_grid(. ~ cyl, scales = "free")
# If scales and space are free, then the mapping between position
# and values in the data will be the same across all panels
mt + facet_grid(. ~ cyl, scales = "free", space = "free")

mt + facet_grid(vs ~ am, scales = "free")
mt + facet_grid(vs ~ am, scales = "free_x")
mt + facet_grid(vs ~ am, scales = "free_y")
mt + facet_grid(vs ~ am, scales = "free", space="free")

# You may need to set your own breaks for consitent display:
mt + facet_grid(. ~ cyl, scales = "free_x", space="free") +
  scale_x_continuous(breaks = seq(10, 36, by = 2))
# Adding scale limits override free scales:
last_plot() + xlim(10, 15)

# Free scales are particularly useful for categorical variables
qplot(cty, model, data=mpg) +
  facet_grid(manufacturer ~ ., scales = "free", space = "free")
# particularly when you reorder factor levels
mpg <- within(mpg, {
  model <- reorder(model, cty)
  manufacturer <- reorder(manufacturer, cty)
})
last_plot()

## End(Not run)
```

`facet_wrap` *facet_wrap*

Description

Wrap a 1d ribbon of panels into 2d.

Usage

```
facet_wrap(facets, nrow = NULL, ncol = NULL, scales = "fixed",
    as.table = TRUE, drop = TRUE, ...)
```

Arguments

facets	NULL
nrow	number of rows
ncol	number of columns
scales	should scales be fixed, free, or free in one dimension (free_x, free_y)
as.table	NULL
drop	NULL
...	other arguments

Details

This page describes facet_wrap, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/facet_wrap.html

Examples

```
## Not run:
d <- ggplot(diamonds, aes(carat, price, fill = ..density..)) +
  xlim(0, 2) + stat_binhex(na.rm = TRUE) + opts(aspect.ratio = 1)
d + facet_wrap(~ color)
d + facet_wrap(~ color, ncol = 1)
d + facet_wrap(~ color, ncol = 4)
d + facet_wrap(~ color, nrow = 1)
d + facet_wrap(~ color, nrow = 3)

# Using multiple variables continues to wrap the long ribbon of
# plots into 2d - the ribbon just gets longer
# d + facet_wrap(~ color + cut)

# You can choose to keep the scales constant across all panels
# or vary the x scale, the y scale or both:
p <- qplot(price, data = diamonds, geom = "histogram", binwidth = 1000)
p + facet_wrap(~ color)
p + facet_wrap(~ color, scales = "free_y")

p <- qplot(displ, hwy, data = mpg)
p + facet_wrap(~ cyl)
```

```
p + facet_wrap(~ cyl, scales = "free")

# Add data that does not contain all levels of the faceting variables
cyl6 <- subset(mpg, cyl == 6)
p + geom_point(data = cyl6, colour = "red", size = 1) +
  facet_wrap(~ cyl)
p + geom_point(data = transform(cyl6, cyl = 7), colour = "red") +
  facet_wrap(~ cyl)
p + geom_point(data = transform(cyl6, cyl = NULL), colour = "red") +
  facet_wrap(~ cyl)

# By default, any empty factor levels will be dropped
mpg$cyl2 <- factor(mpg$cyl, levels = c(2, 4, 5, 6, 8, 10))
qplot(displ, hwy, data = mpg) + facet_wrap(~ cyl2)
# Use drop = FALSE to force their inclusion
qplot(displ, hwy, data = mpg) + facet_wrap(~ cyl2, drop = FALSE)

## End(Not run)
```

`fortify` *Fortify a model with data*

Description

Generic method to supplement the original data with model fit statistics

Usage

```
fortify(model, data, ...)
```

Arguments

`model`	model
`data`	dataset
`...`	other arguments passed to methods

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

`fortify.lm`

`fortify.lm` *Fortify a linear model with its data*

Description

Supplement the data fitted to a linear model with model fit statistics.

Usage

```
fortify.lm(model, data = model$model, ...)
```

Arguments

`model`	linear model
`data`	data set, defaults to data used to fit model
`...`	not used

Details

The following statistics will be added to the data frame:

- .hatDiagonal of the hat matrix
- .sigmaEstimate of residual standard deviation when corresponding observation is dropped from model
- .cooksdCooks distance, `cooks.distance`
- .fittedFitted values of model
- .residResiduals
- .stdresidStandardised residuals

If you have missing values in your model data, you may need to refit the model with `na.action = na.preserve`.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
mod <- lm(mpg ~ wt, data = mtcars)
head(fortify(mod))
head(fortify(mod, mtcars))

plot(mod, which = 1)
qplot(.fitted, .resid, data = mod) + geom_hline() + geom_smooth(se = FALSE)
qplot(.fitted, .stdresid, data = mod) + geom_hline() +
geom_smooth(se = FALSE)
qplot(.fitted, .stdresid, data = fortify(mod, mtcars),
```

```
colour = factor(cyl))
qplot(mpg, .stdresid, data = fortify(mod, mtcars), colour = factor(cyl))

plot(mod, which = 2)
# qplot(sample =.stdresid, data = mod, stat = "qq") + geom_abline()

plot(mod, which = 3)
qplot(.fitted, sqrt(abs(.stdresid)), data = mod) + geom_smooth(se = FALSE)

plot(mod, which = 4)
qplot(seq_along(.cooksd), .cooksd, data = mod, geom = "bar",
stat="identity")

plot(mod, which = 5)
qplot(.hat, .stdresid, data = mod) + geom_smooth(se = FALSE)
ggplot(mod, aes(.hat, .stdresid)) +
geom_vline(size = 2, colour = "white", xintercept = 0) +
geom_hline(size = 2, colour = "white", yintercept = 0) +
geom_point() + geom_smooth(se = FALSE)

qplot(.hat, .stdresid, data = mod, size = .cooksd) +
geom_smooth(se = FALSE, size = 0.5)

plot(mod, which = 6)
ggplot(mod, aes(.hat, .cooksd, data = mod)) +
geom_vline(colour = NA) +
geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") +
geom_smooth(se = FALSE) +
geom_point()
qplot(.hat, .cooksd, size = .cooksd / .hat, data = mod) + scale_area()
```

`fortify.map` *Fortify a map*

Description

Fortify method for map objects

Usage

```
fortify.map(model, data, ...)
```

Arguments

`model`	map object
`data`	ignored
`...`	ignored

Details

This function turns a map into a data frame than can more easily be plotted with ggplot2.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
if (require(maps)) {
ca <- map_data("county", "ca")
qplot(long, lat, data = ca, geom="polygon", group = group)
tx <- map_data("county", "texas")
qplot(long, lat, data = tx, geom="polygon", group = group,
colour = I("white"))
}
```

---

fortify.SpatialPolygonsDataFrame

Fortify spatial polygons and lines

Description

Fortify method for a number of the class from the sp package.

Usage

```
fortify.SpatialPolygonsDataFrame(model, data, region = NULL, ...)
```

Arguments

model	SpatialPolygonsDataFrame
data	not used
region	name of variable to split up regions by
...	not used

Details

To figure out the correct variable name for region, inspect `as.data.frame(model)`.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Fuel economy — *Fuel economy data from 1999 and 2008 for 38 popular models of car*

Description

This dataset contains a subset of the fuel economy data that the EPA makes available on http://fueleconomy.gov. It contains only models which had a new release every year between 1999 and 2008 - this was used as a proxy for the popularity of the car.

- manufacturer.
- model.
- displ. engine displacement, in litres
- year.
- cyl. number of cylinders
- trans. type of transmission
- drv. f = front-wheel drive, r = rear wheel drive, 4 = 4wd
- cty. city miles per gallon
- hwy. highway miles per gallon
- fl.
- class.

Usage

```
data(mpg)
```

Format

A data frame with 234 rows and 11 variables

geom_abline — *geom_abline*

Description

Line, specified by slope and intercept

Usage

```
geom_abline(mapping = NULL, data = NULL, stat = "abline", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

The abline geom adds a line with specified slope and intercept to the plot.

With its siblings geom_hline and geom_vline, it's useful for annotating plots. You can supply the parameters for geom_abline, intercept and slope, in two ways: either explicitly as fixed values, or stored in the data set. If you specify the fixed values (`geom_abline(intercept=0, slope=1)`) then the line will be the same in all panels, but if the intercept and slope are stored in the data, then can vary from panel to panel. See the examples for more ideas.

This page describes geom_abline, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_abline. Aesthetics are mapped to variables in the data with the aes function: `geom_abline(aes(x = var))`

- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `stat_smooth`: To add lines derived from the data
- `geom_hline`: for horizontal lines
- `geom_vline`: for vertical lines
- `geom_segment`: for a more general approach
- http://had.co.nz/ggplot2/geom_abline.html

Examples

```
## Not run:
p <- qplot(wt, mpg, data = mtcars)

# Fixed slopes and intercepts
p + geom_abline() # Can't see it - outside the range of the data
p + geom_abline(intercept = 20)

# Calculate slope and intercept of line of best fit
coef(lm(mpg ~ wt, data = mtcars))
p + geom_abline(intercept = 37, slope = -5)
p + geom_abline(intercept = 10, colour = "red", size = 2)

# See ?stat_smooth for fitting smooth models to data
p + stat_smooth(method="lm", se=FALSE)

# Slopes and intercepts as data
p <- ggplot(mtcars, aes(x = wt, y=mpg), . ~ cyl) + geom_point()
df <- data.frame(a=rnorm(10, 25), b=rnorm(10, 0))
p + geom_abline(aes(intercept=a, slope=b), data=df)

# Slopes and intercepts from linear model
coefs <- ddply(mtcars, .(cyl), function(df) {
  m <- lm(mpg ~ wt, data=df)
  data.frame(a = coef(m)[1], b = coef(m)[2])
})
str(coefs)
p + geom_abline(data=coefs, aes(intercept=a, slope=b))

# It's actually a bit easier to do this with stat_smooth
p + geom_smooth(aes(group=cyl), method="lm")
p + geom_smooth(aes(group=cyl), method="lm", fullrange=TRUE)


## End(Not run)
```

---

geom_area *geom_area*

Description

Area plots

Usage

```
geom_area(mapping = NULL, data = NULL, stat = "identity", position = "stack",
    na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
na.rm	NULL
...	ignored

Details

An area plot is the continuous analog of a stacked bar chart (see geom_bar), and can be used to show how composition of the whole varies over the range of x. Choosing the order in which different components is stacked is very important, as it becomes increasing hard to see the individual pattern as you move up the stack.

An area plot is a special case of geom_ribbon, where the minimum of the range is fixed to 0, and the position adjustment defaults to position_stacked.

This page describes geom_area, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_area. Aesthetics are mapped to variables in the data with the aes function: `geom_area(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- colour: border colour
- fill: internal colour
- size: size
- linetype: line type
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_bar: Discrete intervals (bars)
- geom_linerange: Discrete intervals (lines)
- geom_polygon: General polygons
- http://had.co.nz/ggplot2/geom_area.html

Examples

```
## Not run:
# Examples to come

## End(Not run)
```

`geom_bar` *geom_bar*

Description

Bars, rectangles with bases on x-axis

Usage

```
geom_bar(mapping = NULL, data = NULL, stat = "bin", position = "stack",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

The bar geom is used to produce 1d area plots: bar charts for categorical x, and histograms for continuous y. stat_bin explains the details of these summaries in more detail. In particular, you can use the `weight` aesthetic to create weighted histograms and barcharts where the height of the bar no longer represent a count of observations, but a sum over some other variable. See the examples for a practical example.

By default, multiple x's occuring in the same place will be stacked a top one another by position_stack. If you want them to be dodged from side-to-side, check out position_dodge. Finally, position_fill shows relative propotions at each x by stacking the bars and then stretch or squashing them all to the same height

This page describes geom_bar, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_bar. Aesthetics are mapped to variables in the data with the aes function: `geom_bar(aes(x = var))`

- `x`: x position (**required**)
- `colour`: border colour
- `fill`: internal colour
- `size`: size
- `linetype`: line type
- `weight`: observation weight used in statistical transformation
- `alpha`: transparency

Advice

If you have presummarised data, use <code>stat="identity" to turn off the default summary

Sometimes, bar charts are used not as a distributional summary, but instead of a dotplot. Generally, it's preferable to use a dotplot (see geom_point) as it has a better data-ink ratio. However, if you do want to create this type of plot, you can set y to the value you have calculated, and use stat='identity'.

A bar chart maps the height of the bar to a variable, and so the base of the bar must always been shown to produce a valid visual comparison. Naomi Robbins has a nice article on this topic. This is the reason it doesn't make sense to use a log-scaled y axis.

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `stat_bin`: for more details of the binning alogirithm
- `position_dodge`: for creating side-by-side barcharts
- `position_stack`: for more info on stacking
- http://had.co.nz/ggplot2/geom_bar.html

Examples

```
## Not run:
# Generate data
c <- ggplot(mtcars, aes(factor(cyl)))

c + geom_bar()
c + geom_bar() + coord_flip()
c + geom_bar(fill="white", colour="darkgreen")

# Use qplot
qplot(factor(cyl), data=mtcars, geom="bar")
```

```
qplot(factor(cyl), data=mtcars, geom="bar", fill=factor(cyl))

# Stacked bar charts
qplot(factor(cyl), data=mtcars, geom="bar", fill=factor(vs))
qplot(factor(cyl), data=mtcars, geom="bar", fill=factor(gear))

# Stacked bar charts are easy in ggplot2, but not effective visually,
# particularly when there are many different things being stacked
ggplot(diamonds, aes(clarity, fill=cut)) + geom_bar()
ggplot(diamonds, aes(color, fill=cut)) + geom_bar() + coord_flip()

# Faceting is a good alternative:
ggplot(diamonds, aes(clarity)) + geom_bar() +
  facet_wrap(~ cut)
# If the x axis is ordered, using a line instead of bars is another
# possibility:
ggplot(diamonds, aes(clarity)) +
  geom_freqpoly(aes(group = cut, colour = cut))

# Dodged bar charts
ggplot(diamonds, aes(clarity, fill=cut)) + geom_bar(position="dodge")
# compare with
ggplot(diamonds, aes(cut, fill=cut)) + geom_bar() +
  facet_grid(. ~ clarity)

# But again, probably better to use frequency polygons instead:
ggplot(diamonds, aes(clarity, colour=cut)) +
  geom_freqpoly(aes(group = cut))

# Often we don't want the height of the bar to represent the
# count of observations, but the sum of some other variable.
# For example, the following plot shows the number of diamonds
# of each colour
qplot(color, data=diamonds, geom="bar")
# If, however, we want to see the total number of carats in each colour
# we need to weight by the carat variable
qplot(color, data=diamonds, geom="bar", weight=carat, ylab="carat")

# A bar chart used to display means
meanprice <- tapply(diamonds$price, diamonds$cut, mean)
cut <- factor(levels(diamonds$cut), levels = levels(diamonds$cut))
qplot(cut, meanprice)
qplot(cut, meanprice, geom="bar", stat="identity")
qplot(cut, meanprice, geom="bar", stat="identity", fill = I("grey50"))

## End(Not run)
```

---

geom_bin2d *geom_bin2d*

Description

Add heatmap of 2d bin counts

Usage

```
geom_bin2d(mapping = NULL, data = NULL, stat = "bin2d", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes geom_bin2d, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_bin2d. Aesthetics are mapped to variables in the data with the aes function: `geom_bin2d(aes(x = var))`

- xmin: left (hortizontal minimum) (**required**)
- xmax: right (hortizontal maximum) (**required**)
- ymin: bottom (vertical minimum) (**required**)
- ymax: top (vertical maximum) (**required**)
- colour: border colour
- fill: internal colour
- size: size
- linetype: line type
- weight: observation weight used in statistical transformation
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_bin2d.html

Examples

```
## Not run:
# See ?stat_bin2d

## End(Not run)
```

`geom_blank` *geom_blank*

Description

Blank, draws nothing

Usage

```
geom_blank(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

This page describes geom_blank, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_blank.html

Examples

```
## Not run:
qplot(length, rating, data=movies, geom="blank")
# Nothing to see here!

## End(Not run)
```

`geom_boxplot` *geom_boxplot*

Description

Box and whiskers plot

Usage

```
geom_boxplot(mapping = NULL, data = NULL, stat = "boxplot", position = "dodge",
    outlier.colour = "black", outlier.shape = 16, outlier.size = 2,
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`outlier.colour`	colour for outlying points
`outlier.shape`	shape of outlying points
`outlier.size`	size of outlying points
`...`	other arguments

Details

This page describes geom_boxplot, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_boxplot. Aesthetics are mapped to variables in the data with the aes function: `geom_boxplot(aes(x = var))`

- `x`: x position (**required**)
- `lower`: NULL (**required**)
- `upper`: NULL (**required**)
- `middle`: NULL (**required**)
- `ymin`: bottom (vertical minimum) (**required**)
- `ymax`: top (vertical maximum) (**required**)
- `weight`: observation weight used in statistical transformation
- `colour`: border colour
- `fill`: internal colour
- `size`: size
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `stat_quantile`: View quantiles conditioned on a continuous variable
- `geom_jitter`: Another way to look at conditional distributions
- http://had.co.nz/ggplot2/geom_boxplot.html

Examples

```
## Not run:
p <- ggplot(mtcars, aes(factor(cyl), mpg))

p + geom_boxplot()
qplot(factor(cyl), mpg, data = mtcars, geom = "boxplot")

p + geom_boxplot() + geom_jitter()
p + geom_boxplot() + coord_flip()
qplot(factor(cyl), mpg, data = mtcars, geom = "boxplot") +
  coord_flip()

p + geom_boxplot(outlier.colour = "green", outlier.size = 3)

# Add aesthetic mappings
# Note that boxplots are automatically dodged when any aesthetic is
# a factor
p + geom_boxplot(aes(fill = cyl))
p + geom_boxplot(aes(fill = factor(cyl)))
p + geom_boxplot(aes(fill = factor(vs)))
```

```
p + geom_boxplot(aes(fill = factor(am)))

# Set aesthetics to fixed value
p + geom_boxplot(fill="grey80", colour="#3366FF")
qplot(factor(cyl), mpg, data = mtcars, geom = "boxplot",
  colour = I("#3366FF"))

# Scales vs. coordinate transforms -------
# Scale transformations occur before the boxplot statistics are computed.
# Coordinate transformations occur afterwards.  Observe the effect on the
# number of outliers.
m <- ggplot(movies, aes(y = votes, x = rating,
   group = round_any(rating, 0.5)))
m + geom_boxplot()
m + geom_boxplot() + scale_y_log10()
m + geom_boxplot() + coord_trans(y = "log10")
m + geom_boxplot() + scale_y_log10() + coord_trans(y = "log10")

# Boxplots with continuous x:
# Use the group aesthetic to group observations in boxplots
qplot(year, budget, data = movies, geom = "boxplot")
qplot(year, budget, data = movies, geom = "boxplot",
  group = round_any(year, 10, floor))


## End(Not run)
```

`geom_contour` *geom_contour*

Description

Display contours of a 3d surface in 2d

Usage

```
geom_contour(mapping = NULL, data = NULL, stat = "contour", position = "identity",
    lineend = "butt", linejoin = "round", linemitre = 1, na.rm = FALSE,
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`lineend`	Line end style (round, butt, square)
`linejoin`	Line join style (round, mitre, bevel)

linemitre	Line mitre limit (number greater than 1)
na.rm	NULL
...	other arguments

Details

This page describes geom_contour, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_contour. Aesthetics are mapped to variables in the data with the aes function: geom_contour(aes(x = var))

- x: x position (**required**)
- y: y position (**required**)
- weight: observation weight used in statistical transformation
- colour: border colour
- size: size
- linetype: line type
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_density2d: Draw 2d density contours
- http://had.co.nz/ggplot2/geom_contour.html

Examples

```
## Not run:
# See stat_contour for examples

## End(Not run)
```

`geom_crossbar` *geom_crossbar*

Description

Hollow bar with middle indicated by horizontal line

Usage

```
geom_crossbar(mapping = NULL, data = NULL, stat = "identity",
    position = "identity", fatten = 2, ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`fatten`	a multiplicate factor to fatten middle bar by
`...`	other arguments

Details

This page describes geom_crossbar, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_crossbar. Aesthetics are mapped to variables in the data with the aes function: `geom_crossbar(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `ymin`: bottom (vertical minimum) (**required**)
- `ymax`: top (vertical maximum) (**required**)
- `colour`: border colour
- `fill`: internal colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_errorbar: error bars
- geom_pointrange: range indicated by straight line, with point in the middle
- geom_linerange: range indicated by straight line + examples
- stat_summary: examples of these guys in use
- geom_smooth: for continuous analog
- http://had.co.nz/ggplot2/geom_crossbar.html

Examples

```
## Not run:
# See geom_linerange for examples

## End(Not run)
```

---

geom_density *geom_density*

Description

Display a smooth density estimate

Usage

```
geom_density(mapping = NULL, data = NULL, stat = "density", position = "identity",
    na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
na.rm	NULL
...	ignored

Details

A smooth density estimate calculated by stat_densityThis page describes geom_density, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_density. Aesthetics are mapped to variables in the data with the aes function: `geom_density(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `fill`: internal colour
- `weight`: observation weight used in statistical transformation
- `colour`: border colour
- `alpha`: transparency
- `size`: size
- `linetype`: line type

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_histogram: for the histogram
- http://had.co.nz/ggplot2/geom_density.html

Examples

```
## Not run:
# See stat_density for examples

## End(Not run)
```

---

geom_density2d *geom_density2d*

Description

Contours from a 2d density estimate

Usage

```
geom_density2d(mapping = NULL, data = NULL, stat = "density2d",
    position = "identity", lineend = "butt", linejoin = "round",
    linemitre = 1, na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
lineend	Line end style (round, butt, square)
linejoin	Line join style (round, mitre, bevel)
linemitre	Line mitre limit (number greater than 1)
na.rm	NULL
...	other arguments

Details

Perform a 2D kernel density estimatation using kde2d and display the results with contours.

This page describes geom_density2d, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_density2d. Aesthetics are mapped to variables in the data with the aes function: `geom_density2d(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `weight`: observation weight used in statistical transformation
- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Advice

This can be useful for dealing with overplotting.

Author(s)

Hadley Wickham, `http://had.co.nz/`

See Also

- geom_contour: contour drawing geom
- stat_sum: another way of dealing with overplotting
- http://had.co.nz/ggplot2/geom_density2d.html

Examples

```
## Not run:
# See stat_density2d for examples

## End(Not run)
```

---

geom_errorbar *geom_errorbar*

Description

Error bars

Usage

```
geom_errorbar(mapping = NULL, data = NULL, stat = "identity",
    position = "identity", ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes geom_errorbar, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_errorbar. Aesthetics are mapped to variables in the data with the aes function: `geom_errorbar(aes(x = var))`

- `x`: x position (**required**)
- `ymin`: bottom (vertical minimum) (**required**)
- `ymax`: top (vertical maximum) (**required**)
- `colour`: border colour
- `size`: size
- `linetype`: line type
- `width`: width
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `geom_pointrange`: range indicated by straight line, with point in the middle
- `geom_linerange`: range indicated by straight line
- `geom_crossbar`: hollow bar with middle indicated by horizontal line
- `stat_summary`: examples of these guys in use
- `geom_smooth`: for continuous analog
- http://had.co.nz/ggplot2/geom_errorbar.html

Examples

```
## Not run:
# Create a simple example dataset
df <- data.frame(
  trt = factor(c(1, 1, 2, 2)),
  resp = c(1, 5, 3, 4),
  group = factor(c(1, 2, 1, 2)),
  se = c(0.1, 0.3, 0.3, 0.2)
)
df2 <- df[c(1,3),]

# Define the top and bottom of the errorbars
limits <- aes(ymax = resp + se, ymin=resp - se)

p <- ggplot(df, aes(fill=group, y=resp, x=trt))
p + geom_bar(position="dodge", stat="identity")

# Because the bars and errorbars have different widths
# we need to specify how wide the objects we are dodging are
dodge <- position_dodge(width=0.9)
```

```
p + geom_bar(position=dodge) + geom_errorbar(limits, position=dodge, width=0.25)

p <- ggplot(df2, aes(fill=group, y=resp, x=trt))
p + geom_bar(position=dodge)
p + geom_bar(position=dodge) + geom_errorbar(limits, position=dodge, width=0.25)

p <- ggplot(df, aes(colour=group, y=resp, x=trt))
p + geom_point() + geom_errorbar(limits, width=0.2)
p + geom_pointrange(limits)
p + geom_crossbar(limits, width=0.2)

# If we want to draw lines, we need to manually set the
# groups which define the lines - here the groups in the
# original dataframe
p + geom_line(aes(group=group)) + geom_errorbar(limits, width=0.2)

## End(Not run)
```

`geom_errorbarh` *geom_errorbarh*

Description

Horizontal error bars

Usage

```
geom_errorbarh(mapping = NULL, data = NULL, stat = "identity",
    position = "identity", ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

This page describes geom_errorbarh, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_errorbarh. Aesthetics are mapped to variables in the data with the aes function: `geom_errorbarh(aes(x = var))`

- `x`: x position (**required**)
- `xmin`: left (hortizontal minimum) (**required**)
- `xmax`: right (hortizontal maximum) (**required**)
- `colour`: border colour
- `size`: size
- `linetype`: line type
- `height`: height
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `geom_errorbar`: vertical error bars
- http://had.co.nz/ggplot2/geom_errorbarh.html

Examples

```
## Not run:
df <- data.frame(
  trt = factor(c(1, 1, 2, 2)),
  resp = c(1, 5, 3, 4),
  group = factor(c(1, 2, 1, 2)),
  se = c(0.1, 0.3, 0.3, 0.2)
)

# Define the top and bottom of the errorbars

p <- ggplot(df, aes(resp, trt, colour = group))
p + geom_point() +
  geom_errorbarh(aes(xmax = resp + se, xmin = resp - se))


## End(Not run)
```

geom_freqpoly *geom_freqpoly*

Description

Frequency polygon

Usage

```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes geom_freqpoly, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_freqpoly. Aesthetics are mapped to variables in the data with the aes function: `geom_freqpoly(aes(x = var))`

- colour: border colour
- size: size
- linetype: line type
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_histogram: Histogram
- http://had.co.nz/ggplot2/geom_freqpoly.html

Examples

```
## Not run:
qplot(carat, data = diamonds, geom="freqpoly")
qplot(carat, data = diamonds, geom="freqpoly", binwidth = 0.1)
qplot(carat, data = diamonds, geom="freqpoly", binwidth = 0.01)

qplot(price, data = diamonds, geom="freqpoly", binwidth = 1000)
qplot(price, data = diamonds, geom="freqpoly", binwidth = 1000,
  colour = color)
qplot(price, ..density.., data = diamonds, geom="freqpoly",
  binwidth = 1000, colour = color)


## End(Not run)
```

---

geom_hex *geom_hex*

Description

Tile the plane with hexagons

Usage

```
geom_hex(mapping = NULL, data = NULL, stat = "binhex", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes geom_hex, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_hex. Aesthetics are mapped to variables in the data with the aes function: `geom_hex(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `colour`: border colour
- `fill`: internal colour
- `size`: size
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_hex.html

Examples

```
## Not run:
# See ?stat_binhex for examples

## End(Not run)
```

---

geom_histogram *geom_histogram*

Description

Histogram

Usage

```
geom_histogram(mapping = NULL, data = NULL, stat = "bin", position = "stack",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

geom_histogram is an alias for geom_bar + stat_bin so you will need to look at the documentation for those objects to get more information about the parameters.

This page describes geom_histogram, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_histogram. Aesthetics are mapped to variables in the data with the aes function: `geom_histogram(aes(x = var))`

- `x`: x position (**required**)
- `colour`: border colour
- `fill`: internal colour
- `size`: size
- `linetype`: line type
- `weight`: observation weight used in statistical transformation
- `alpha`: transparency

Advice

geom_histogram only allows you to set the width of the bins (with the binwidth parameter), not the number of bins, and it certainly does not suport the use of common heuristics to select the number of bins. In practice, you will need to use multiple bin widths to discover all the signal in the data, and having bins with meaningful widths (rather than some arbitrary fraction of the range of the data) is more interpretable.

Author(s)

Hadley Wickham, `http://had.co.nz/`

See Also

- `stat_bin`: for more details of the binning alogirithm
- `position_dodge`: for creating side-by-side barcharts
- `position_stack`: for more info on stacking
- `http://had.co.nz/ggplot2/geom_histogram.html`

Examples

```
## Not run:

# Simple examles
qplot(rating, data=movies, geom="histogram")
qplot(rating, data=movies, weight=votes, geom="histogram")
qplot(rating, data=movies, weight=votes, geom="histogram", binwidth=1)
qplot(rating, data=movies, weight=votes, geom="histogram", binwidth=0.1)

# More complex
m <- ggplot(movies, aes(x=rating))
m + geom_histogram()
m + geom_histogram(aes(y = ..density..)) + geom_density()

m + geom_histogram(binwidth = 1)
m + geom_histogram(binwidth = 0.5)
m + geom_histogram(binwidth = 0.1)

# Add aesthetic mappings
m + geom_histogram(aes(weight = votes))
m + geom_histogram(aes(y = ..count..))
m + geom_histogram(aes(fill = ..count..))

# Change scales
m + geom_histogram(aes(fill = ..count..)) +
  scale_fill_gradient("Count", low = "green", high = "red")

# Often we don't want the height of the bar to represent the
# count of observations, but the sum of some other variable.
# For example, the following plot shows the number of movies
# in each rating.
qplot(rating, data=movies, geom="bar", binwidth = 0.1)
# If, however, we want to see the number of votes cast in each
# category, we need to weight by the votes variable
qplot(rating, data=movies, geom="bar", binwidth = 0.1,
  weight=votes, ylab = "votes")

m <- ggplot(movies, aes(x = votes))
# For transformed scales, binwidth applies to the transformed data.
# The bins have constant width on the transformed scale.
m + geom_histogram() + scale_x_log10()
m + geom_histogram(binwidth = 1) + scale_x_log10()
m + geom_histogram() + scale_x_sqrt()
m + geom_histogram(binwidth = 10) + scale_x_sqrt()

# For transformed coordinate systems, the binwidth applies to the
# raw data.  The bins have constant width on the original scale.

# Using log scales does not work here, because the first
# bar is anchored at zero, and so when transformed becomes negative
# infinity.  This is not a problem when transforming the scales, because
# no observations have 0 ratings.
```

```
should_stop(m + geom_histogram() + coord_trans(x = "log10"))
m + geom_histogram() + coord_trans(x = "sqrt")
m + geom_histogram(binwidth=1000) + coord_trans(x = "sqrt")

# You can also transform the y axis.  Remember that the base of the bars
# has value 0, so log transformations are not appropriate
m <- ggplot(movies, aes(x = rating))
m + geom_histogram(binwidth = 0.5) + scale_y_sqrt()
m + geom_histogram(binwidth = 0.5) + scale_y_reverse()

# Set aesthetics to fixed value
m + geom_histogram(colour = "darkgreen", fill = "white", binwidth = 0.5)

# Use facets
m <- m + geom_histogram(binwidth = 0.5)
m + facet_grid(Action ~ Comedy)

# Often more useful to use density on the y axis when facetting
m <- m + aes(y = ..density..)
m + facet_grid(Action ~ Comedy)
m + facet_wrap(~ mpaa)

# Multiple histograms on the same graph
# see ?position, ?position_fill, etc for more details.
ggplot(diamonds, aes(x=price)) + geom_bar()
hist_cut <- ggplot(diamonds, aes(x=price, fill=cut))
hist_cut + geom_bar() # defaults to stacking
hist_cut + geom_bar(position="fill")
hist_cut + geom_bar(position="dodge")

# This is easy in ggplot2, but not visually effective.  It's better
# to use a frequency polygon or density plot.  Like this:
ggplot(diamonds, aes(price, ..density.., colour = cut)) +
  geom_freqpoly(binwidth = 1000)
# Or this:
ggplot(diamonds, aes(price, colour = cut)) +
  geom_density()
# Or if you want to be fancy, maybe even this:
ggplot(diamonds, aes(price, fill = cut)) +
  geom_density(alpha = 0.2)
# Which looks better when the distributions are more distinct
ggplot(diamonds, aes(depth, fill = cut)) +
  geom_density(alpha = 0.2) + xlim(55, 70)


## End(Not run)
```

---

geom_hline *geom_hline*

Description

Line, horizontal

Usage

```
geom_hline(mapping = NULL, data = NULL, stat = "hline", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

This geom allows you to annotate the plot with horizontal lines (see geom_vline and geom_abline for other types of lines)

There are two ways to use it. You can either specify the intercept of the line in the call to the geom, in which case the line will be in the same position in every panel. Alternatively, you can supply a different intercept for each panel using a data.frame. See the examples for the differences

This page describes geom_hline, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_hline. Aesthetics are mapped to variables in the data with the aes function: `geom_hline(aes(x = var))`

- colour: border colour
- size: size
- linetype: line type
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_vline: for vertical lines
- geom_abline: for lines defined by a slope and intercept
- geom_segment: for a more general approach
- http://had.co.nz/ggplot2/geom_hline.html

Examples

```
## Not run:
p <- ggplot(mtcars, aes(x = wt, y=mpg)) + geom_point()

p + geom_hline(aes(yintercept=mpg))
p + geom_hline(yintercept=20)
p + geom_hline(yintercept=seq(10, 30, by=5))

# To display different lines in different facets, you need to
# create a data frame.
p <- qplot(mpg, wt, data=mtcars, facets = vs ~ am)

hline.data <- data.frame(z = 1:4, vs = c(0,0,1,1), am = c(0,1,0,1))
p + geom_hline(aes(yintercept = z), hline.data)

## End(Not run)
```

---

geom_jitter *geom_jitter*

Description

Points, jittered to reduce overplotting

Usage

```
geom_jitter(mapping = NULL, data = NULL, stat = "identity", position = "jitter",
    na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
na.rm	NULL
...	ignored

Details

The jitter geom is a convenient default for geom_point with position = 'jitter'. See position_jitter for more details on adjusting the amount of jittering.

This page describes geom_jitter, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_jitter. Aesthetics are mapped to variables in the data with the aes function: `geom_jitter(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `shape`: shape of point
- `colour`: border colour
- `size`: size
- `fill`: internal colour
- `alpha`: transparency

Advice

It is often useful for plotting categorical data.

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `geom_point`: Regular, unjittered points
- `geom_boxplot`: Another way of looking at the conditional distribution of a variable
- `position_jitter`: For examples, using jittering with other geoms
- http://had.co.nz/ggplot2/geom_jitter.html

Examples

```
## Not run:
p <- ggplot(movies, aes(x=mpaa, y=rating))
p + geom_point()
p + geom_point(position = "jitter")

# Add aesthetic mappings
p + geom_jitter(aes(colour=rating))
```

```
# Vary parameters
p + geom_jitter(position=position_jitter(width=5))
p + geom_jitter(position=position_jitter(height=5))

# Use qplot instead
qplot(mpaa, rating, data=movies, geom="jitter")
qplot(mpaa, rating, data=movies, geom=c("boxplot","jitter"))
qplot(mpaa, rating, data=movies, geom=c("jitter", "boxplot"))

## End(Not run)
```

`geom_line` *geom_line*

Description

Connect observations, in ordered by x value

Usage

```
geom_line(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	other arguments

Details

This page describes geom_line, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_line. Aesthetics are mapped to variables in the data with the aes function: `geom_line(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `geom_path`: Connect observations, in original order
- `geom_segment`: Line segments
- `geom_ribbon`: Fill between line and x-axis
- http://had.co.nz/ggplot2/geom_line.html

Examples

```
## Not run:
# Summarise number of movie ratings by year of movie
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(nu
}))

p <- ggplot(mry, aes(x=year, y=number, group=rating))
p + geom_line()

# Add aesthetic mappings
p + geom_line(aes(size = rating))
p + geom_line(aes(colour = rating))

# Change scale
p + geom_line(aes(colour = rating)) + scale_colour_gradient(low="red")
p + geom_line(aes(size = rating)) + scale_size(to = c(0.1, 3))

# Set aesthetics to fixed value
p + geom_line(colour = "red", size = 1)

# Use qplot instead
qplot(year, number, data=mry, group=rating, geom="line")
```

```
# Using a time series
qplot(date, pop, data=economics, geom="line")
qplot(date, pop, data=economics, geom="line", log="y")
qplot(date, pop, data=subset(economics, date > as.Date("2006-1-1")), geom="line")
qplot(date, pop, data=economics, size=unemploy/pop, geom="line")

# See scale_date for examples of plotting multiple times series on
# a single graph

# A simple pcp example

y2005 <- runif(300, 20, 120)
y2010 <- y2005 * runif(300, -1.05, 1.5)
group <- rep(LETTERS[1:3], each = 100)

df <- data.frame(id = seq_along(group), group, y2005, y2010)
dfm <- reshape::melt(df, id.var = c("id", "group"))
ggplot(dfm, aes(variable, value, group = id, colour = group)) +
  geom_path(alpha = 0.5)

## End(Not run)
```

`geom_linerange` *geom_linerange*

Description

An interval represented by a vertical line

Usage

```
geom_linerange(mapping = NULL, data = NULL, stat = "identity",
    position = "identity", ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

This page describes geom_linerange, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_linerange. Aesthetics are mapped to variables in the data with the aes function: `geom_linerange(aes(x = var))`

- `x`: x position (**required**)
- `ymin`: bottom (vertical minimum) (**required**)
- `ymax`: top (vertical maximum) (**required**)
- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_errorbar: error bars
- geom_pointrange: range indicated by straight line, with point in the middle
- geom_crossbar: hollow bar with middle indicated by horizontal line
- stat_summary: examples of these guys in use
- geom_smooth: for continuous analog
- http://had.co.nz/ggplot2/geom_linerange.html

Examples

```
## Not run:
# Generate data: means and standard errors of means for prices
# for each type of cut
dmod <- lm(price ~ cut, data=diamonds)
cuts <- data.frame(cut=unique(diamonds$cut), predict(dmod, data.frame(cut = unique(diamonds$

qplot(cut, fit, data=cuts)
# With a bar chart, we are comparing lengths, so the y-axis is
# automatically extended to include 0
qplot(cut, fit, data=cuts, geom="bar")

# Display estimates and standard errors in various ways
se <- ggplot(cuts, aes(cut, fit,
  ymin = fit - se.fit, ymax=fit + se.fit, colour = cut))
se + geom_linerange()
se + geom_pointrange()
```

```
se + geom_errorbar(width = 0.5)
se + geom_crossbar(width = 0.5)

# Use coord_flip to flip the x and y axes
se + geom_linerange() + coord_flip()

## End(Not run)
```

`geom_path` *geom_path*

Description

Connect observations, in original order

Usage

```
geom_path(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    lineend = "butt", linejoin = "round", linemitre = 1, na.rm = FALSE,
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`lineend`	Line end style (round, butt, square)
`linejoin`	Line join style (round, mitre, bevel)
`linemitre`	Line mitre limit (number greater than 1)
`na.rm`	NULL
`...`	other arguments

Details

This page describes geom_path, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_path. Aesthetics are mapped to variables in the data with the aes function: `geom_path(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `geom_line`: Functional (ordered) lines
- `geom_polygon`: Filled paths (polygons)
- `geom_segment`: Line segments
- http://had.co.nz/ggplot2/geom_path.html

Examples

```
## Not run:
# Generate data
myear <- ddply(movies, .(year), colwise(mean, .(length, rating)))
p <- ggplot(myear, aes(length, rating))
p + geom_path()

# Add aesthetic mappings
p + geom_path(aes(size = year))
p + geom_path(aes(colour = year))

# Change scale
p + geom_path(aes(size = year)) + scale_size(to = c(1, 3))

# Set aesthetics to fixed value
p + geom_path(colour = "green")

# Control line join parameters
df <- data.frame(x = 1:3, y = c(4, 1, 9))
base <- ggplot(df, aes(x, y))
base + geom_path(size = 10)
base + geom_path(size = 10, lineend = "round")
base + geom_path(size = 10, linejoin = "mitre", lineend = "butt")

# Use qplot instead
```

```
qplot(length, rating, data=myear, geom="path")

# Using economic data:
# How is unemployment and personal savings rate related?
qplot(unemploy/pop, psavert, data=economics)
qplot(unemploy/pop, psavert, data=economics, geom="path")
qplot(unemploy/pop, psavert, data=economics, geom="path", size=as.numeric(date))

# How is rate of unemployment and length of unemployment?
qplot(unemploy/pop, uempmed, data=economics)
qplot(unemploy/pop, uempmed, data=economics, geom="path")
qplot(unemploy/pop, uempmed, data=economics, geom="path") +
  geom_point(data=head(economics, 1), colour="red") +
  geom_point(data=tail(economics, 1), colour="blue")
qplot(unemploy/pop, uempmed, data=economics, geom="path") +
  geom_text(data=head(economics, 1), label="1967", colour="blue") +
  geom_text(data=tail(economics, 1), label="2007", colour="blue")

# geom_path removes missing values on the ends of a line.
# use na.rm = T to suppress the warning message
df <- data.frame(
  x = 1:5,
  y1 = c(1, 2, 3, 4, NA),
  y2 = c(NA, 2, 3, 4, 5),
  y3 = c(1, 2, NA, 4, 5),
  y4 = c(1, 2, 3, 4, 5))
qplot(x, y1, data = df, geom = c("point","line"))
qplot(x, y2, data = df, geom = c("point","line"))
qplot(x, y3, data = df, geom = c("point","line"))
qplot(x, y4, data = df, geom = c("point","line"))

# Setting line type vs colour/size
# Line type needs to be applied to a line as a whole, so it can
# not be used with colour or size that vary across a line

x <- seq(0.01, .99, length=100)
df <- data.frame(x = rep(x, 2), y = c(qlogis(x), 2 * qlogis(x)), group = rep(c("a","b"), eac
p <- ggplot(df, aes(x=x, y=y, group=group))

# Should work
p + geom_line(linetype = 2)
p + geom_line(aes(colour = group), linetype = 2)
p + geom_line(aes(colour = x))

# Should fail
should_stop(p + geom_line(aes(colour = x), linetype=2))


## End(Not run)
```

---

geom_point *geom_point*

Description

Points, as for a scatterplot

Usage

```
geom_point(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
na.rm	NULL
...	ignored

Details

The point geom is used to create scatterplots.

This page describes geom_point, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_point. Aesthetics are mapped to variables in the data with the aes function: `geom_point(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- shape: shape of point
- colour: border colour
- size: size
- fill: internal colour
- alpha: transparency

Advice

The scatterplot is useful for displaying the relationship between two continuous variables, although it can also be used with one continuous and one categorical variable, or two categorical variables. See geom_jitter for possibilities.

The *bubblechart* is a scatterplot with a third variable mapped to the size of points. There are no special names for scatterplots where another variable is mapped to point shape or colour, however.

The biggest potential problem with a scatterplot is overplotting: whenever you have more than a few points, points may be plotted on top of one another. This can severely distort the visual appearance of the plot. There is no one solution to this problem, but there are some techniques that can help. You can add additional information with stat_smooth, stat_quantile or stat_density2d. If you have few unique x values, geom_boxplot may also be useful. Alternatively, you can summarise the number of points at each location and display that in some way, using stat_sum. Another technique is to use transparent points, `geom_point(colour=alpha('black', 0.05))`

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_size: To see how to scale area of points, instead of radius
- geom_jitter: Jittered points for categorical data
- http://had.co.nz/ggplot2/geom_point.html

Examples

```
## Not run:
p <- ggplot(mtcars, aes(wt, mpg))
p + geom_point()

# Add aesthetic mappings
p + geom_point(aes(colour = qsec))
p + geom_point(aes(alpha = qsec))
p + geom_point(aes(colour = factor(cyl)))
p + geom_point(aes(shape = factor(cyl)))
p + geom_point(aes(size = qsec))

# Change scales
p + geom_point(aes(colour = cyl)) + scale_colour_gradient(low = "blue")
p + geom_point(aes(size = qsec)) + scale_area()
p + geom_point(aes(shape = factor(cyl))) + scale_shape(solid = FALSE)

# Set aesthetics to fixed value
p + geom_point(colour = "red", size = 3)
qplot(wt, mpg, data = mtcars, colour = I("red"), size = I(3))

# Varying alpha is useful for large datasets
d <- ggplot(diamonds, aes(carat, price))
d + geom_point(alpha = 1/10)
```

```
d + geom_point(alpha = 1/20)
d + geom_point(alpha = 1/100)

# You can create interesting shapes by layering multiple points of
# different sizes
p <- ggplot(mtcars, aes(mpg, wt))
p + geom_point(colour="grey50", size = 4) + geom_point(aes(colour = cyl))
p + aes(shape = factor(cyl)) +
  geom_point(aes(colour = factor(cyl)), size = 4) +
  geom_point(colour="grey90", size = 1.5)
p + geom_point(colour="black", size = 4.5) +
  geom_point(colour="pink", size = 4) +
  geom_point(aes(shape = factor(cyl)))

# These extra layers don't usually appear in the legend, but we can
# force their inclusion
p + geom_point(colour="black", size = 4.5, legend = TRUE) +
  geom_point(colour="pink", size = 4, legend = TRUE) +
  geom_point(aes(shape = factor(cyl)))

# Transparent points:
qplot(mpg, wt, data = mtcars, size = I(5), alpha = I(0.2))

# geom_point warns when missing values have been dropped from the data set
# and not plotted, you can turn this off by setting na.rm = TRUE
mtcars2 <- transform(mtcars, mpg = ifelse(runif(32) < 0.2, NA, mpg))
qplot(wt, mpg, data = mtcars2)
qplot(wt, mpg, data = mtcars2, na.rm = TRUE)

# Use qplot instead
qplot(wt, mpg, data = mtcars)
qplot(wt, mpg, data = mtcars, colour = factor(cyl))
qplot(wt, mpg, data = mtcars, colour = I("red"))

## End(Not run)
```

`geom_pointrange` *geom_pointrange*

Description

An interval represented by a vertical line, with a point in the middle

Usage

```
geom_pointrange(mapping = NULL, data = NULL, stat = "identity",
    position = "identity", ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes geom_pointrange, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_pointrange. Aesthetics are mapped to variables in the data with the aes function: `geom_pointrange(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- ymin: bottom (vertical minimum) (**required**)
- ymax: top (vertical maximum) (**required**)
- colour: border colour
- size: size
- linetype: line type
- shape: shape of point
- fill: internal colour
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_errorbar: error bars
- geom_linerange: range indicated by straight line, + examples
- geom_crossbar: hollow bar with middle indicated by horizontal line
- stat_summary: examples of these guys in use
- geom_smooth: for continuous analog
- http://had.co.nz/ggplot2/geom_pointrange.html

Examples

```
## Not run:
# See geom_linerange for examples

## End(Not run)
```

`geom_polygon` *geom_polygon*

Description

Polygon, a filled path

Usage

```
geom_polygon(mapping = NULL, data = NULL, stat = "identity",
    position = "identity", ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

This page describes geom_polygon, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_polygon. Aesthetics are mapped to variables in the data with the aes function: `geom_polygon(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `colour`: border colour
- `fill`: internal colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_path: an unfilled polygon
- geom_ribbon: a polygon anchored on the x-axis
- http://had.co.nz/ggplot2/geom_polygon.html

Examples

```
## Not run:
# When using geom_polygon, you will typically need two data frames:
# one contains the coordinates of each polygon (positions),  and the
# other the values associated with each polygon (values).  An id
# variable links the two together

ids <- factor(c("1.1", "2.1", "1.2", "2.2", "1.3", "2.3"))

values <- data.frame(
  id = ids,
  value = c(3, 3.1, 3.1, 3.2, 3.15, 3.5)
)

positions <- data.frame(
  id = rep(ids, each = 4),
  x = c(2, 1, 1.1, 2.2, 1, 0, 0.3, 1.1, 2.2, 1.1, 1.2, 2.5, 1.1, 0.3,
  0.5, 1.2, 2.5, 1.2, 1.3, 2.7, 1.2, 0.5, 0.6, 1.3),
  y = c(-0.5, 0, 1, 0.5, 0, 0.5, 1.5, 1, 0.5, 1, 2.1, 1.7, 1, 1.5,
  2.2, 2.1, 1.7, 2.1, 3.2, 2.8, 2.1, 2.2, 3.3, 3.2)
)

# Currently we need to manually merge the two together
datapoly <- merge(values, positions, by=c("id"))

(p <- ggplot(datapoly, aes(x=x, y=y)) + geom_polygon(aes(fill=value, group=id)))

# Which seems like a lot of work, but then it's easy to add on
# other features in this coordinate system, e.g.:

stream <- data.frame(
  x = cumsum(runif(50, max = 0.1)),
  y = cumsum(runif(50,max = 0.1))
)

p + geom_line(data = stream, colour="grey30", size = 5)

# And if the positions are in longitude and latitude, you can use
# coord_map to produce different map projections.

## End(Not run)
```

geom_quantile *geom_quantile*

Description

Add quantile lines from a quantile regression

Usage

```
geom_quantile(mapping = NULL, data = NULL, stat = "quantile",
    position = "identity", lineend = "butt", linejoin = "round",
    linemitre = 1, na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
lineend	Line end style (round, butt, square)
linejoin	Line join style (round, mitre, bevel)
linemitre	Line mitre limit (number greater than 1)
na.rm	NULL
...	other arguments

Details

This page describes geom_quantile, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_quantile. Aesthetics are mapped to variables in the data with the aes function: `geom_quantile(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- weight: observation weight used in statistical transformation
- colour: border colour
- size: size
- linetype: line type
- alpha: transparency

Advice

This can be used as a continuous analogue of a geom_boxplot.

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_line: Functional (ordered) lines
- geom_polygon: Filled paths (polygons)
- geom_segment: Line segments
- http://had.co.nz/ggplot2/geom_quantile.html

Examples

```
## Not run:
# See stat_quantile for examples

## End(Not run)
```

`geom_rect` *geom_rect*

Description

2d rectangles

Usage

```
geom_rect(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

This page describes geom_rect, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_rect. Aesthetics are mapped to variables in the data with the aes function: `geom_rect(aes(x = var))`

- `xmin`: left (hortizontal minimum) (**required**)
- `xmax`: right (hortizontal maximum) (**required**)
- `ymin`: bottom (vertical minimum) (**required**)
- `ymax`: top (vertical maximum) (**required**)
- `colour`: border colour
- `fill`: internal colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_rect.html

Examples

```
## Not run:
df <- data.frame(
  x = sample(10, 20, replace = TRUE),
  y = sample(10, 20, replace = TRUE)
)
ggplot(df, aes(xmin = x, xmax = x + 1, ymin = y, ymax = y + 2)) +
geom_rect()

## End(Not run)
```

geom_ribbon *geom_ribbon*

Description

Ribbons, y range with continuous x values

Usage

```
geom_ribbon(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
na.rm	NULL
...	ignored

Details

This page describes geom_ribbon, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_ribbon. Aesthetics are mapped to variables in the data with the aes function: `geom_ribbon(aes(x = var))`

- x: x position (**required**)
- ymin: bottom (vertical minimum) (**required**)
- ymax: top (vertical maximum) (**required**)
- colour: border colour
- fill: internal colour
- size: size
- linetype: line type
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- geom_bar: Discrete intervals (bars)
- geom_linerange: Discrete intervals (lines)
- geom_polygon: General polygons
- http://had.co.nz/ggplot2/geom_ribbon.html

Examples

```
## Not run:
# Generate data
huron <- data.frame(year = 1875:1972, level = as.vector(LakeHuron))
huron$decade <- round_any(huron$year, 10, floor)

h <- ggplot(huron, aes(x=year))

h + geom_ribbon(aes(ymin=0, ymax=level))
h + geom_area(aes(y = level))

# Add aesthetic mappings
h + geom_ribbon(aes(ymin=level-1, ymax=level+1))
h + geom_ribbon(aes(ymin=level-1, ymax=level+1)) + geom_line(aes(y=level))

# Take out some values in the middle for an example of NA handling
huron[huron$year > 1900 & huron$year < 1910, "level"] <- NA
h <- ggplot(huron, aes(x=year))
h + geom_ribbon(aes(ymin=level-1, ymax=level+1)) + geom_line(aes(y=level))

# Another data set, with multiple y's for each x
m <- ggplot(movies, aes(y=votes, x=year))
(m <- m + geom_point())

# The default summary isn't that useful
m + stat_summary(geom="ribbon", fun.ymin="min", fun.ymax="max")
m + stat_summary(geom="ribbon", fun.data="median_hilow")

# Use qplot instead
qplot(year, level, data=huron, geom=c("area", "line"))

## End(Not run)
```

---

geom_rug *geom_rug*

Description

Marginal rug plots

Usage

```
geom_rug(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes geom_rug, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_rug. Aesthetics are mapped to variables in the data with the aes function: `geom_rug(aes(x = var))`

- colour: border colour
- size: size
- linetype: line type
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_rug.html

Examples

```
## Not run:
p <- ggplot(mtcars, aes(x=wt, y=mpg))
p + geom_point()
p + geom_point() + geom_rug()
p + geom_point() + geom_rug(position='jitter')

## End(Not run)
```

`geom_segment` *geom_segment*

Description

Single line segments

Usage

```
geom_segment(mapping = NULL, data = NULL, stat = "identity",
    position = "identity", ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	other arguments

Details

This page describes geom_segment, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_segment. Aesthetics are mapped to variables in the data with the aes function: `geom_segment(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `xend`: NULL (**required**)

- `yend`: NULL (**required**)
- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `geom_path`: Connect observations, in original order
- `geom_line`: Connect observations, in ordered by x value
- http://had.co.nz/ggplot2/geom_segment.html

Examples

```
## Not run:
require("maps")

xlim <- range(seals$long)
ylim <- range(seals$lat)
usamap <- data.frame(map("world", xlim = xlim, ylim = ylim, plot =
FALSE)[c("x","y")])
usamap <- rbind(usamap, NA, data.frame(map('state', xlim = xlim, ylim
= ylim, plot = FALSE)[c("x","y")]))
names(usamap) <- c("long", "lat")

p <- ggplot(seals, aes(x = long, y = lat))
(p <- p + geom_segment(aes(xend = long + delta_long, yend = lat + delta_lat), arrow=arrow(le
p + geom_path(data = usamap) + scale_x_continuous(limits=xlim)

# You can also use geom_segment to recreate plot(type = "h") :
counts <- as.data.frame(table(x = rpois(100,5)))
counts$x <- as.numeric(as.character(counts$x))
with(counts, plot(x, Freq, type = "h", lwd = 10))

qplot(x, Freq, data = counts, geom="segment",
  yend = 0, xend = x, size = I(10))



## End(Not run)
```

geom_smooth *geom_smooth*

Description

Add a smoothed condition mean.

Usage

```
geom_smooth(mapping = NULL, data = NULL, stat = "smooth", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes geom_smooth, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_smooth. Aesthetics are mapped to variables in the data with the aes function: `geom_smooth(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- colour: border colour
- fill: internal colour
- size: size
- linetype: line type
- weight: observation weight used in statistical transformation
- alpha: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_smooth.html

Examples

```
## Not run:
# See stat_smooth for examples of using built in model fitting
# if you need some more flexible, this example shows you how to
# plot the fits from any model of your choosing
qplot(wt, mpg, data=mtcars, colour=factor(cyl))

model <- lm(mpg ~ wt + factor(cyl), data=mtcars)
grid <- with(mtcars, expand.grid(
  wt = seq(min(wt), max(wt), length = 20),
  cyl = levels(factor(cyl))
))

grid$mpg <- stats::predict(model, newdata=grid)

qplot(wt, mpg, data=mtcars, colour=factor(cyl)) + geom_line(data=grid)

# or with standard errors

err <- stats::predict(model, newdata=grid, se = TRUE)
grid$ucl <- err$fit + 1.96 * err$se.fit
grid$lcl <- err$fit - 1.96 * err$se.fit

qplot(wt, mpg, data=mtcars, colour=factor(cyl)) +
  geom_smooth(aes(ymin = lcl, ymax = ucl), data=grid, stat="identity")

## End(Not run)
```

---

geom_step *geom_step*

Description

Connect observations by stairs

Usage

```
geom_step(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    direction = "hv", ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
direction	direction of stairs: 'vh' for vertical then horizontal, or 'hv' for horizontal then vertical
...	other arguments

Details

Equivalent to plot(type='s').This page describes geom_step, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_step. Aesthetics are mapped to variables in the data with the aes function: `geom_step(aes(x = var))`

- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_step.html

Examples

```
## Not run:
# Simple quantiles/ECDF from examples(plot)
x <- sort(rnorm(47))
qplot(seq_along(x), x, geom="step")

# Steps go horizontally, then vertically (default)
qplot(seq_along(x), x, geom="step", direction = "hv")
plot(x, type = "s")
# Steps go vertically, then horizontally
qplot(seq_along(x), x, geom="step", direction = "vh")
```

```
plot(x, type = "S")

# Also works with other aesthetics
df <- data.frame(
  x = sort(rnorm(50)),
  trt = sample(c("a", "b"), 50, rep = T)
)
qplot(seq_along(x), x, data = df, geom="step", colour = trt)


## End(Not run)
```

---

geom_text *geom_text*

Description

Textual annotations

Usage

```
geom_text(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    parse = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
stat	statistic used by this layer
position	position adjustment used by this layer
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath
...	other arguments

Details

This page describes geom_text, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with geom_text. Aesthetics are mapped to variables in the data with the aes function: `geom_text(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `label`: text label (**required**)
- `colour`: border colour
- `size`: size
- `angle`: angle
- `hjust`: horizontal justification, between 0 and 1
- `vjust`: vertical justification, between 0 and 1
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_text.html

Examples

```
## Not run:
p <- ggplot(mtcars, aes(x=wt, y=mpg, label=rownames(mtcars)))

p + geom_text()
p <- p + geom_point()

# Set aesthetics to fixed value
p + geom_text()
p + geom_point() + geom_text(hjust=0, vjust=0)
p + geom_point() + geom_text(angle = 45)

# Add aesthetic mappings
p + geom_text(aes(colour=factor(cyl)))
p + geom_text(aes(colour=factor(cyl))) + scale_colour_discrete(l=40)

p + geom_text(aes(size=wt))
p + geom_text(aes(size=wt)) + scale_size(to=c(3,6))

# You can display expressions by setting parse = TRUE.  The
# details of the display are described in ?plotmath, but note that
# geom_text uses strings, not expressions.
p + geom_text(aes(label = paste(wt, "^(", cyl, ")", sep = "")),
  parse = T)
```

```
# Use qplot instead
qplot(wt, mpg, data = mtcars, label = rownames(mtcars),
   geom=c("point", "text"))
qplot(wt, mpg, data = mtcars, label = rownames(mtcars), size = wt) +
  geom_text(colour = "red")

## End(Not run)
```

`geom_tile` *geom_tile*

Description

Tile plot as densely as possible, assuming that every tile is the same size.

Usage

```
geom_tile(mapping = NULL, data = NULL, stat = "identity", position = "identity",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

Similar to levelplot and image.

This page describes geom_tile, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_tile. Aesthetics are mapped to variables in the data with the aes function: `geom_tile(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `fill`: internal colour

- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/geom_tile.html

Examples

```
## Not run:
# Generate data
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)
 df
}
p <- ggplot(pp(20), aes(x=x,y=y))

p + geom_tile() #pretty useless!

# Add aesthetic mappings
p + geom_tile(aes(fill=z))

# Change scale
p + geom_tile(aes(fill=z)) + scale_fill_gradient(low="green", high="red")

# Use qplot instead
qplot(x, y, data=pp(20), geom="tile", fill=z)
qplot(x, y, data=pp(100), geom="tile", fill=z)

# Missing values
p <- ggplot(pp(20)[sample(20*20, size=200),], aes(x=x,y=y,fill=z))
p + geom_tile()

# Input that works with image
image(t(volcano)[ncol(volcano):1,])
ggplot(melt(volcano), aes(x=X1, y=X2, fill=value)) + geom_tile()

# inspired by the image-density plots of Ken Knoblauch
cars <- ggplot(mtcars, aes(y=factor(cyl), x=mpg))
cars + geom_point()
cars + stat_bin(aes(fill=..count..), geom="tile", binwidth=3, position="identity")
cars + stat_bin(aes(fill=..density..), geom="tile", binwidth=3, position="identity")
```

```
cars + stat_density(aes(fill=..density..), geom="tile", position="identity")
cars + stat_density(aes(fill=..count..), geom="tile", position="identity")

# Another example with with unequal tile sizes
x.cell.boundary <- c(0, 4, 6, 8, 10, 14)
example <- data.frame(
  x = rep(c(2, 5, 7, 9, 12), 2),
  y = factor(rep(c(1,2), each=5)),
  z = rep(1:5, each=2),
  w = rep(diff(x.cell.boundary), 2)
)

qplot(x, y, fill=z, data=example, geom="tile")
qplot(x, y, fill=z, data=example, geom="tile", width=w)
qplot(x, y, fill=factor(z), data=example, geom="tile", width=w)

# You can manually set the colour of the tiles using
# scale_manual
col <- c("darkblue", "blue", "green", "orange", "red")
qplot(x, y, fill=col[z], data=example, geom="tile", width=w, group=1) + scale_fill_identity(

## End(Not run)
```

`geom_vline` *geom_vline*

Description

Line, vertical

Usage

```
geom_vline(mapping = NULL, data = NULL, stat = "vline", position = "identity",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`stat`	statistic used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

This geom allows you to annotate the plot with vertical lines (see geom_hline and geom_abline for other types of lines)

There are two ways to use it. You can either specify the intercept of the line in the call to the geom, in which case the line will be in the same position in every panel. Alternatively, you can supply a different intercept for each panel using a data.frame. See the examples for the differences

This page describes geom_vline, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with geom_vline. Aesthetics are mapped to variables in the data with the aes function: `geom_vline(aes(x = var))`

- `colour`: border colour
- `size`: size
- `linetype`: line type
- `alpha`: transparency

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `geom_hline`: for horizontal lines
- `geom_abline`: for lines defined by a slope and intercept
- `geom_segment`: for a more general approach
- http://had.co.nz/ggplot2/geom_vline.html

Examples

```
## Not run:
# Fixed lines
p <- ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()
p + geom_vline(xintercept = 5)
p + geom_vline(xintercept = 1:5)
p + geom_vline(xintercept = 1:5, colour="green")

last_plot() + coord_equal()
last_plot() + coord_flip()

p2 <- p + aes(colour = factor(cyl))
p2 + geom_vline(xintercept = 15)
```

```
## End(Not run)
```

`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`	round cells to at most this value

Details

A fluctutation diagram is a graphical representation of a contingency table. This fuction currently only supports 2D contingency tabless 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))

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
ggfluctuation(table(movies$Action, movies$Comedy))
ggfluctuation(table(movies$Action, movies$mpaa))
ggfluctuation(table(movies$Action, movies$Comedy), type="colour")
ggfluctuation(table(warpbreaks$breaks, warpbreaks$tension))
```

ggmissing *Missing values plot*

Description

Create a plot to illustrate patterns of missing values

Usage

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

Arguments

data	data.frame
avoid	whether missings should be stacked or dodged, see geom_bar 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) where this plot was copied from.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

ggstructure, ggorder

Examples

```
mmissing <- movies
mmissing[sample(nrow(movies), 1000), sample(ncol(movies), 5)] <- NA
ggmissing(mmissing)
ggmissing(mmissing, order=FALSE, missing.only = FALSE)
ggmissing(mmissing, avoid="dodge") + scale_y_sqrt()
```

ggorder *Order plot*

Description

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

Usage

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

Arguments

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

Author(s)

Hadley Wickham <h.wickham@gmail.com>

ggpcp *Parallel coordinates plot.*

Description

Generate a plot "template" for a parallel coordinates plot.

Usage

```
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 geom 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

```
ggpcp(mtcars) + geom_line()
ggpcp(mtcars, scale="var") + geom_line()
ggpcp(mtcars, vars=names(mtcars)[3:6], formula= . ~cyl, scale="I") + geom_line()
ggpcp(mtcars, scale="I") + geom_boxplot(aes(group=variable))
ggpcp(mtcars, vars=names(mtcars[2:6])) + geom_line()
p <- ggpcp(mtcars, vars=names(mtcars[2:6]))
p + geom_line()
p + geom_line(aes(colour=mpg))
```

---

ggplot *Create a new plot*

Description

Create a new ggplot plot

Usage

```
ggplot(data = NULL, ...)
```

Arguments

data	default data set
...	other arguments passed to specific methods

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

http://had.co.nz/ggplot2

ggplot.data.frame *Create a new plot*

Description

Create a new ggplot plot

Usage

```
ggplot.data.frame(data, mapping=aes(), ..., environment = globalenv())
```

Arguments

data	default data frame
mapping	default list of aesthetic mappings (these can be colour, size, shape, line type – see individual geom functions for more details)
...	ignored
environment	environment in which evaluation of aesthetics should occur

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

http://had.co.nz/ggplot2

ggsave *ggsave*

Description

Save a ggplot with sensible defaults

Usage

```
ggsave(filename=default_name(plot), plot = last_plot(), device=default_device(filer
```

Arguments

filename	file name/filename of plot
plot	plot to save, defaults to last plot displayed
device	device to use, automatically extract from file name extension
path	path to save plot to (if you just want to set path and not filename)
scale	scaling factor
width	width (in inches)
height	height (in inches)
dpi	dpi to use for raster graphics
keep	plot components to keep
drop	plot components to drop
...	other arguments passed to graphics device

Details

ggsave is a convenient function for saving a plot. It defaults to saving the last plot that you displayed, and for a default size uses the size of the current graphics device. It also guesses the type of graphics device from the extension. This means the only argument you need to supply is the filename.

ggsave currently recognises the extensions eps/ps, tex (pictex), pdf, jpeg, tiff, png, bmp, svg and wmf (windows only).

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
## Not run:
ratings <- qplot(rating, data=movies, geom="histogram")
qplot(length, data=movies, geom="histogram")
ggsave(file="length-hist.pdf")
ggsave(file="length-hist.png")
ggsave(ratings, file="ratings.pdf")
ggsave(ratings, file="ratings.pdf", width=4, height=4)
# make twice as big as on screen
ggsave(ratings, file="ratings.pdf", scale=2)

## End(Not run)
```

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

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
ggstructure(mtcars)
```

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

Description

The internet movie database, http://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, http://imdb.com/help/show_leaf?about, including information about the data collection process, http://imdb.com/help/show_leaf?infosource.

IMDB makes their raw data available at http://uk.imdb.com/interfaces/. 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 relevent information and store 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/

---

label_both *Label facets with value and variable*

Description

Join together facet value and the name of the variable to create a label.

Usage

```
label_both(variable, value)
```

Arguments

variable	variable name passed in by facetter
value	variable value passed in by facetter

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
p <- qplot(wt, mpg, data = mtcars)
p + facet_grid(~ cyl)
p + facet_grid(~ cyl, labeller = label_both)
```

---

label_bquote *Label facet with 'bquoted' expressions*

Description

Create facet labels which contain the facet label in a larger expression

Usage

```
label_bquote(expr = beta ^ .(x))
```

Arguments

expr	expression to use

Details

See bquote for details on the syntax of the argument. The label value is x.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

plotmath

Examples

```
p <- qplot(wt, mpg, data = mtcars)
p + facet_grid(~ vs, labeller = label_bquote(alpha ^ .(x)))
p + facet_grid(~ vs, labeller = label_bquote(.(x) ^ .(x)))
```

label_parsed *Label facets with parsed label.*

Description

Parses the facet label, as if

Usage

```
label_parsed(variable, value)
```

Arguments

variable	variable name passed in by facetter
value	variable value passed in by facetter

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

plotmath

Examples

```
mtcars$cyl2 <- factor(mtcars$cyl, labels = c("alpha", "beta", "gamma"))
qplot(wt, mpg, data = mtcars) + facet_grid(. ~ cyl2)
qplot(wt, mpg, data = mtcars) + facet_grid(. ~ cyl2,
labeller = label_parsed)
```

label_value *Label facets with their value*

Description

The default facet labelling just uses the value of the variable

Usage

```
label_value(variable, value)
```

Arguments

variable	variable name passed in by facetter
value	variable value passed in by facetter

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
p <- qplot(wt, mpg, data = mtcars)
p + facet_grid(~ cyl)
p + facet_grid(~ cyl, labeller = label_value)
```

---

labs *Change axis labels and legend titles*

Description

This is a convenience function that saves some typing when modifying the axis labels or legend titles

Usage

```
labs(...)
```

Arguments

...	a list of new names in the form aesthetic = "new name"

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
p <- qplot(mpg, wt, data = mtcars)
p + labs(x = "New x label")
p + xlab("New x label")
p + ylab("New y label")

# This should work indepdendently of other functions that modify the
# the scale names
p + ylab("New y label") + ylim(2, 4)
p + ylim(2, 4) + ylab("New y label")

# The labs function also modifies legend labels
p <- qplot(mpg, wt, data = mtcars, colour = cyl)
p + labs(colour = "Cylinders")

# Can also pass in a list, if that is more convenient
p + labs(list(x = "X", y = "Y"))
```

last_plot *Retrieve last plot modified/created.*

Description

Whenever a plot is created or modified, it is recorded.

Usage

```
last_plot()
```

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

ggsave

Mammals sleep *An updated and expanded version of the mammals sleep dataset*

Description

This is an updated and expanded version of the mammals sleep dataset. Updated sleep times and weights were taken from V. M. Savage and G. B. West. A quantitative, theoretical framework for understanding mammalian sleep. Proceedings of the National Academy of Sciences, 104 (3):1051-1056, 2007. Additional variables order, conservation status and vore were added from wikipedia data.

- name. common name
- genus.
- vore. carnivore, omnivore or herbivore?
- order.
- conservation. the conservation status of the animal
- sleep_total. total amount of sleep, in hours
- sleep_rem. rem sleep, in hours
- sleep_cycle. length of sleep cycle, in hours
- awake. amount of time spent awake, in hours
- brainwt. brain weight in kilograms
- bodywt. body weight in kilograms

Usage

```
data(msleep)
```

Format

A data frame with 83 rows and 11 variables

---

map_data *Map data*

Description

Convert map to data frame

Usage

```
map_data(map, region = ".")
```

Arguments

map	map name
region	region name

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
if (require(maps)) {
states <- map_data("state")
arrests <- USArrests
names(arrests) <- tolower(names(arrests))
arrests$region <- tolower(rownames(USArrests))

choro <- merge(states, arrests, sort = FALSE, by = "region")
choro <- choro[order(choro$order), ]
qplot(long, lat, data = choro, group = group, fill = assault,
geom="polygon")
qplot(long, lat, data = choro, group = group, fill = assault / murder,
geom="polygon")
}
```

mean_se *Mean + se's.*

Description

Mean and standard errors on either side.

Usage

```
mean_se(x, mult = 1)
```

Arguments

x	numeric vector
mult	number of multiples of standard error

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

for use with stat_summary

Midwest demographics *Demographic information of midwest counties*

Description

The variables are as follows:

- PID
- county
- state
- area
- poptotal. Total population
- popdensity. Population density
- popwhite. Number of whites.
- popblack. Number of blacks.
- popamerindian. Number of American Indians.
- popasian. Number of Asians.

- popother. Number of other races.
- percwhite. Percent white.
- percblack. Percent black.
- percamerindan. Percent American Indian.
- percasian. Percent Asian.
- percother. Percent other races.
- popadults. Number of adults.
- perchsd.
- percollege. Percent college educated.
- percprof. Percent profession.
- poppovertyknown.
- percpovertyknown
- percbelowpoverty
- percchildbelowpovert
- percadultpoverty
- percelderlypoverty
- inmetro. In a metro area.
- category'

Usage

```
data(midwest)
```

Format

A data frame with 437 rows and 28 variables

Nodoc *See website for documentation*

Description

All documentation is available at http://had.co.nz/ggplot2. Future versions will include more documentation in the package itself.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

opts *Plot options*

Description

Set options/theme elements for a single plot

Usage

```
opts(...)
```

Arguments

...	named list of theme settings

Details

Use this function if you want to modify a few theme settings for a single plot.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
p <- qplot(mpg, wt, data = mtcars)
p
p + opts(panel_background = theme_rect(colour = "pink"))
p + theme_bw()
```

percent *Percent formatter*

Description

Multiply by one hundred and display percent sign

Usage

```
percent(x)
```

Arguments

x	numeric vector to format

Author(s)

Hadley Wickham <h.wickham@gmail.com>

plotmatrix *Code to create a scatterplot matrix (experimental)*

Description

Crude experimental scatterplot matrix

Usage

```
plotmatrix(data, mapping=aes(), colour="black")
```

Arguments

data	data frame
mapping	any additional aesthetic mappings (do not use x and y)
colour	default point colour

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
plotmatrix(mtcars[, 1:3])
plotmatrix(mtcars[, 1:3]) + geom_smooth(method="lm")
```

position_dodge *position_dodge*

Description

Adjust position by dodging overlaps to the side

Usage

```
position_dodge(width = NULL, height = NULL, ...)
```

Arguments

width	NULL
height	NULL
...	ignored

Details

This page describes position_dodge, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/position_dodge.html

Examples

```
## Not run:
ggplot(mtcars, aes(x=factor(cyl), fill=factor(vs))) +
  geom_bar(position="dodge")
ggplot(diamonds, aes(x=price, fill=cut)) + geom_bar(position="dodge")
# see ?geom_boxplot and ?geom_bar for more examples

# Dodging things with different widths is tricky
df <- data.frame(x=c("a","a","b","b"), y=1:4)
(p <- qplot(x, y, data=df, position="dodge", geom="bar", stat="identity"))

p + geom_linerange(aes(ymin = y-1, ymax = y+1), position="dodge")
# You need to explicitly specify the width for dodging
p + geom_linerange(aes(ymin = y-1, ymax = y+1),
  position = position_dodge(width = 0.9))

# Similarly with error bars:
p + geom_errorbar(aes(ymin = y-1, ymax = y+1), width = 0.2,
  position="dodge")
p + geom_errorbar(aes(ymin = y-1, ymax = y+1, width = 0.2),
  position = position_dodge(width = 0.90))

## End(Not run)
```

---

position_fill *position_fill*

Description

Stack overlapping objects on top of one another, and standardise have equal height

Usage

```
position_fill(width = NULL, height = NULL, ...)
```

Arguments

width	NULL
height	NULL
...	ignored

Details

This page describes position_fill, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/position_fill.html

Examples

```
## Not run:
# See ?geom_bar and ?geom_area for more examples
ggplot(mtcars, aes(x=factor(cyl), fill=factor(vs))) + geom_bar(position="fill")

cde <- geom_histogram(position="fill", binwidth = 500)

ggplot(diamonds, aes(x=price)) + cde
ggplot(diamonds, aes(x=price, fill=cut)) + cde
ggplot(diamonds, aes(x=price, fill=clarity)) + cde
ggplot(diamonds, aes(x=price, fill=color)) + cde

## End(Not run)
```

position_identity *position_identity*

Description

Don't adjust position

Usage

```
position_identity(width = NULL, height = NULL, ...)
```

Arguments

width	NULL
height	NULL
...	ignored

Details

This page describes position_identity, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/position_identity.html

Examples

```
## Not run:
  # Coming soon

## End(Not run)
```

position_jitter *position_jitter*

Description

Jitter points to avoid overplotting

Usage

```
position_jitter(width = NULL, height = NULL, ...)
```

Arguments

width	degree of jitter in x direction. Defaults to 40% of the resolution of the data.
height	degree of jitter in y direction. Defaults to 40% of the resolution of the data.
...	other arguments

Details

This page describes position_jitter, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/position_jitter.html

Examples

```
## Not run:
qplot(am, vs, data=mtcars)

# Default amount of jittering will generally be too much for
# small datasets:
qplot(am, vs, data=mtcars, position="jitter")
# Control the amount as follows
qplot(am, vs, data=mtcars, position=position_jitter(w=0.1, h=0.1))

# The default works better for large datasets, where it will
# will up as much space as a boxplot or a bar
qplot(cut, price, data=diamonds, geom=c("boxplot", "jitter"))

## End(Not run)
```

position_stack *position_stack*

Description

Stack overlapping objects on top of one another

Usage

```
position_stack(width = NULL, height = NULL, ...)
```

Arguments

width	NULL
height	NULL
...	ignored

Details

This page describes position_stack, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/position_stack.html

Examples

```
## Not run:
# Stacking is the default behaviour for most area plots:
ggplot(mtcars, aes(factor(cyl), fill = factor(vs))) + geom_bar()

ggplot(diamonds, aes(price)) + geom_histogram(binwidth=500)
ggplot(diamonds, aes(price, fill = cut)) + geom_histogram(binwidth=500)

# Stacking is also useful for time series
data.set <- data.frame(
  Time = c(rep(1, 4),rep(2, 4), rep(3, 4), rep(4, 4)),
  Type = rep(c('a', 'b', 'c', 'd'), 4),
  Value = rpois(16, 10)
```

```
)

qplot(Time, Value, data = data.set, fill = Type, geom = "area")
# If you want to stack lines, you need to say so:
qplot(Time, Value, data = data.set, colour = Type, geom = "line")
qplot(Time, Value, data = data.set, colour = Type, geom = "line",
  position = "stack")
# But realise that this makes it *much* harder to compare individual
# trends

## End(Not run)
```

Presidential terms *Terms of 10 presidents from Eisenhower to Bush W.*

Description

The names of each president, the start and end date of their term, and their party of 10 US presidents from Eisenhower to Bush W.

Usage

```
data(presidential)
```

Format

A data frame with 10 rows and 4 variables

---

qplot *Quick plot.*

Description

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

Usage

```
qplot(x, y = NULL, z=NULL, ..., data, facets = . ~ ., margins=FALSE, geom = "auto",
```

Arguments

x	x values
y	y values
z	z values
...	other arguments passed on to the geom functions
data	data frame to use (optional)
facets	faceting formula to use
margins	whether or not margins will be displayed
geom	geom to use (can be a vector of multiple names)
stat	statistic to use (can be a vector of multiple names)
position	position adjustment to use (can be a vector of multiple names)
xlim	limits for x axis (aesthetics to range of data)
ylim	limits for y axis (aesthetics to range of data)
log	which variables to log transform ("x", "y", or "xy")
main	character vector or expression for plot title
xlab	character vector or expression for x axis label
ylab	character vector or expression for y axis label
asp	the y/x aspect ratio

Details

You can use it like you'd use the `plot` function.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
# Use data from data.frame
qplot(mpg, wt, data=mtcars)
qplot(mpg, wt, data=mtcars, colour=cyl)
qplot(mpg, wt, data=mtcars, size=cyl)
qplot(mpg, wt, data=mtcars, facets=vs ~ am)

# Use data from local environment
attach(mtcars)
qplot(hp, wt)
qplot(hp, wt, colour=cyl)
qplot(hp, wt, size=cyl)
qplot(hp, wt, facets=vs ~ am)

qplot(1:10, rnorm(10), colour = runif(10))
qplot(1:10, letters[1:10])
mod <- lm(mpg ~ wt, data=mtcars)
```

```
qplot(resid(mod), fitted(mod))
qplot(resid(mod), fitted(mod), facets = . ~ vs)

f <- function() {
a <- 1:10
b <- a ^ 2
qplot(a, b)
}
f()

# qplot will attempt to guess what geom you want depending on the input
# both x and y supplied = scatterplot
qplot(mpg, wt, data = mtcars)
# just x supplied = histogram
qplot(mpg, data = mtcars)
# just y supplied = scatterplot, with x = seq_along(y)
qplot(y = mpg, data = mtcars)

# Use different geoms
qplot(mpg, wt, geom="path")
qplot(factor(cyl), wt, geom=c("boxplot", "jitter"))
```

`rescale` *Rescale numeric vector*

Description

Rescale numeric vector to have specified minimum and maximum.

Usage

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

Arguments

`x`	data to rescale
`to`	range to scale to
`from`	range to scale from, defaults to range of data
`clip`	should values be clipped to specified range?

Author(s)

Hadley Wickham <h.wickham@gmail.com>

scale_alpha_continuous *scale_alpha_continuous*

Description

Alpha scale for continuous variable

Usage

```
scale_alpha_continuous(name = NULL, limits = NULL, breaks = NULL,
    labels = NULL, trans = NULL, to = c(0.1, 1), legend = TRUE,
    ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
limits	numeric vector of length 2, giving the extent of the scale
breaks	numeric vector indicating where breaks should lie
labels	character vector giving labels associated with breaks
trans	a transformer to use
to	numeric vector of length 2, giving minimum and maximum after transformation
legend	NULL
...	ignored

Details

This page describes scale_alpha_continuous, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_discrete: Discrete position scales
- http://had.co.nz/ggplot2/scale_alpha_continuous.html

Examples

```
## Not run:
(p <- qplot(mpg, cyl, data=mtcars, alpha=cyl))
p + scale_alpha("cylinders")
p + scale_alpha("number\nof\ncylinders")

p + scale_alpha(to = c(0.4, 0.8))

## End(Not run)
```

`scale_brewer` *scale_brewer*

Description

Sequential, diverging and qualitative colour scales from colorbrewer.org

Usage

```
scale_colour_brewer(name = NULL, palette = 1, type = "qual",
    na.colour = "grey80", limits = NULL, breaks = NULL, labels = NULL,
    formatter = identity, legend = TRUE, ...)
```

Arguments

`name`	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
`palette`	Either numeric or character. If numeric, selects the nth palette of type type. If character, selects the named palette. Get a complete list of all parameters by running `RColorBrewer::display.brewer.all(n=8, exact.n=FALSE)`
`type`	Type of scale. One of 'div' (diverging), 'qual' (qualitative, the default), 'seq' (sequential), or 'all' (all). Only used when palette is numeric.
`na.colour`	colour to use for missing values
`limits`	numeric vector of length 2, giving the extent of the scale
`breaks`	numeric vector indicating where breaks should lie
`labels`	character vector giving labels associated with breaks
`formatter`	NULL
`legend`	NULL
`...`	other arguments

Details

See colorbrewer.org for more info

This page describes scale_brewer, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/scale_brewer.html

Examples

```
## Not run:
dsamp <- diamonds[sample(nrow(diamonds), 1000), ]
(d <- qplot(carat, price, data=dsamp, colour=clarity))

# Change scale label
d + scale_colour_brewer()
d + scale_colour_brewer("clarity")
d + scale_colour_brewer(expression(clarity[beta]))

# Select brewer palette to use, see ?brewer.pal for more details
d + scale_colour_brewer(type="seq")
d + scale_colour_brewer(type="seq", palette=3)

RColorBrewer::display.brewer.all(n=8, exact.n=FALSE)

d + scale_colour_brewer(palette="Blues")
d + scale_colour_brewer(palette="Set1")

# scale_fill_brewer works just the same as
# scale_colour_brewer but for fill colours
ggplot(diamonds, aes(x=price, fill=cut)) +
  geom_histogram(position="dodge", binwidth=1000) +
  scale_fill_brewer()


## End(Not run)
```

---

scale_continuous *scale_continuous*

Description

Continuous position scale

Usage

```
scale_x_continuous(name = NULL, limits = NULL, breaks = NULL,
    labels = NULL, trans = NULL, expand = c(0.05, 0), minor_breaks = NULL,
    formatter = "scientific", legend = TRUE, ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
limits	numeric vector of length 2, giving the extent of the scale
breaks	numeric vector indicating where breaks should lie
labels	character vector giving labels associated with breaks
trans	a transformer to use
expand	numeric vector of length 2, giving multiplicative and additive expansion factors
minor_breaks	NULL
formatter	NULL
legend	NULL
...	ignored

Details

This page describes scale_continuous, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_discrete: Discrete position scales
- http://had.co.nz/ggplot2/scale_continuous.html

Examples

```
## Not run:
(m <- qplot(rating, votes, data=subset(movies, votes > 1000), na.rm = T))

# Manipulating the default position scales lets you:

#  * change the axis labels
m + scale_y_continuous("number of votes")
m + scale_y_continuous(expression(votes^alpha))
```

```
#  * modify the axis limits
m + scale_y_continuous(limits=c(0, 5000))
m + scale_y_continuous(limits=c(1000, 10000))
m + scale_x_continuous(limits=c(7, 8))

# you can also use the short hand functions xlim and ylim
m + ylim(0, 5000)
m + ylim(1000, 10000)
m + xlim(7, 8)

#  * choose where the ticks appear
m + scale_x_continuous(breaks=1:10)
m + scale_x_continuous(breaks=c(1,3,7,9))

#  * manually label the ticks
m + scale_x_continuous(breaks=c(2,5,8), labels=c("two", "five", "eight"))
m + scale_x_continuous(breaks=c(2,5,8), labels=c("horrible", "ok", "awesome"))
m + scale_x_continuous(breaks=c(2,5,8), labels=expression(Alpha, Beta, Omega))

# There are also a wide range of transformations you can use:
m + scale_y_log10()
m + scale_y_log()
m + scale_y_log2()
m + scale_y_sqrt()
m + scale_y_reverse()
# see ?transformer for a full list

# You can control the formatting of the labels with the formatter
# argument.  Some common formats are built in:
x <- rnorm(10) * 100000
y <- seq(0, 1, length = 10)
p <- qplot(x, y)
p + scale_y_continuous(formatter = "percent")
p + scale_y_continuous(formatter = "dollar")
p + scale_x_continuous(formatter = "comma")

# qplot allows you to do some of this with a little less typing:
#   * axis limits
qplot(rating, votes, data=movies, ylim=c(1e4, 5e4))
#   * axis labels
qplot(rating, votes, data=movies, xlab="My x axis", ylab="My y axis")
#   * log scaling
qplot(rating, votes, data=movies, log="xy")

## End(Not run)
```

---

scale_date *scale_date*

Description

Position scale, date

Usage

```
scale_x_date(name = NULL, limits = NULL, major = NULL, minor = NULL,
    format = NULL, expand = c(0.05, 0), ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
limits	numeric vector of length 2, giving the extent of the scale
major	NULL
minor	NULL
format	NULL
expand	numeric vector of length 2, giving multiplicative and additive expansion factors
...	ignored

Details

This page describes scale_date, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_discrete: Discrete position scales
- http://had.co.nz/ggplot2/scale_date.html

Examples

```
## Not run:
# We'll start by creating some nonsense data with dates
df <- data.frame(
  date = seq(Sys.Date(), len=100, by="1 day")[sample(100, 50)],
  price = runif(50)
)
df <- df[order(df$date), ]
dt <- qplot(date, price, data=df, geom="line") + opts(aspect.ratio = 1/4)
```

```
# We can control the format of the labels, and the frequency of
# the major and minor tickmarks.  See ?format.Date and ?seq.Date
# for more details.
dt + scale_x_date()
dt + scale_x_date(format="
dt + scale_x_date(format="
dt + scale_x_date(major="months", minor="weeks", format="
dt + scale_x_date(major="months", minor="3 days", format="
dt + scale_x_date(major="years", format="

# The date scale will attempt to pick sensible defaults for
# major and minor tick marks
qplot(date, price, data=df[1:10,], geom="line")
qplot(date, price, data=df[1:4,], geom="line")

df <- data.frame(
  date = seq(Sys.Date(), len=1000, by="1 day"),
  price = runif(500)
)
qplot(date, price, data=df, geom="line")

# A real example using economic time series data
qplot(date, psavert, data=economics)
qplot(date, psavert, data=economics, geom="path")

end <- max(economics$date)
last_plot() + scale_x_date(lim = c(as.Date("2000-1-1"), end))
last_plot() + scale_x_date(lim = c(as.Date("2005-1-1"), end))
last_plot() + scale_x_date(lim = c(as.Date("2006-1-1"), end))

# If we want to display multiple series, one for each variable
# it's easiest to first change the data from a "wide" to a "long"
# format:
em <- melt(economics, id = "date")

# Then we can group and facet by the new "variable" variable
qplot(date, value, data = em, geom = "line", group = variable)
qplot(date, value, data = em, geom = "line", group = variable) +
  facet_grid(variable ~ ., scale = "free_y")


## End(Not run)
```

`scale_datetime` *scale_datetime*

Description

Position scale, date time

Usage

```
scale_x_datetime(name = NULL, limits = NULL, major = NULL, minor = NULL,
    format = NULL, expand = c(0.05, 0), tz = "", ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
limits	numeric vector of length 2, giving the extent of the scale
major	NULL
minor	NULL
format	NULL
expand	numeric vector of length 2, giving multiplicative and additive expansion factors
tz	NULL
...	ignored

Details

This page describes scale_datetime, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_discrete: Discrete position scales
- http://had.co.nz/ggplot2/scale_datetime.html

Examples

```
## Not run:
start <- ISOdate(2001, 1, 1, tz = "")
df <- data.frame(
  day30  = start + round(runif(100, max = 30 * 86400)),
  day7   = start + round(runif(100, max = 7 * 86400)),
  day    = start + round(runif(100, max = 86400)),
  hour10 = start + round(runif(100, max = 10 * 3600)),
  hour5 = start + round(runif(100, max = 5 * 3600)),
  hour  = start + round(runif(100, max = 3600)),
  min10 = start + round(runif(100, max = 10 * 60)),
  min5  = start + round(runif(100, max = 5 * 60)),
  min   = start + round(runif(100, max = 60)),
```

```
  sec10 = start + round(runif(100, max = 10)),
  y = runif(100)
)

# Automatic scale selection
qplot(sec10, y, data = df)
qplot(min, y, data = df)
qplot(min5, y, data = df)
qplot(min10, y, data = df)
qplot(hour, y, data = df)
qplot(hour5, y, data = df)
qplot(hour10, y, data = df)
qplot(day, y, data = df)
qplot(day30, y, data = df)

# Manual scale selection
qplot(day30, y, data = df)
last_plot() + scale_x_datetime(major = "2 weeks")
last_plot() + scale_x_datetime(major = "2 weeks", minor = "1 week")
last_plot() + scale_x_datetime(major = "10 days")
# See ?strptime for formatting parameters
last_plot() + scale_x_datetime(major = "10 days", format = "


## End(Not run)
```

`scale_discrete` *scale_discrete*

Description

Discrete position scale

Usage

```
scale_x_discrete(name = NULL, expand = c(0.05, 0.55), limits = NULL,
    breaks = NULL, labels = NULL, formatter = identity, drop = FALSE,
    legend = TRUE, ...)
```

Arguments

`name`	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
`expand`	numeric vector of length 2, giving multiplicative and additive expansion factors
`limits`	numeric vector of length 2, giving the extent of the scale
`breaks`	numeric vector indicating where breaks should lie
`labels`	character vector giving labels associated with breaks
`formatter`	NULL

drop	NULL
legend	NULL
...	ignored

Details

This page describes scale_discrete, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/scale_discrete.html

Examples

```
## Not run:
qplot(cut, data=diamonds, stat="bin")
qplot(cut, data=diamonds, geom="bar")

# The discrete position scale is added automatically whenever you
# have a discrete position.

(d <- qplot(cut, clarity, data=subset(diamonds, carat > 1), geom="jitter"))

d + scale_x_discrete("Cut")
d + scale_x_discrete("Cut", labels = c("Fair" = "F","Good" = "G",
  "Very Good" = "VG","Perfect" = "P","Ideal" = "I"))

d + scale_y_discrete("Clarity")
d + scale_x_discrete("Cut") + scale_y_discrete("Clarity")

# Use limits to adjust the which levels (and in what order)
# are displayed
d + scale_x_discrete(limits=c("Fair","Ideal"))

# you can also use the short hand functions xlim and ylim
d + xlim("Fair","Ideal", "Good")
d + ylim("I1", "IF")

# See ?reorder to reorder based on the values of another variable
qplot(manufacturer, cty, data=mpg)
qplot(reorder(manufacturer, cty), cty, data=mpg)
qplot(reorder(manufacturer, displ), cty, data=mpg)
```

```
# Use abbreviate as a formatter to reduce long names
qplot(reorder(manufacturer, cty), cty, data=mpg) +
  scale_x_discrete(formatter = "abbreviate")


## End(Not run)
```

---

scale_gradient *scale_gradient*

Description

Smooth gradient between two colours

Usage

```
scale_colour_gradient(name = NULL, low = "#3B4FB8", high = "#B71B1A",
    space = "rgb", ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
low	colour at low end of scale
high	colour at high end of scale
space	colour space to interpolate through, rgb or Lab, see ?colorRamp for details
...	other arguments

Details

This page describes scale_gradient, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_gradient2: continuous colour scale with midpoint
- colorRamp: for details of interpolation algorithm
- http://had.co.nz/ggplot2/scale_gradient.html

Examples

```
## Not run:
# It's hard to see, but look for the bright yellow dot
# in the bottom right hand corner
dsub <- subset(diamonds, x > 5 & x < 6 & y > 5 & y < 6)
(d <- qplot(x, y, data=dsub, colour=z))
# That one point throws our entire scale off.  We could
# remove it, or manually tweak the limits of the scale

# Tweak scale limits.  Any points outside these
# limits will not be plotted, but will continue to affect the
# calculate of statistics, etc
d + scale_colour_gradient(limits=c(3, 10))
d + scale_colour_gradient(limits=c(3, 4))
# Setting the limits manually is also useful when producing
# multiple plots that need to be comparable

# Alternatively we could try transforming the scale:
d + scale_colour_gradient(trans = "log")
d + scale_colour_gradient(trans = "sqrt")

# Other more trivial manipulations, including changing the name
# of the scale and the colours.

d + scale_colour_gradient("Depth")
d + scale_colour_gradient(expression(Depth[mm]))

d + scale_colour_gradient(limits=c(3, 4), low="red")
d + scale_colour_gradient(limits=c(3, 4), low="red", high="white")
# Much slower
d + scale_colour_gradient(limits=c(3, 4), low="red", high="white", space="Lab")
d + scale_colour_gradient(limits=c(3, 4), space="Lab")

# scale_fill_continuous works similarly, but for fill colours
(h <- qplot(x - y, data=dsub, geom="histogram", binwidth=0.01, fill=..count..))
h + scale_fill_continuous(low="black", high="pink", limits=c(0,3100))

## End(Not run)
```

`scale_gradient2` *scale_gradient2*

Description

Smooth gradient between three colours (high, low and midpoints)

Usage

```
scale_colour_gradient2(name = NULL, low = muted("red"), mid = "white",
    high = muted("blue"), midpoint = 0, space = "rgb", ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
low	colour at low end of scale
mid	colour at mid point of scale
high	colour at high end of scale
midpoint	position of mid point of scale, defaults to 0
space	colour space to interpolate through, rgb or Lab, see ?colorRamp for details
...	other arguments

Details

This page describes scale_gradient2, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_gradient: continuous colour scale
- colorRamp: for details of interpolation algorithm
- http://had.co.nz/ggplot2/scale_gradient2.html

Examples

```
## Not run:
dsub <- subset(diamonds, x > 5 & x < 6 & y > 5 & y < 6)
dsub$diff <- with(dsub, sqrt(abs(x-y))* sign(x-y))
(d <- qplot(x, y, data=dsub, colour=diff))

d + scale_colour_gradient2()
# Change scale name
d + scale_colour_gradient2(expression(sqrt(abs(x - y))))
d + scale_colour_gradient2("Difference\nbetween\nwidth and\nheight")

# Change limits and colours
d + scale_colour_gradient2(limits=c(-0.2, 0.2))

# Using "muted" colours makes for pleasant graphics
# (and they have better perceptual properties too)
d + scale_colour_gradient2(low="red", high="blue")
d + scale_colour_gradient2(low=muted("red"), high=muted("blue"))
```

```
# Using the Lab colour space also improves perceptual properties
# at the price of slightly slower operation
d + scale_colour_gradient2(space="Lab")

# About 5
# idea to avoid that combination
d + scale_colour_gradient2(high=muted("green"))

# We can also make the middle stand out
d + scale_colour_gradient2(mid=muted("green"), high="white", low="white")

# or use a non zero mid point
(d <- qplot(carat, price, data=diamonds, colour=price/carat))
d + scale_colour_gradient2(midpoint=mean(diamonds$price / diamonds$carat))

# Fill gradients work much the same way
p <- qplot(letters[1:5], 1:5, fill= c(-3, 3, 5, 2, -2), geom="bar")
p + scale_fill_gradient2("fill")
# Note how positive and negative values of the same magnitude
# have similar intensity

## End(Not run)
```

`scale_gradientn` *scale_gradientn*

Description

Smooth gradient between n colours

Usage

```
scale_colour_gradientn(name = NULL, colours, values = NULL, rescale = TRUE,
    space = "rgb", ...)
```

Arguments

`name`	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
`colours`	NULL
`values`	NULL
`rescale`	NULL
`space`	colour space to interpolate through, rgb or Lab, see ?colorRamp for details
`...`	other arguments

Details

This page describes scale_gradientn, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_gradient: continuous colour scale with midpoint
- colorRamp: for details of interpolation algorithm
- http://had.co.nz/ggplot2/scale_gradientn.html

Examples

```
## Not run:
# scale_colour_gradient make it easy to use existing colour palettes

dsub <- subset(diamonds, x > 5 & x < 6 & y > 5 & y < 6)
dsub$diff <- with(dsub, sqrt(abs(x-y))* sign(x-y))
(d <- qplot(x, y, data=dsub, colour=diff))

d + scale_colour_gradientn(colour = rainbow(7))
breaks <- c(-0.5, 0, 0.5)
d + scale_colour_gradientn(colour = rainbow(7),
  breaks = breaks, labels = format(breaks))

d + scale_colour_gradientn(colour = topo.colors(10))
d + scale_colour_gradientn(colour = terrain.colors(10))

# You can force them to be symmetric by supplying a vector of
# values, and turning rescaling off
max_val <- max(abs(dsub$diff))
values <- seq(-max_val, max_val, length = 11)

d + scale_colour_gradientn(colours = topo.colors(10),
  values = values, rescale = FALSE)
d + scale_colour_gradientn(colours = terrain.colors(10),
  values = values, rescale = FALSE)



## End(Not run)
```

`scale_grey` *scale_grey*

Description

Sequential grey colour scale

Usage

```
scale_colour_grey(name = NULL, start = 0.2, end = 0.8, limits = NULL,
    breaks = NULL, labels = NULL, formatter = identity, legend = TRUE,
    ...)
```

Arguments

`name`	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
`start`	starting grey colour (between 0 and 1)
`end`	ending grey colour (between 0 and 1)
`limits`	numeric vector of length 2, giving the extent of the scale
`breaks`	numeric vector indicating where breaks should lie
`labels`	character vector giving labels associated with breaks
`formatter`	NULL
`legend`	NULL
`...`	other arguments

Details

Based on ?gray.colors

This page describes scale_grey, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Author(s)

Hadley Wickham, `http://had.co.nz/`

See Also

- `http://had.co.nz/ggplot2/scale_grey.html`

Examples

```
## Not run:
p <- qplot(mpg, wt, data=mtcars, colour=factor(cyl))
p + scale_colour_grey()
p + scale_colour_grey(end = 0)

# You may want to turn off the pale grey background with this scale
p + scale_colour_grey() + theme_bw()

## End(Not run)
```

---

scale_hue *scale_hue*

Description

Qualitative colour scale with evenly spaced hues

Usage

```
scale_colour_hue(name = NULL, h = c(0, 360) + 15, l = 65, c = 100,
    limits = NULL, breaks = NULL, labels = NULL, h.start = 0,
    direction = 1, formatter = identity, legend = TRUE, ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
h	range of hues to use, in [0, 360]
l	luminance (lightness), in [0, 100]
c	chroma (intensity of colour)
limits	numeric vector of length 2, giving the extent of the scale
breaks	numeric vector indicating where breaks should lie
labels	character vector giving labels associated with breaks
h.start	hue to start at
direction	direction to travel around the colour wheel, 1 = clockwise, -1 = counter-clockwise
formatter	NULL
legend	NULL
...	other arguments

Details

This page describes scale_hue, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/scale_hue.html

Examples

```
## Not run:
dsamp <- diamonds[sample(nrow(diamonds), 1000), ]
(d <- qplot(carat, price, data=dsamp, colour=clarity))

# Change scale label
d + scale_colour_hue()
d + scale_colour_hue("clarity")
d + scale_colour_hue(expression(clarity[beta]))

# Adjust luminosity and chroma
d + scale_colour_hue(l=40, c=30)
d + scale_colour_hue(l=70, c=30)
d + scale_colour_hue(l=70, c=150)
d + scale_colour_hue(l=80, c=150)

# Change range of hues used
d + scale_colour_hue(h=c(0, 90))
d + scale_colour_hue(h=c(90, 180))
d + scale_colour_hue(h=c(180, 270))
d + scale_colour_hue(h=c(270, 360))

# Vary opacity
# (only works with pdf, quartz and cairo devices)
d <- ggplot(dsamp, aes(carat, price, colour = clarity))
d + geom_point(alpha = 0.9)
d + geom_point(alpha = 0.5)
d + geom_point(alpha = 0.2)

## End(Not run)
```

---

scale_identity *scale_identity*

Description

Use values without scaling

Usage

```
scale_alpha_identity(name = NULL, breaks = NULL, labels = NULL,
    formatter = NULL, legend = TRUE, ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
breaks	numeric vector indicating where breaks should lie
labels	character vector giving labels associated with breaks
formatter	NULL
legend	NULL
...	ignored

Details

This page describes scale_identity, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/scale_identity.html

Examples

```
## Not run:
colour <- c("red", "green", "blue", "yellow")
qplot(1:4, 1:4, fill = colour, geom = "tile")
qplot(1:4, 1:4, fill = colour, geom = "tile") + scale_fill_identity()

# To get a legend, you also need to supply the labels to
# be used on the legend
qplot(1:4, 1:4, fill = colour, geom = "tile") +
  scale_fill_identity("trt", labels = letters[1:4], breaks = colour)

# cyl scaled to appropriate size
qplot(mpg, wt, data = mtcars, size = cyl)

# cyl used as point size
qplot(mpg, wt, data = mtcars, size = cyl) + scale_size_identity()

## End(Not run)
```

scale_linetype_discrete *scale_linetype_discrete*

Description

Scale for line patterns

Usage

```
scale_linetype_discrete(name = NULL, expand = c(0.05, 0.55),
    limits = NULL, breaks = NULL, labels = NULL, formatter = identity,
    drop = FALSE, legend = TRUE, ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
expand	numeric vector of length 2, giving multiplicative and additive expansion factors
limits	numeric vector of length 2, giving the extent of the scale
breaks	numeric vector indicating where breaks should lie
labels	character vector giving labels associated with breaks
formatter	NULL
drop	NULL
legend	NULL
...	ignored

Details

This page describes scale_linetype_discrete, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/scale_linetype_discrete.html

Examples

```
## Not run:
ec_scaled <- data.frame(
  date = economics$date,
  rescaler(economics[, -(1:2)], "range")
)
ecm <- melt(ec_scaled, id = "date")

qplot(date, value, data=ecm, geom="line", group=variable)
qplot(date, value, data=ecm, geom="line", linetype=variable)
qplot(date, value, data=ecm, geom="line", colour=variable)

# See scale_manual for more flexibility

## End(Not run)
```

---

scale_manual *scale_manual*

Description

Create your own discrete scale

Usage

```
scale_colour_manual(name = NULL, values = NULL, limits = NULL,
    breaks = NULL, labels = NULL, formatter = identity, legend = TRUE,
    ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
values	NULL
limits	numeric vector of length 2, giving the extent of the scale
breaks	numeric vector indicating where breaks should lie
labels	character vector giving labels associated with breaks
formatter	NULL
legend	NULL
...	ignored

Details

This page describes scale_manual, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/scale_manual.html

Examples

```
## Not run:
p <- qplot(mpg, wt, data = mtcars, colour = factor(cyl))

p + scale_colour_manual(values = c("red","blue", "green"))
p + scale_colour_manual(
  values = c("8" = "red","4" = "blue","6" = "green"))

# As with other scales you can use breaks to control the appearance
# of the legend
cols <- c("8" = "red","4" = "blue","6" = "darkgreen", "10" = "orange")
p + scale_colour_manual(values = cols)
p + scale_colour_manual(values = cols, breaks = c("4", "6", "8"))
p + scale_colour_manual(values = cols, breaks = c("8", "6", "4"))
p + scale_colour_manual(values = cols, breaks = c("4", "6", "8"),
  labels = c("four", "six", "eight"))

# And limits to control the possible values of the scale
p + scale_colour_manual(values = cols, limits = c("4", "8"))
p + scale_colour_manual(values = cols, limits = c("4", "6", "8", "10"))


## End(Not run)
```

---

scale_shape_discrete
scale_shape_discrete

Description

Scale for shapes, aka glyphs

Usage

```
scale_shape_discrete(name = NULL, solid = TRUE, limits = NULL,
    breaks = NULL, labels = NULL, formatter = identity, legend = TRUE,
    ...)
```

Arguments

name	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
solid	NULL
limits	numeric vector of length 2, giving the extent of the scale
breaks	numeric vector indicating where breaks should lie
labels	character vector giving labels associated with breaks
formatter	NULL
legend	NULL
...	ignored

Details

This page describes scale_shape_discrete, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/scale_shape_discrete.html

Examples

```
## Not run:
dsmall <- diamonds[sample(nrow(diamonds), 100), ]

(d <- qplot(carat, price, data=dsmall, shape=cut))
d + scale_shape(solid = TRUE) # the default
d + scale_shape(solid = FALSE)
d + scale_shape(name="Cut of diamond")
d + scale_shape(name="Cut of\ndiamond")

# To change order of levels, change order of
# underlying factor
levels(dsmall$cut) <- c("Fair", "Good", "Very Good", "Premium", "Ideal")

# Need to recreate plot to pick up new data
qplot(price, carat, data=dsmall, shape=cut)

# Or for short:
d
```

```
## End(Not run)
```

`scale_size_continuous` *scale_size_continuous*

Description

Size scale for continuous variable

Usage

```
scale_size_continuous(name = NULL, limits = NULL, breaks = NULL,
    labels = NULL, trans = NULL, to = c(1, 6), legend = TRUE,
    ...)
```

Arguments

`name`	name of scale to appear in legend or on axis. Maybe be an expression: see ?plotmath
`limits`	numeric vector of length 2, giving the extent of the scale
`breaks`	numeric vector indicating where breaks should lie
`labels`	character vector giving labels associated with breaks
`trans`	a transformer to use
`to`	a numeric vector of length 2 that specifies the minimum and maximum size of the plotting symbol after transformation.
`legend`	NULL
`...`	other arguments

Details

This page describes scale_size_continuous, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- scale_manual: for sizing discrete variables
- http://had.co.nz/ggplot2/scale_size_continuous.html

Examples

```
## Not run:
(p <- qplot(mpg, cyl, data=mtcars, size=cyl))
p + scale_size("cylinders")
p + scale_size("number\nof\ncylinders")

p + scale_size(to = c(0, 10))
p + scale_size(to = c(1, 2))

# Map area, instead of width/radius
# Perceptually, this is a little better
p + scale_area()
p + scale_area(to = c(1, 25))

# Also works with factors, but not a terribly good
# idea, unless your factor is ordered, as in this example
qplot(mpg, cyl, data=mtcars, size=factor(cyl))

# To control the size mapping for discrete variable, use
# scale_size_manual:
last_plot() + scale_size_manual(values=c(2,4,6))


## End(Not run)
```

---

scientific *Scientific formatter*

Description

Default scientific formatting

Usage

```
scientific(x)
```

Arguments

x	numeric vector to format

Author(s)

Hadley Wickham <h.wickham@gmail.com>

`Seals vector field` *Vector field of seal movements*

Description

This vector field was produced from the data described in Brillinger, D.R., Preisler, H.K., Ager, A.A. and Kie, J.G. "An exploratory data analysis (EDA) of the paths of moving animals". J. Statistical Planning and Inference 122 (2004), 43-63, using the methods of Brillinger, D.R., "Learning a potential function from a trajectory", Signal Processing Letters. December (2007).

Usage

```
data(seals)
```

Format

A data frame with 1155 rows and 4 variables

References

http://www.stat.berkeley.edu/~brill/Papers/jspifinal.pdf

`stat_abline` *stat_abline*

Description

Add a line with slope and intercept

Usage

```
stat_abline(mapping = NULL, data = NULL, geom = "abline", position = "identity",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`geom`	geometric used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

This page describes stat_abline, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_abline.html

Examples

```
## Not run:
# See geom_abline for examples

## End(Not run)
```

---

stat_bin *stat_bin*

Description

Bin data

Usage

```
stat_bin(mapping = NULL, data = NULL, geom = "bar", position = "stack",
    width = 0.9, drop = FALSE, right = TRUE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
width	Width of bars when used with categorical data
drop	If TRUE, remove all bins with zero counts
right	Should intervals be closed on the right (a, b], or not [a, b)
...	other arguments

Details

Missing values are currently silently dropped.

This page describes stat_bin, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_bin. Aesthetics are mapped to variables in the data with the aes function: `stat_bin(aes(x = var))`

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

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_bin.html

Examples

```
## Not run:
simple <- data.frame(x = rep(1:10, each = 2))
base <- ggplot(simple, aes(x))
# By default, right = TRUE, and intervals are of the form (a, b]
base + stat_bin(binwidth = 1, drop = FALSE, right = TRUE, col = "black")
# If right = FALSE intervals are of the form [a, b)
base + stat_bin(binwidth = 1, drop = FALSE, right = FALSE, col = "black")

m <- ggplot(movies, aes(x=rating))
m + stat_bin()
m + stat_bin(binwidth=0.1)
m + stat_bin(breaks=seq(4,6, by=0.1))
# See geom_histogram for more histogram examples

# To create a unit area histogram, use aes(y = ..density..)
(linehist <- m + stat_bin(aes(y = ..density..), binwidth=0.1,
  geom="line", position="identity"))
linehist + stat_density(colour="blue", fill=NA)

# Also works with categorical variables
ggplot(movies, aes(x=mpaa)) + stat_bin()
qplot(mpaa, data=movies, stat="bin")


## End(Not run)
```

stat_bin2d *stat_bin2d*

Description

Bin 2d plane into rectangles

Usage

```
stat_bin2d(mapping = NULL, data = NULL, geom = "rect", position = "identity",
    bins = 30, drop = TRUE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
bins	NULL
drop	NULL
...	ignored

Details

This page describes stat_bin2d, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_bin2d. Aesthetics are mapped to variables in the data with the aes function: `stat_bin2d(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- fill: internal colour

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- stat_binhex: For hexagonal binning
- http://had.co.nz/ggplot2/stat_bin2d.html

Examples

```
## Not run:
d <- ggplot(diamonds, aes(carat, price))
d + stat_bin2d()
d + geom_bin2d()

# You can control the size of the bins by specifying the number of
# bins in each direction:
d + stat_bin2d(bins = 10)
d + stat_bin2d(bins = 30)

# Or by specifying the width of the bins
d + stat_bin2d(binwidth = c(1, 1000))
d + stat_bin2d(binwidth = c(.1, 500))

# Or with a list of breaks
x <- seq(min(diamonds$carat), max(diamonds$carat), by = 0.1)
y <- seq(min(diamonds$price), max(diamonds$price), length = 50)
d + stat_bin2d(breaks = list(x = x, y = y))

# With qplot
qplot(x, y, data = diamonds, geom="bin2d",
  xlim = c(4, 10), ylim = c(4, 10))
qplot(x, y, data = diamonds, geom="bin2d", binwidth = c(0.1, 0.1),
  xlim = c(4, 10), ylim = c(4, 10))

## End(Not run)
```

---

stat_binhex *stat_binhex*

Description

Bin 2d plane into hexagons

Usage

```
stat_binhex(mapping = NULL, data = NULL, geom = "hex", position = "identity",
    bins = 30, na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
bins	NULL
na.rm	NULL
...	ignored

Details

This page describes stat_binhex, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_binhex. Aesthetics are mapped to variables in the data with the aes function: `stat_binhex(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- fill: internal colour

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- stat_bin2d: For rectangular binning
- http://had.co.nz/ggplot2/stat_binhex.html

Examples

```
## Not run:
d <- ggplot(diamonds, aes(carat, price))
d + stat_binhex()
d + geom_hex()

# You can control the size of the bins by specifying the number of
# bins in each direction:
d + stat_binhex(bins = 10)
d + stat_binhex(bins = 30)
```

```
# Or by specifying the width of the bins
d + stat_binhex(binwidth = c(1, 1000))
d + stat_binhex(binwidth = c(.1, 500))

# With qplot
qplot(x, y, data = diamonds, geom="hex", xlim = c(4, 10), ylim = c(4, 10))
qplot(x, y, data = diamonds, geom="hex", xlim = c(4, 10), ylim = c(4, 10),
  binwidth = c(0.1, 0.1))

## End(Not run)
```

`stat_boxplot` *stat_boxplot*

Description

Calculate components of box and whisker plot

Usage

```
stat_boxplot(mapping = NULL, data = NULL, geom = "boxplot", position = "dodge",
    na.rm = FALSE, coef = 1.5, ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`geom`	geometric used by this layer
`position`	position adjustment used by this layer
`na.rm`	NULL
`coef`	NULL
`...`	ignored

Details

This page describes stat_boxplot, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with stat_boxplot. Aesthetics are mapped to variables in the data with the aes function: `stat_boxplot(aes(x = var))`

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

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_boxplot.html

Examples

```
## Not run:
# See geom_boxplot for examples

## End(Not run)
```

---

stat_contour *stat_contour*

Description

Contours of 3d data

Usage

```
stat_contour(mapping = NULL, data = NULL, geom = "path", position = "identity",
    na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
na.rm	NULL
...	ignored

Details

This page describes stat_contour, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_contour. Aesthetics are mapped to variables in the data with the aes function: `stat_contour(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position (**required**)
- `z`: NULL (**required**)
- `order`: NULL

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_contour.html

Examples

```
## Not run:
# Generate data
volcano3d <- melt(volcano)
names(volcano3d) <- c("x", "y", "z")

# Basic plot
v <- ggplot(volcano3d, aes(x, y, z = z))
v + stat_contour()

# Setting bins creates evenly spaced contours in the range of the data
v + stat_contour(bins = 2)
v + stat_contour(bins = 10)

# Setting binwidth does the same thing, parameterised by the distance
# between contours
v + stat_contour(binwidth = 2)
v + stat_contour(binwidth = 5)
v + stat_contour(binwidth = 10)
v + stat_contour(binwidth = 2, size = 0.5, colour = "grey50") +
  stat_contour(binwidth = 10, size = 1)

# Add aesthetic mappings
v + stat_contour(aes(size = ..level..))
v + stat_contour(aes(colour = ..level..))

# Change scale
v + stat_contour(aes(colour = ..level..), size = 2) +
  scale_colour_gradient(low = "brown", high = "white")

# Set aesthetics to fixed value
v + stat_contour(colour = "red")
v + stat_contour(size = 2, linetype = 4)
```

```
# Try different geoms
v + stat_contour(geom="polygon", aes(fill=..level..))
v + geom_tile(aes(fill = z)) + stat_contour()

# Use qplot instead
qplot(x, y, z, data = volcano3d, geom = "contour")
qplot(x, y, z, data = volcano3d, stat = "contour", geom = "path")

## End(Not run)
```

---

stat_density *stat_density*

Description

Density estimation, 1D

Usage

```
stat_density(mapping = NULL, data = NULL, geom = "area", position = "stack",
    adjust = 1, kernel = "gaussian", trim = FALSE, na.rm = FALSE,
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
adjust	see ?density for details
kernel	kernel used for density estimation, see density for details
trim	NULL
na.rm	NULL
...	other arguments

Details

This page describes stat_density, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_density. Aesthetics are mapped to variables in the data with the aes function: `stat_density(aes(x = var))`

- `x`: x position (**required**)
- `y`: y position
- `fill`: internal colour

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `stat_bin`: for the histogram
- `density`: for details of the algorithm used
- http://had.co.nz/ggplot2/stat_density.html

Examples

```
## Not run:
m <- ggplot(movies, aes(x=rating))
m + geom_density()

# Adjust parameters
m + geom_density(kernel = "rectangular")
m + geom_density(kernel = "biweight")
m + geom_density(kernel = "epanechnikov")
m + geom_density(adjust=1/5) # Very rough
m + geom_density(adjust=5) # Very smooth

# Adjust aesthetics
m + geom_density(aes(fill=factor(Drama)), size=2)
# Scale so peaks have same height:
m + geom_density(aes(fill=factor(Drama), y = ..scaled..), size=2)

m + geom_density(colour="darkgreen", size=2)
m + geom_density(colour="darkgreen", size=2, fill=NA)
m + geom_density(colour="darkgreen", size=2, fill="green")

# Change scales
(m <- ggplot(movies, aes(x=votes)) + geom_density(trim = TRUE))
m + scale_x_log10()
m + coord_trans(x="log10")
m + scale_x_log10() + coord_trans(x="log10")

# Also useful with
m + stat_bin()

# Make a volcano plot
```

```
ggplot(diamonds, aes(x = price)) +
  stat_density(aes(ymax = ..density..,  ymin = -..density..),
    fill = "grey50", colour = "grey50",
    geom = "ribbon", position = "identity") +
  facet_grid(. ~ cut) +
  coord_flip()

# Stacked density plots
# If you want to create a stacked density plot, you need to use
# the 'count' (density * n) variable instead of the default density

# Loses marginal densities
qplot(rating, ..density.., data=movies, geom="density", fill=mpaa, position="stack")
# Preserves marginal densities
qplot(rating, ..count.., data=movies, geom="density", fill=mpaa, position="stack")

# You can use position="fill" to produce a conditional density estimate
qplot(rating, ..count.., data=movies, geom="density", fill=mpaa, position="fill")

# Need to be careful with weighted data
m <- ggplot(movies, aes(x=rating, weight=votes))
m + geom_histogram(aes(y = ..count..)) + geom_density(fill=NA)

m <- ggplot(movies, aes(x=rating, weight=votes/sum(votes)))
m + geom_histogram(aes(y=..density..)) + geom_density(fill=NA, colour="black")

movies$decade <- round_any(movies$year, 10)
m <- ggplot(movies, aes(x=rating, colour=decade, group=decade))
m + geom_density(fill=NA)
m + geom_density(fill=NA) + aes(y = ..count..)

# Use qplot instead
qplot(length, data=movies, geom="density", weight=rating)
qplot(length, data=movies, geom="density", weight=rating/sum(rating))

## End(Not run)
```

`stat_density2d` *stat_density2d*

Description

Density estimation, 2D

Usage

```
stat_density2d(mapping = NULL, data = NULL, geom = "density2d",
    position = "identity", na.rm = FALSE, contour = TRUE, n = 100,
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
na.rm	NULL
contour	If TRUE, contour the results of the 2d density estimation.
n	number of grid points in each direction
...	other arguments passed on to ?kde2d

Details

This page describes stat_density2d, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_density2d. Aesthetics are mapped to variables in the data with the aes function: `stat_density2d(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- colour: border colour
- size: size

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_density2d.html

Examples

```
## Not run:
m <- ggplot(movies, aes(x=rating, y=length)) +
  geom_point() +
  scale_y_continuous(limits=c(1, 500))
m + geom_density2d()

dens <- MASS::kde2d(movies$rating, movies$length, n=100)
densdf <- data.frame(expand.grid(rating = dens$x, length = dens$y),
```

```
 z = as.vector(dens$z))
m + geom_contour(aes(z=z), data=densdf)

m + geom_density2d() + scale_y_log10()
m + geom_density2d() + coord_trans(y="log10")

m + stat_density2d(aes(fill = ..level..), geom="polygon")

qplot(rating, length, data=movies, geom=c("point","density2d")) +
  ylim(1, 500)

# If you map an aesthetic to a categorical variable, you will get a
# set of contours for each value of that variable
qplot(rating, length, data = movies, geom = "density2d",
  colour = factor(Comedy), ylim = c(0, 150))
qplot(rating, length, data = movies, geom = "density2d",
  colour = factor(Action), ylim = c(0, 150))
qplot(carat, price, data = diamonds, geom = "density2d", colour = cut)

# Another example ------
d <- ggplot(diamonds, aes(carat, price)) + xlim(1,3)
d + geom_point() + geom_density2d()

# If we turn contouring off, we can use use geoms like tiles:
d + stat_density2d(geom="tile", aes(fill = ..density..), contour = FALSE)
last_plot() + scale_fill_gradient(limits=c(1e-5,8e-4))

# Or points:
d + stat_density2d(geom="point", aes(size = ..density..), contour = FALSE)

## End(Not run)
```

`stat_function` *stat_function*

Description

Superimpose a function

Usage

```
stat_function(mapping = NULL, data = NULL, geom = "path", position = "identity",
    fun, n = 101, args = list(), ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`geom`	geometric used by this layer

position	position adjustment used by this layer
fun	function to use
n	number of points to interpolate along
args	list of additional arguments to pass to fun
...	other arguments

Details

This page describes stat_function, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_function. Aesthetics are mapped to variables in the data with the aes function: `stat_function(aes(x = var))`

- y: y position

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_function.html

Examples

```
## Not run:
x <- rnorm(100)
base <- qplot(x, geom="density")
base + stat_function(fun = dnorm, colour = "red")
base + stat_function(fun = dnorm, colour = "red", arg = list(mean = 3))

## End(Not run)
```

stat_hline *stat_hline*

Description

Add a horizontal line

Usage

```
stat_hline(mapping = NULL, data = NULL, geom = "hline", position = "identity",
    intercept, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
intercept	NULL
...	ignored

Details

This page describes stat_hline, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_hline.html

Examples

```
## Not run:
# See geom_hline for examples

## End(Not run)
```

stat_identity *stat_identity*

Description

Don't transform data

Usage

```
stat_identity(mapping = NULL, data = NULL, geom = "point", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes stat_identity, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_identity.html

Examples

```
## Not run:
# Doesn't do anything, so hard to come up a useful example

## End(Not run)
```

stat_qq *stat_qq*

Description

Calculation for quantile-quantile plot

Usage

```
stat_qq(mapping = NULL, data = NULL, geom = "point", position = "identity",
    distribution = qnorm, dparams = list(), na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
distribution	NULL
dparams	Parameters for distribution function
na.rm	NULL
...	Other arguments passed to distribution function

Details

This page describes stat_qq, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_qq. Aesthetics are mapped to variables in the data with the aes function: `stat_qq(aes(x = var))`

- sample: NULL (**required**)
- x: x position
- y: y position

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_qq.html

Examples

```
## Not run:
# From ?qqplot
y <- rt(200, df = 5)
qplot(sample = y, stat="qq")

# qplot is smart enough to use stat_qq if you use sample
qplot(sample = y)
qplot(sample = precip)

qplot(sample = y, dist = qt, dparams = list(df = 5))

df <- data.frame(y)
ggplot(df, aes(sample = y)) + stat_qq()
ggplot(df, aes(sample = y)) + geom_point(stat = "qq")

# Use fitdistr from MASS to estimate distribution params
params <- as.list(MASS::fitdistr(y, "t")$estimate)
ggplot(df, aes(sample = y)) + stat_qq(dist = qt, dparam = params)

# Using to explore the distribution of a variable
qplot(sample = mpg, data = mtcars)
qplot(sample = mpg, data = mtcars, colour = factor(cyl))

## End(Not run)
```

---

stat_quantile *stat_quantile*

Description

Continuous quantiles

Usage

```
stat_quantile(mapping = NULL, data = NULL, geom = "quantile",
    position = "identity", quantiles = c(0.25, 0.5, 0.75), formula = y ~
        x, method = "rq", na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer

position	position adjustment used by this layer
quantiles	conditional quantiles of y to calculate and display
formula	formula relating y variables to x variables
method	NULL
na.rm	NULL
...	other arguments

Details

This page describes stat_quantile, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_quantile. Aesthetics are mapped to variables in the data with the aes function: `stat_quantile(aes(x = var))`

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

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_quantile.html

Examples

```
## Not run:
msamp <- movies[sample(nrow(movies), 1000), ]
m <- ggplot(msamp, aes(y=rating, x=year)) + geom_point()
m + stat_quantile()
m + stat_quantile(quantiles = 0.5)
m + stat_quantile(quantiles = seq(0.1, 0.9, by=0.1))

# Doesn't work.  Not sure why.
# m + stat_quantile(method = rqss, formula = y ~ qss(x), quantiles = 0.5)

# Add aesthetic mappings
m + stat_quantile(aes(weight=votes))

# Change scale
m + stat_quantile(aes(colour = ..quantile..), quantiles = seq(0.05, 0.95, by=0.05))
m + stat_quantile(aes(colour = ..quantile..), quantiles = seq(0.05, 0.95, by=0.05)) +
```

```
  scale_colour_gradient2(midpoint=0.5, low="green", mid="yellow", high="green")

# Set aesthetics to fixed value
m + stat_quantile(colour="red", size=2, linetype=2)

# Use qplot instead
qplot(year, rating, data=movies, geom="quantile")


## End(Not run)
```

---

stat_smooth *stat_smooth*

Description

Add a smoother

Usage

```
stat_smooth(mapping = NULL, data = NULL, geom = "smooth", position = "identity",
    method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE,
    level = 0.95, na.rm = FALSE, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
method	smoothing method (function) to use, eg. lm, glm, gam, loess, rlm
formula	formula to use in smoothing function, eg. y ~ x, y ~ poly(x, 2), y ~ log(x)
se	display confidence interval around smooth? (true by default, see level to control)
n	number of points to evaluate smoother at
fullrange	should the fit span the full range of the plot, or just the data
level	level of confidence interval to use (0.95 by default)
na.rm	NULL
...	other arguments are passed to smoothing function

Details

Aids the eye in seeing patterns in the presence of overplotting.This page describes stat_smooth, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_smooth. Aesthetics are mapped to variables in the data with the aes function: `stat_smooth(aes(x = var))`

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

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `lm`: for linear smooths
- `glm`: for generalised linear smooths
- `loess`: for local smooths
- http://had.co.nz/ggplot2/stat_smooth.html

Examples

```
## Not run:
c <- ggplot(mtcars, aes(qsec, wt))
c + stat_smooth()
c + stat_smooth() + geom_point()

# Adjust parameters
c + stat_smooth(se = FALSE) + geom_point()

c + stat_smooth(span = 0.9) + geom_point()
c + stat_smooth(method = "lm") + geom_point()

library(splines)
c + stat_smooth(method = "lm", formula = y ~ ns(x,3)) +
  geom_point()
c + stat_smooth(method = MASS::rlm, formula= y ~ ns(x,3)) + geom_point()

# The default confidence band uses a transparent colour.
# This currently only works on a limited number of graphics devices
# (including Quartz, PDF, and Cairo) so you may need to set the
# fill colour to a opaque colour, as shown below
c + stat_smooth(fill = "grey50", size = 2, alpha = 1)
c + stat_smooth(fill = "blue", size = 2, alpha = 1)

# The colour of the line can be controlled with the colour aesthetic
c + stat_smooth(fill="blue", colour="darkblue", size=2)
c + stat_smooth(fill="blue", colour="darkblue", size=2, alpha = 0.2)
```

```
c + geom_point() +
  stat_smooth(fill="blue", colour="darkblue", size=2, alpha = 0.2)

# Smoothers for subsets
c <- ggplot(mtcars, aes(y=wt, x=mpg)) + facet_grid(. ~ cyl)
c + stat_smooth(method=lm) + geom_point()
c + stat_smooth(method=lm, fullrange=T) + geom_point()

# Geoms and stats are automatically split by aesthetics that are factors
c <- ggplot(mtcars, aes(y=wt, x=mpg, colour=factor(cyl)))
c + stat_smooth(method=lm) + geom_point()
c + stat_smooth(method=lm, aes(fill = factor(cyl))) + geom_point()
c + stat_smooth(method=lm, fullrange=TRUE, alpha = 0.1) + geom_point()

# Use qplot instead
qplot(qsec, wt, data=mtcars, geom=c("smooth", "point"))

# Example with logistic regression
data("kyphosis", package="rpart")
qplot(Age, Kyphosis, data=kyphosis)
qplot(Age, data=kyphosis, facets = . ~ Kyphosis, binwidth = 10)
qplot(Age, Kyphosis, data=kyphosis, position="jitter")
qplot(Age, Kyphosis, data=kyphosis, position=position_jitter(height=0.1))

qplot(Age, as.numeric(Kyphosis) - 1, data = kyphosis) +
  stat_smooth(method="glm", family="binomial")
qplot(Age, as.numeric(Kyphosis) - 1, data=kyphosis) +
  stat_smooth(method="glm", family="binomial", formula = y ~ ns(x, 2))


## End(Not run)
```

`stat_spoke` *stat_spoke*

Description

Convert angle and radius to xend and yend

Usage

```
stat_spoke(mapping = NULL, data = NULL, geom = "segment", position = "identity",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`geom`	geometric used by this layer

position	position adjustment used by this layer
...	ignored

Details

This page describes stat_spoke, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Aesthetics

The following aesthetics can be used with stat_spoke. Aesthetics are mapped to variables in the data with the aes function: `stat_spoke(aes(x = var))`

- x: x position (**required**)
- y: y position (**required**)
- angle: angle (**required**)
- radius: NULL (**required**)
- xend: NULL
- yend: NULL

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_spoke.html

Examples

```
## Not run:
df <- expand.grid(x = 1:10, y=1:10)
df$angle <- runif(100, 0, 2*pi)
df$speed <- runif(100, 0, 0.5)

qplot(x, y, data=df) + stat_spoke(aes(angle=angle), radius = 0.5)
last_plot() + scale_y_reverse()

qplot(x, y, data=df) + stat_spoke(aes(angle=angle, radius=speed))

## End(Not run)
```

`stat_sum` *stat_sum*

Description

Sum unique values. Useful for overplotting on scatterplots

Usage

```
stat_sum(mapping = NULL, data = NULL, geom = "point", position = "identity",
    ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`geom`	geometric used by this layer
`position`	position adjustment used by this layer
`...`	ignored

Details

This page describes stat_sum, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with stat_sum. Aesthetics are mapped to variables in the data with the aes function: `stat_sum(aes(x = var))`

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

Author(s)

Hadley Wickham, `http://had.co.nz/`

See Also

- `ggfluctuation`: Fluctuation diagram, which is very similar
- `http://had.co.nz/ggplot2/stat_sum.html`

Examples

```
## Not run:
d <- ggplot(diamonds, aes(x = cut, y = clarity))
# Need to control which group proportion calculated over
# Overall proportion
d + stat_sum(aes(group = 1))
d + stat_sum(aes(group = 1)) + scale_size(to = c(3, 10))
d + stat_sum(aes(group = 1)) + scale_area(to = c(3, 10))
# by cut
d + stat_sum(aes(group = cut))
d + stat_sum(aes(group = cut, colour = cut))
# by clarity
d + stat_sum(aes(group = clarity))
d + stat_sum(aes(group = clarity, colour = cut))

# Instead of proportions, can also use sums
d + stat_sum(aes(size = ..n..))

# Can also weight by another variable
d + stat_sum(aes(group = 1, weight = price))
d + stat_sum(aes(group = 1, weight = price, size = ..n..))

# Or using qplot
qplot(cut, clarity, data = diamonds)
qplot(cut, clarity, data = diamonds, stat = "sum", group = 1)

## End(Not run)
```

`stat_summary` *stat_summary*

Description

Summarise y values at every unique x

Usage

```
stat_summary(mapping = NULL, data = NULL, geom = "pointrange",
    position = "identity", ...)
```

Arguments

`mapping`	mapping between variables and aesthetics generated by aes
`data`	dataset used in this layer, if not specified uses plot dataset
`geom`	geometric used by this layer
`position`	position adjustment used by this layer
`...`	other arguments

Details

stat_summary allows for tremendous flexibilty in the specification of summary functions. The summary function can either operate on a data frame (with argument name data) or on a vector. A simple vector function is easiest to work with as you can return a single number, but is somewhat less flexible. If your summary function operates on a data.frame it should return a data frame with variables that the geom can use.

This page describes stat_summary, see `layer` and `qplot` for how to create a complete plot from individual components.

Value

A `layer`

Aesthetics

The following aesthetics can be used with stat_summary. Aesthetics are mapped to variables in the data with the aes function: `stat_summary(aes(x = var))`

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

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- `geom_errorbar`: error bars
- `geom_pointrange`: range indicated by straight line, with point in the middle
- `geom_linerange`: range indicated by straight line
- `geom_crossbar`: hollow bar with middle indicated by horizontal line
- `stat_smooth`: for continuous analog
- http://had.co.nz/ggplot2/stat_summary.html

Examples

```
## Not run:
# Basic operation on a small dataset
c <- qplot(cyl, mpg, data=mtcars)
c + stat_summary(fun.data = "mean_cl_boot", colour = "red")

p <- qplot(cyl, mpg, data = mtcars, stat="summary", fun.y = "mean")
p
# Don't use ylim to zoom into a summary plot - this throws the
# data away
p + ylim(15, 30)
# Instead use coord_cartesian
p + coord_cartesian(ylim = c(15, 30))
```

```
# You can supply individual functions to summarise the value at
# each x:

stat_sum_single <- function(fun, geom="point", ...) {
  stat_summary(fun.y=fun, colour="red", geom=geom, size = 3, ...)
}

c + stat_sum_single(mean)
c + stat_sum_single(mean, geom="line")
c + stat_sum_single(median)
c + stat_sum_single(sd)

c + stat_summary(fun.y = mean, fun.ymin = min, fun.ymax = max,
  colour = "red")

c + aes(colour = factor(vs)) + stat_summary(fun.y = mean, geom="line")

# Alternatively, you can supply a function that operates on a data.frame.
# A set of useful summary functions is provided from the Hmisc package:

stat_sum_df <- function(fun, geom="crossbar", ...) {
  stat_summary(fun.data=fun, colour="red", geom=geom, width=0.2, ...)
}

c + stat_sum_df("mean_cl_boot")
c + stat_sum_df("mean_sdl")
c + stat_sum_df("mean_sdl", mult=1)
c + stat_sum_df("median_hilow")

# There are lots of different geoms you can use to display the summaries

c + stat_sum_df("mean_cl_normal")
c + stat_sum_df("mean_cl_normal", geom = "errorbar")
c + stat_sum_df("mean_cl_normal", geom = "pointrange")
c + stat_sum_df("mean_cl_normal", geom = "smooth")

# Summaries are much more useful with a bigger data set:
m <- ggplot(movies, aes(x=round(rating), y=votes)) + geom_point()
m2 <- m +
   stat_summary(fun.data = "mean_cl_boot", geom = "crossbar",
     colour = "red", width = 0.3)
m2
# Notice how the overplotting skews off visual perception of the mean
# supplementing the raw data with summary statisitcs is _very_ important

# Next, we'll look at votes on a log scale.

# Transforming the scale performs the transforming before the statistic.
# This means we're calculating the summary on the logged data
m2 + scale_y_log10()
# Transforming the coordinate system performs the transforming after the
# statistic. This means we're calculating the summary on the raw data,
```

```
# and stretching the geoms onto the log scale.  Compare the widths of the
# standard errors.
m2 + coord_trans(y="log10")

## End(Not run)
```

---

stat_unique *stat_unique*

Description

Remove duplicates

Usage

```
stat_unique(mapping = NULL, data = NULL, geom = "point", position = "identity",
    ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
...	ignored

Details

This page describes stat_unique, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_unique.html

Examples

```
## Not run:
ggplot(mtcars, aes(x=vs, y=am)) + geom_point(colour="#00000010")
ggplot(mtcars, aes(x=vs, y=am)) + geom_point(colour="#00000010", stat="unique")

## End(Not run)
```

stat_vline *stat_vline*

Description

Add a vertical line

Usage

```
stat_vline(mapping = NULL, data = NULL, geom = "vline", position = "identity",
    intercept, ...)
```

Arguments

mapping	mapping between variables and aesthetics generated by aes
data	dataset used in this layer, if not specified uses plot dataset
geom	geometric used by this layer
position	position adjustment used by this layer
intercept	NULL
...	ignored

Details

This page describes stat_vline, see layer and qplot for how to create a complete plot from individual components.

Value

A layer

Author(s)

Hadley Wickham, http://had.co.nz/

See Also

- http://had.co.nz/ggplot2/stat_vline.html

Examples

```
## Not run:
# See geom_vline for examples

## End(Not run)
```

theme_blank *Theme element: blank*

Description

This theme element draws nothing, and assigns no space

Usage

```
theme_blank()
```

Author(s)

Hadley Wickham <h.wickham@gmail.com>

theme_bw *Black and white theme*

Description

Produce a theme with white background and black gridlines

Usage

```
theme_bw(base_size =  12, base_family = "")
```

Arguments

base_size	base font size
base_family	base font family

Author(s)

Hadley Wickham <h.wickham@gmail.com>

theme_grey *Grey theme*

Description

Produce a theme with grey background and white gridlines

Usage

```
theme_grey(base_size = 12, base_family = "")
```

Arguments

base_size	base font size
base_family	base font family

Author(s)

Hadley Wickham <h.wickham@gmail.com>

theme_line *Theme element: line*

Description

This element draws a line between two (or more) points

Usage

```
theme_line(colour = "black", size = 0.5, linetype = 1)
```

Arguments

colour	line color
size	line size
linetype	line type

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

polylineGrob for underlying grid function, link{theme_segment}

theme_rect *Theme element: rectangle*

Description

This element draws a rectangular box

Usage

```
theme_rect(fill = NA, colour = "black", size = 0.5, linetype = 1)
```

Arguments

fill	fill colour
colour	border color
size	border size
linetype	border linetype

Details

This is most often used for backgrounds and borders

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

rectGrob for underlying grid function

theme_segment *Theme element: segments*

Description

This element draws segments between a set of points

Usage

```
theme_segment(colour = "black", size = 0.5, linetype = 1)
```

Arguments

colour	line color
size	line size
linetype	line type

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

segmentsGrob for underlying grid function, link{theme_line}

---

theme_text *Theme element: text*

Description

This element adds text

Usage

```
theme_text(family = "", face = "plain", colour = "black", size = 10, hjust = 0.5, v
```

Arguments

family	font family
face	font face ("plain", "italic", "bold")
colour	text colour
size	text size (in pts)
hjust	horizontal justification (in [0, 1])
vjust	vertical justification (in [0, 1])
angle	angle (in [0, 360])
lineheight	line height

Author(s)

Hadley Wickham <h.wickham@gmail.com>

See Also

textGrob for underlying grid function

theme_update *Get, set and update themes.*

Description

These three functions get, set and update themes.

Usage

```
theme_update(...)
```

Arguments

...	named list of theme settings

Details

Use theme_update to modify a small number of elements of the current theme or use theme_set to completely override it.

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
qplot(mpg, wt, data = mtcars)
old <- theme_set(theme_bw())
qplot(mpg, wt, data = mtcars)
theme_set(old)
qplot(mpg, wt, data = mtcars)

old <- theme_update(panel.background = theme_rect(colour = "pink"))
qplot(mpg, wt, data = mtcars)
theme_set(old)
theme_get()

qplot(mpg, wt, data=mtcars, colour=mpg) +
opts(legend.position=c(0.95, 0.95), legend.justification = c(1, 1))
last_plot() +
opts(legend.background = theme_rect(fill = "white", col="white", size =3))
```

update_geom_defaults *Update geom defaults*

Description

Modify geom aesthetic defaults for future plots

Usage

```
update_geom_defaults(geom, new)
```

Arguments

geom	name of geom to modify
new	named list of aesthetics

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
update_geom_defaults("point", list(colour = "darkblue"))
qplot(mpg, wt, data = mtcars)
update_geom_defaults("point", list(colour = "black"))
```

update_stat_defaults *Update geom defaults*

Description

Modify geom aesthetic defaults for future plots

Usage

```
update_stat_defaults(geom, new)
```

Arguments

geom	name of geom to modify
new	named list of aesthetics

Author(s)

Hadley Wickham <h.wickham@gmail.com>

`US economic time series` *US economic time series*

Description

This dataset was produced from US economic time series data available from http://research.stlouisfed.org/fred2.

- date. Month of data collection
- psavert, personal savings rate, http://research.stlouisfed.org/fred2/series/PSAVERT/
- pce, personal consumption expenditures, in billions of dollars, http://research.stlouisfed.org/fred2/series/PCE
- unemploy, number of unemployed in thousands, http://research.stlouisfed.org/fred2/series/UNEMPLOY
- uempmed, median duration of unemployment, in week, http://research.stlouisfed.org/fred2/series/UEMPMED
- pop, total population, in thousands, http://research.stlouisfed.org/fred2/series/POP

Usage

```
data(economics)
```

Format

A data frame with 478 rows and 6 variables

`xlim` *Set x limits*

Description

Convenience function to set the limits of the x axis.

Usage

```
xlim(...)
```

Arguments

`...`	if numeric, will create a continuos scale, if factor or character, will create a discrete scale
	limits

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
xlim(15, 20)
xlim(20, 15)
xlim(c(10, 20))
xlim("a", "b", "c")
qplot(mpg, wt, data=mtcars) + xlim(15, 20)
```

---

ylim *Set y limits*

Description

Convenience function to set the limits of the y axis.

Usage

```
ylim(...)
```

Arguments

...	if numeric, will create a continuos scale, if factor or character, will create a discrete scale
	limits

Author(s)

Hadley Wickham <h.wickham@gmail.com>

Examples

```
ylim(15, 20)
ylim(c(10, 20))
ylim("a", "b", "c")
qplot(mpg, wt, data=mtcars) + ylim(15, 20)
```

Index