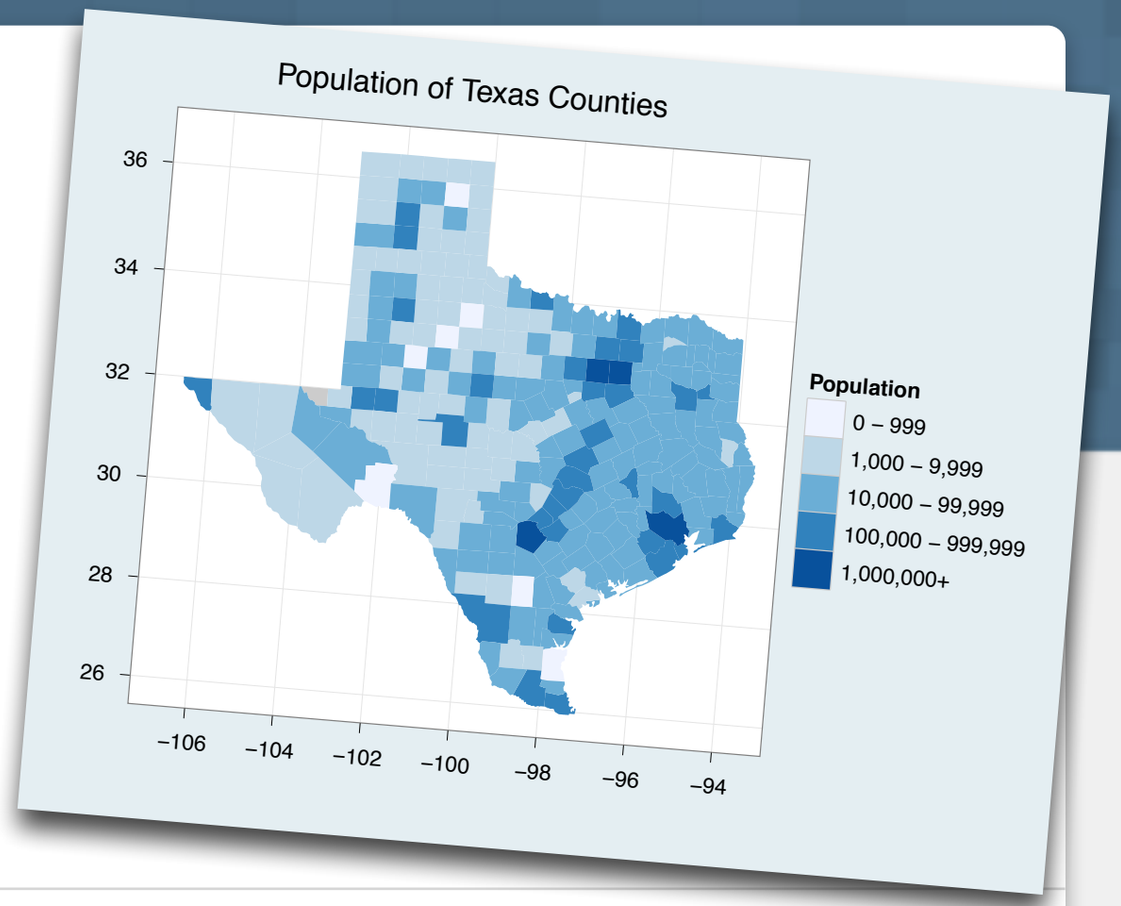
All Training materials are provided "as is" and without warranty and RStudio disclaims any and all express and implied warranties including without limitation the implied warranties of title, fitness for a particular purpose, merchantability and noninfringement.

Customizing Plots

Polish your plots however you please with ggplot2



Garrett Grolemund

Master Instructor, RStudio

April 2014

1. Titles
2. Coordinate Systems
3. Scales
4. Themes
5. Axis labels
6. Legends

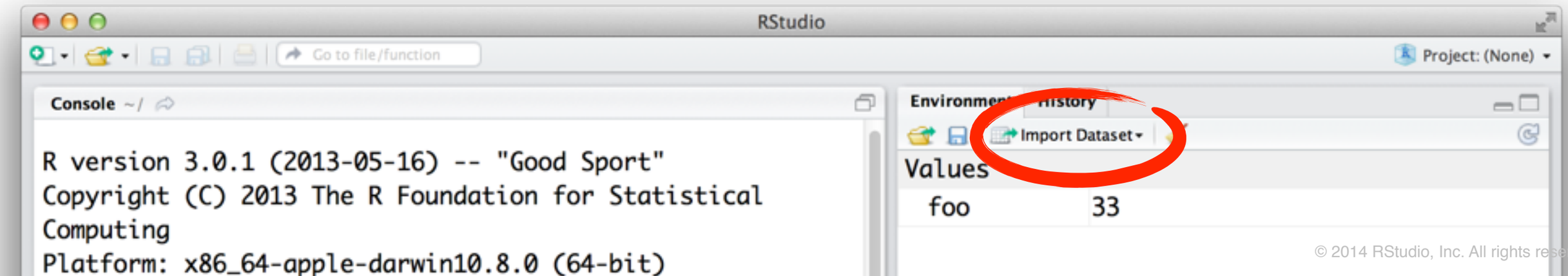
Customizing graphics

Texas population data

We'll use a graph based on this data set as an example. It displays the populations of counties in Texas.

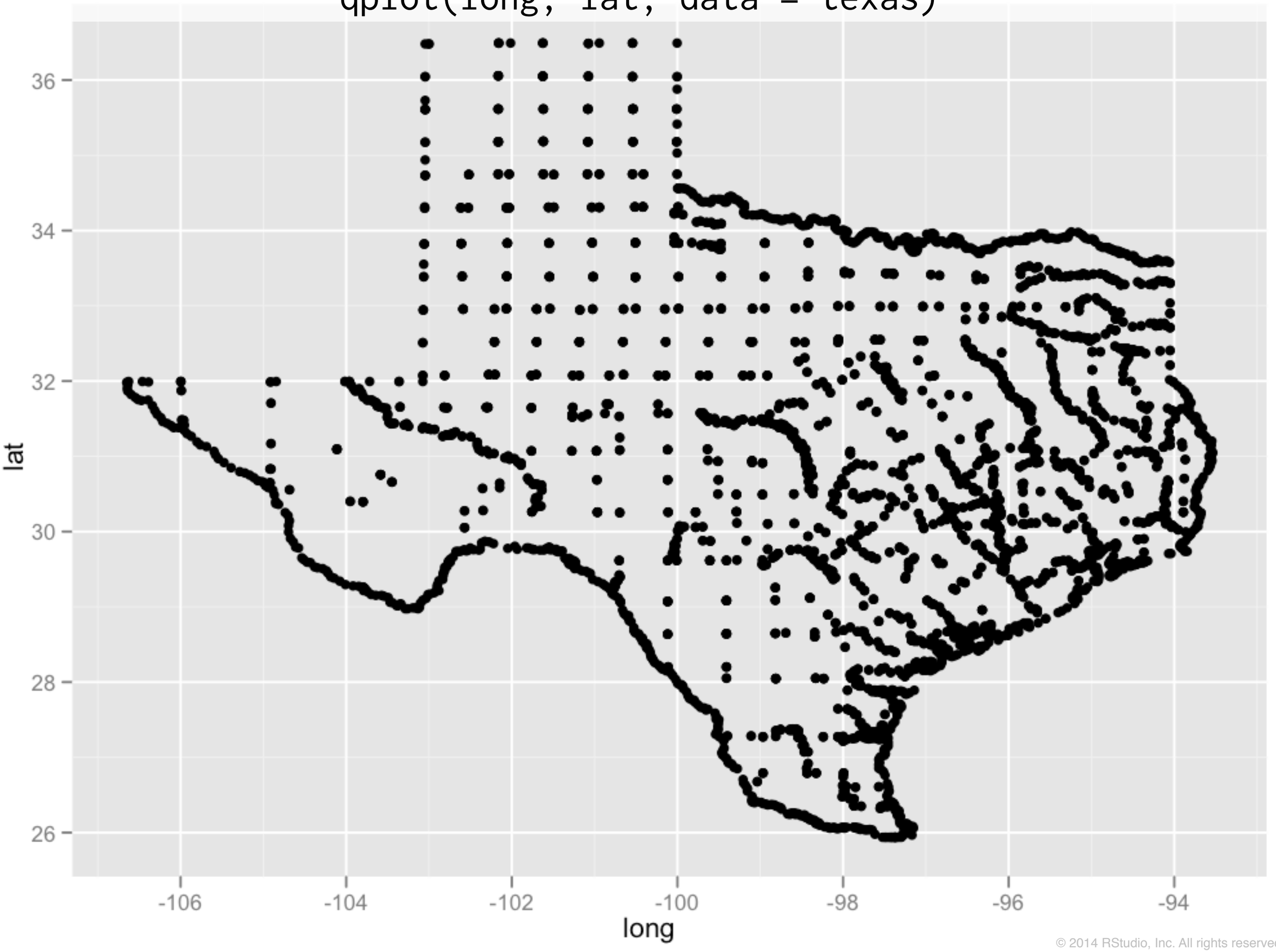
```
texas <- read.csv("data/texas.csv")
View(texas)
```

or

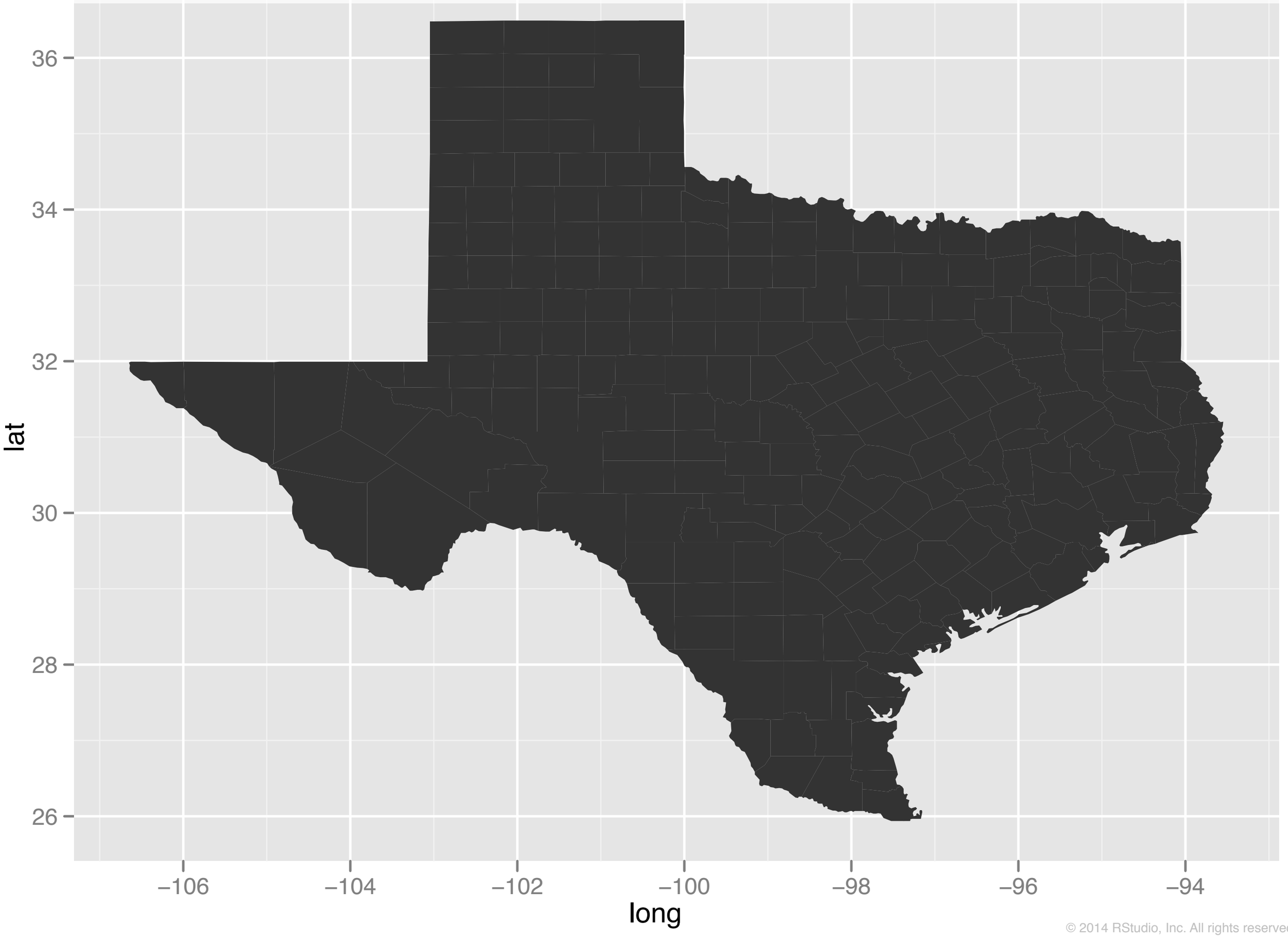


```
head(texas)
#      long      lat group order state   county   pop   bin
# -95.75271 31.53560     1     1 texas anderson 56474 < 1e5
# -95.76989 31.55852     1     2 texas anderson 56474 < 1e5
# -95.76416 31.58143     1     3 texas anderson 56474 < 1e5
# -95.72979 31.58143     1     4 texas anderson 56474 < 1e5
# -95.74698 31.61008     1     5 texas anderson 56474 < 1e5
# -95.72405 31.63873     1     6 texas anderson 56474 < 1e5
```

```
qplot(long, lat, data = texas)
```



```
qplot(long, lat, data = texas, geom = "polygon", group = group)
```



groups points into different polygons

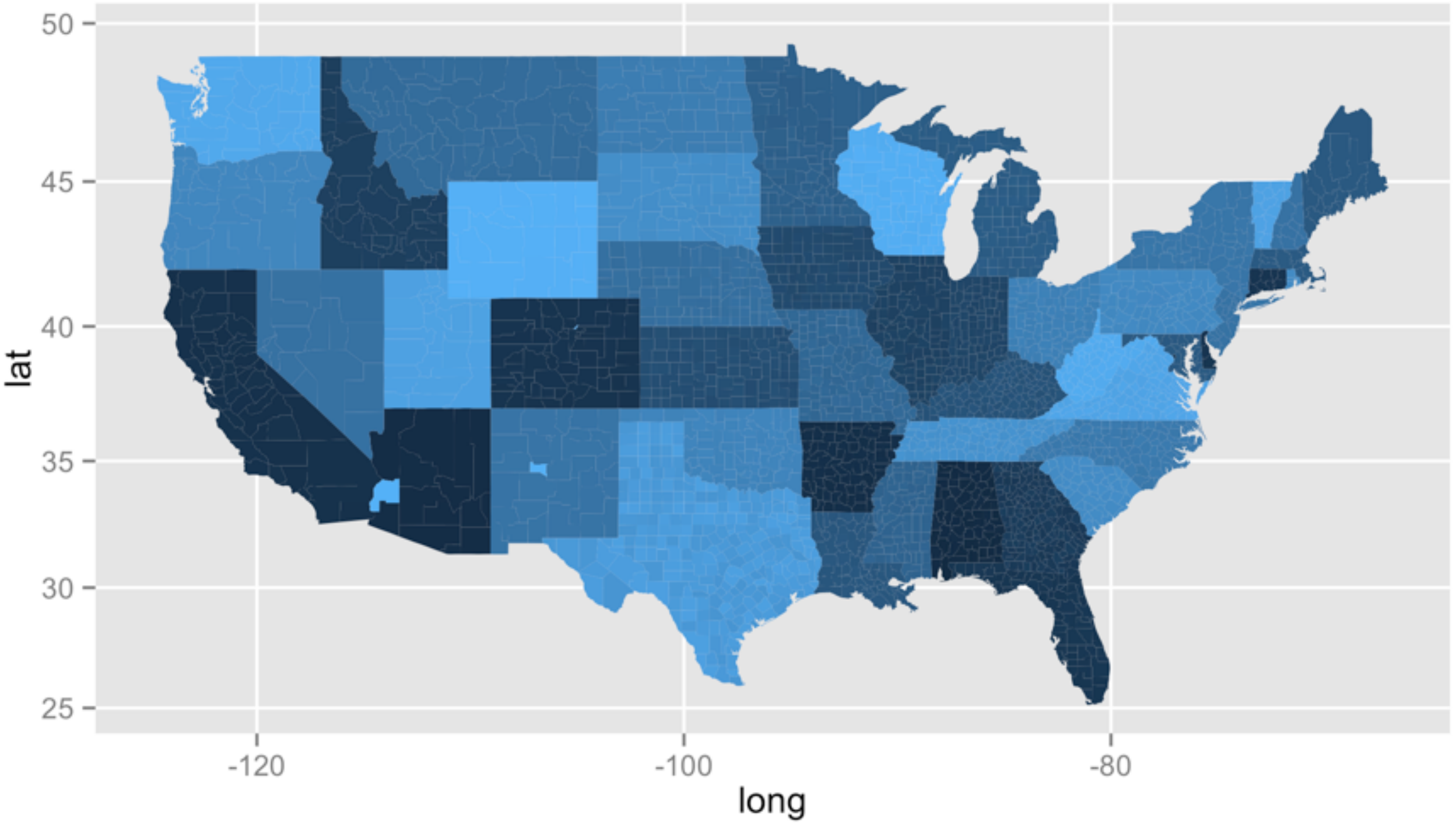
row order matters

```
head(texas)
#      long      lat group order state   county   pop    bin
# -95.75271 31.53560     1     1 texas anderson 56474 < 1e5
# -95.76989 31.55852     1     2 texas anderson 56474 < 1e5
# -95.76416 31.58143     1     3 texas anderson 56474 < 1e5
# -95.72979 31.58143     1     4 texas anderson 56474 < 1e5
# -95.74698 31.61008     1     5 texas anderson 56474 < 1e5
# -95.72405 31.63873     1     6 texas anderson 56474 < 1e5
```

```
texas2 <- borders[sample(nrow(texas)), ]
qplot(long, lat, data = texas2, geom = "polygon", group = group)
```

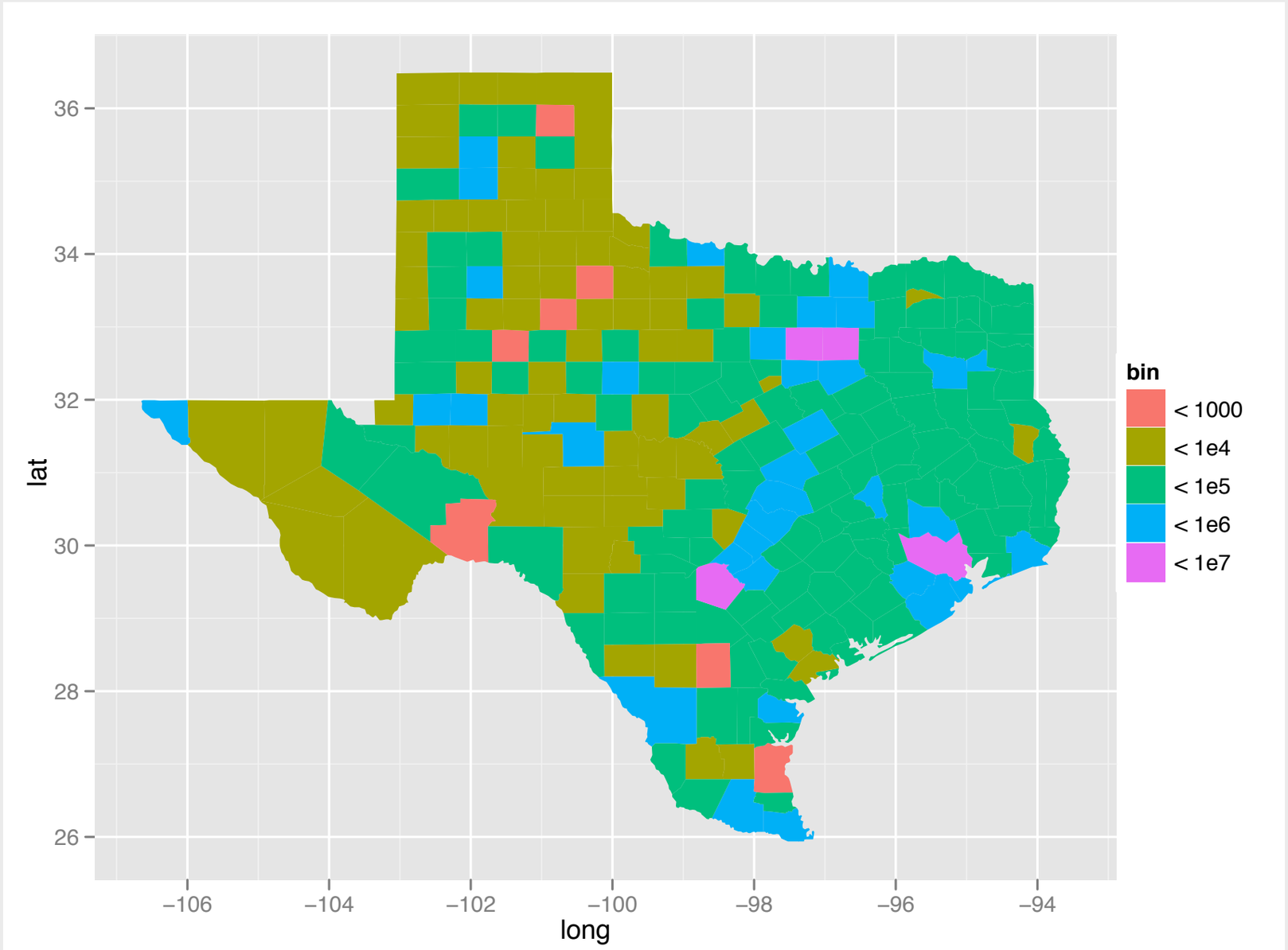
Map data

```
# install.packages("maps")

library(maps)
counties <- map_data("county")

qplot(long, lat, data = counties, geom = "polygon",
  group = group, fill = group)

help(package = "maps")
```



Your Turn

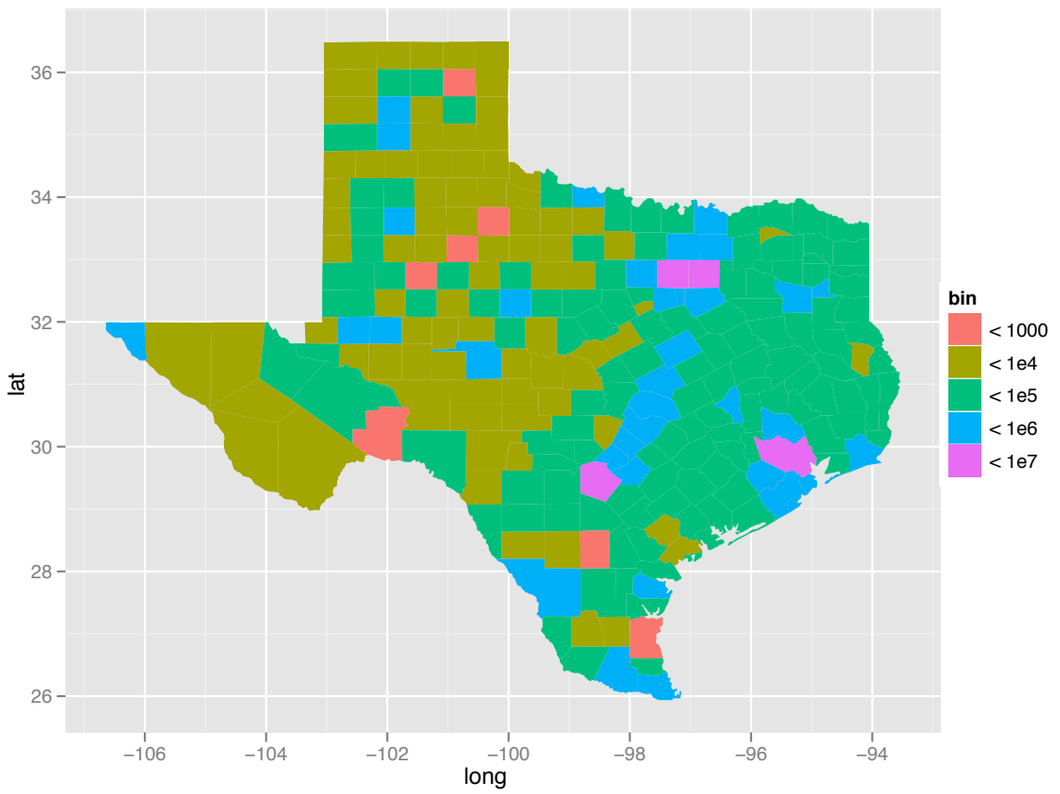
Use `texas` to recreate this map.

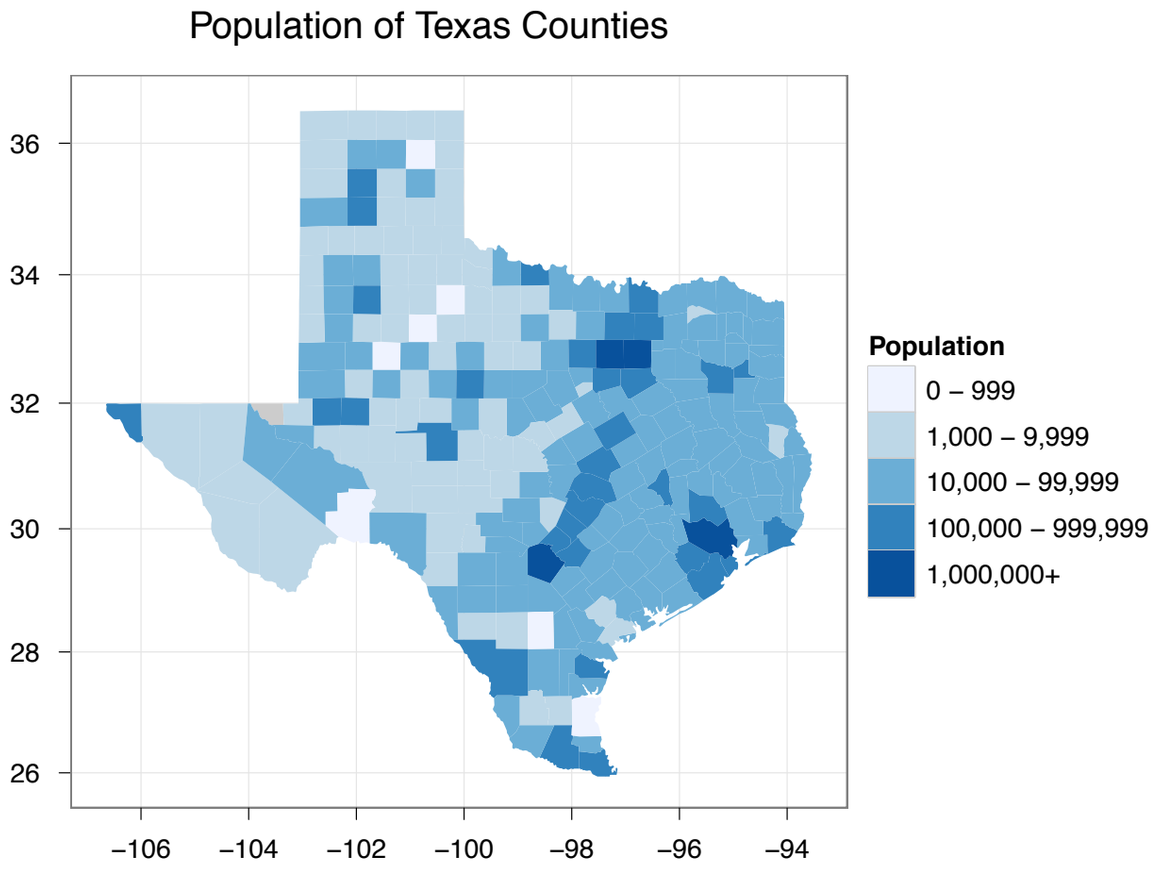


Notice that I'm saving my plot to the object `tx`. You should too. This will make it easy to refer to the plot later.

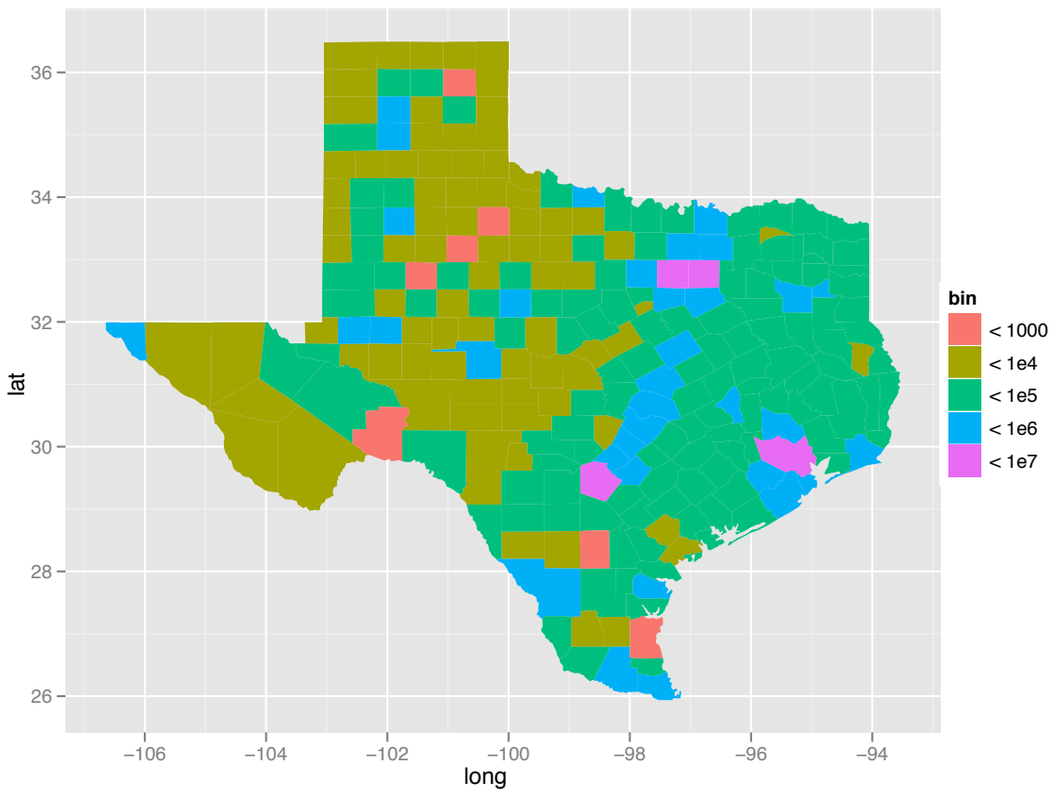
```
tx <- qplot(long, lat, data = texas,
  geom = "polygon", fill = bin, group = group)

  tx
```

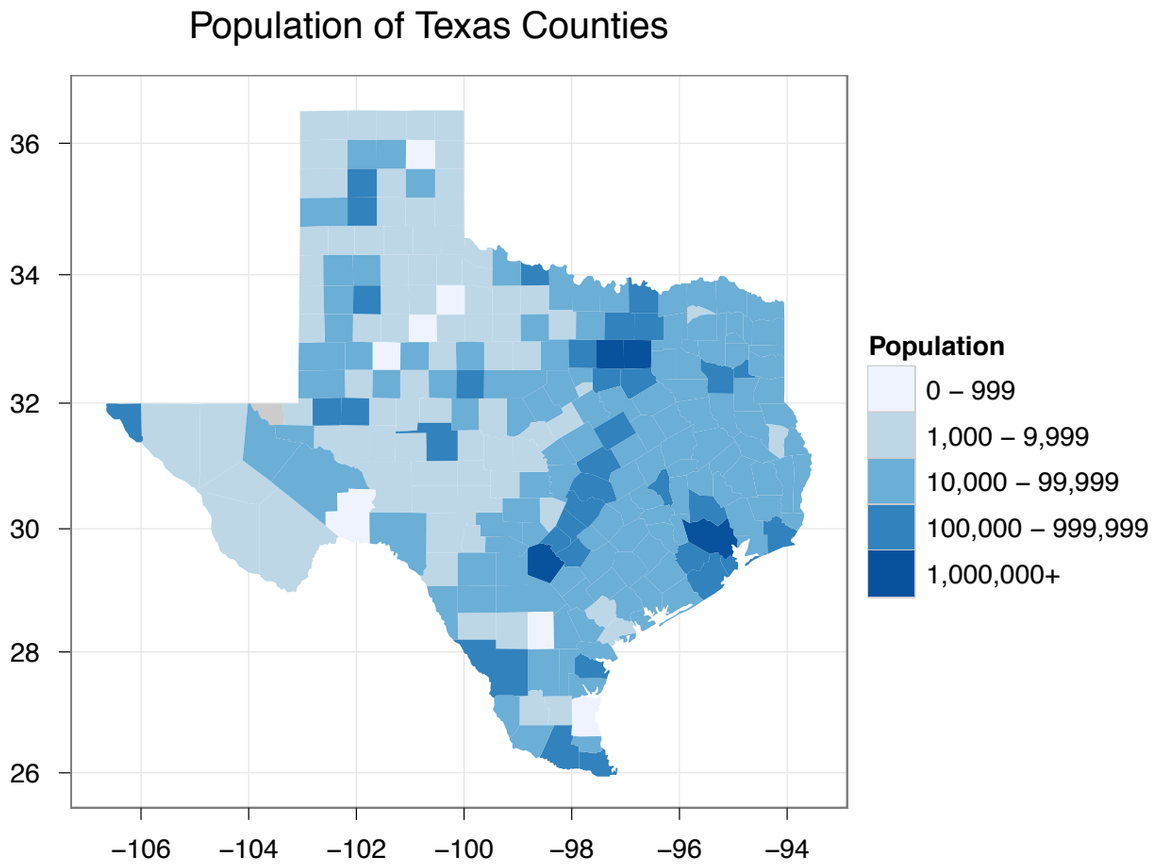
Which do you prefer?





Which do you prefer?





- title
- correct aspect ratio
- color scheme
- white background
- axis labels
- legend labels

Our goal

Title

ggtitle

You modify ggplot2 graphs by adding objects to them.

```
tx + ggtitle("Population of Texas Counties")
```

Creates a ggplot2 title

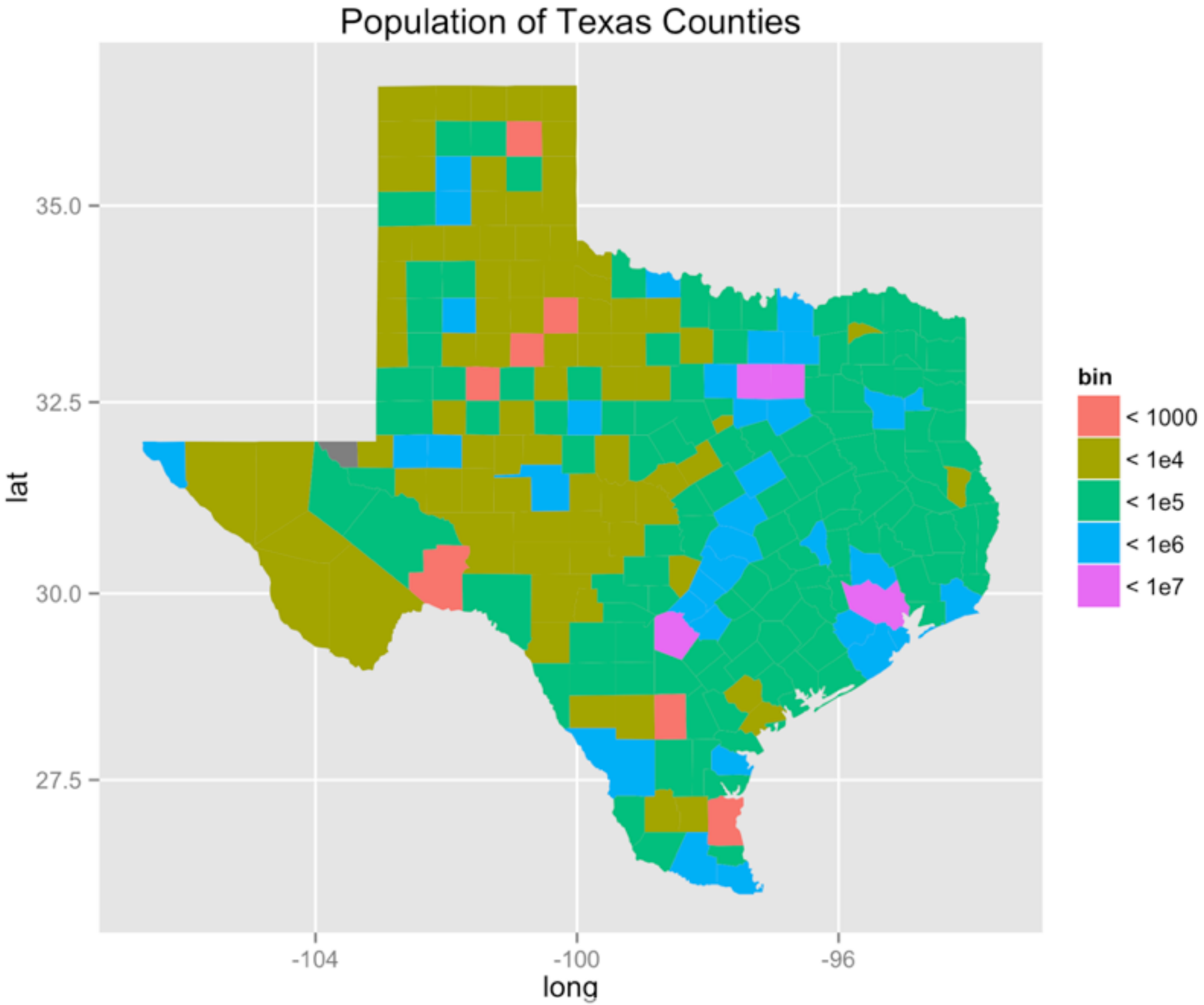
ggtitle

You modify ggplot2 graphs by adding objects to them.

```
tx + ggtitle("Population of Texas Counties")
```

Adds it to the graph

Creates a ggplot2 title




```
tx + ggtitle("Population of Texas Counties")
```

Additions are not permanent!

```
tx + ggtitle("Population of Texas Counties")

tx
```

```
# to create a new graph that always has a title

tx2 <- tx + ggtitle("Population of Texas Counties")

tx2
```

Manipulating plots

What is in a ggplot object?

```
cp <- qplot(carat, price, data = diamonds)
class(cp)
# "gg"     "ggplot"
```

```
str(cp)
# List of 9
#  $ data       :'data.frame':    53940 obs. of  10 variables:
#   ..$ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
#   ..$ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
#   ..$ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
#   ..$ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
#   ..$ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
#   ..$ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
#   ..$ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
#   ..$ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
#   ..$ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
#   ..$ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
#  $ layers     :List of 1
#   ..$ :Classes 'proto', 'environment' <environment: 0x7fcf5d08da38>
#  $ scales     :Reference class 'Scales' [package "ggplot2"] with 1 fields
#   ..$ scales: list()
#   ..and 21 methods, of which 9 are possibly relevant:
#   ..  add, clone, find, get_scales, has_scale, initialize, input, n, non_position_scales
#  $ mapping    :List of 2
#   ..$ x: symbol carat
#   ..$ y: symbol price
#  $ theme      : list()
#  $ coordinates:List of 1
#   ..$ limits:List of 2
#   .. ..$ x: NULL
#   .. ..$ y: NULL
#   ..- attr(*, "class")= chr [1:2] "cartesian" "coord"
#  $ facet      :List of 1
#   ..$ shrink: logi TRUE
#   ..- attr(*, "class")= chr [1:2] "null" "facet"
#  $ plot_env   :<environment: R_GlobalEnv>
#  $ labels     :List of 2
#   ..$ x: chr "carat"
#   ..$ y: chr "price"
#  - attr(*, "class")= chr [1:2] "gg" "ggplot"#
```

We are going to override things that are already here as defaults

Coordinate systems

coordinate systems

Determine the coordinate plane to draw the graph on.

```
cp$coordinates
# $limits
# $limits$x
# NULL

# $limits$y
# NULL

# attr(,"class")
# [1] "cartesian" "coord"
```

Always begins with coord_

coordinate system's name

open and closed parentheses

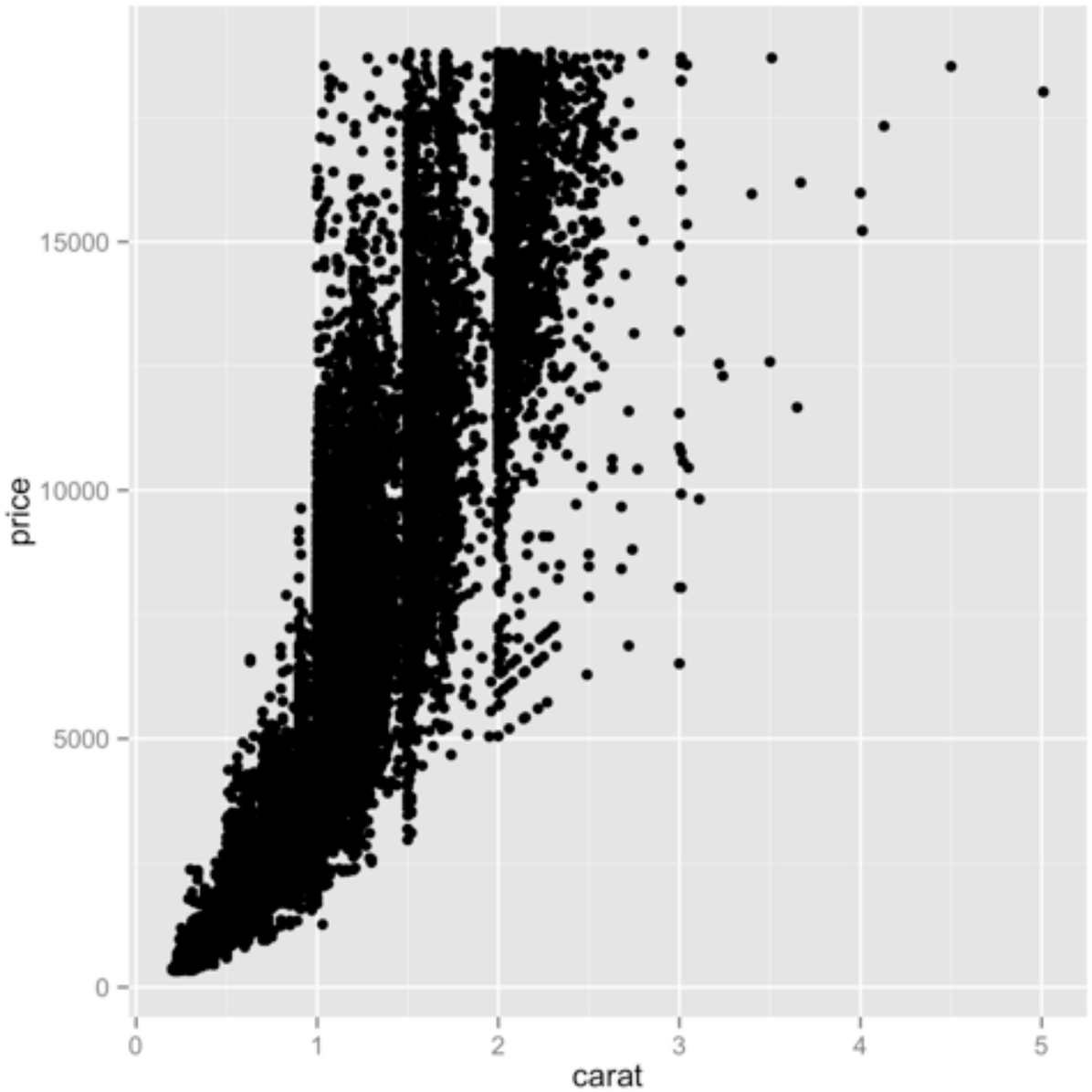
```
coord_cartesian()
```

cp2 <- cp + coord_polar()

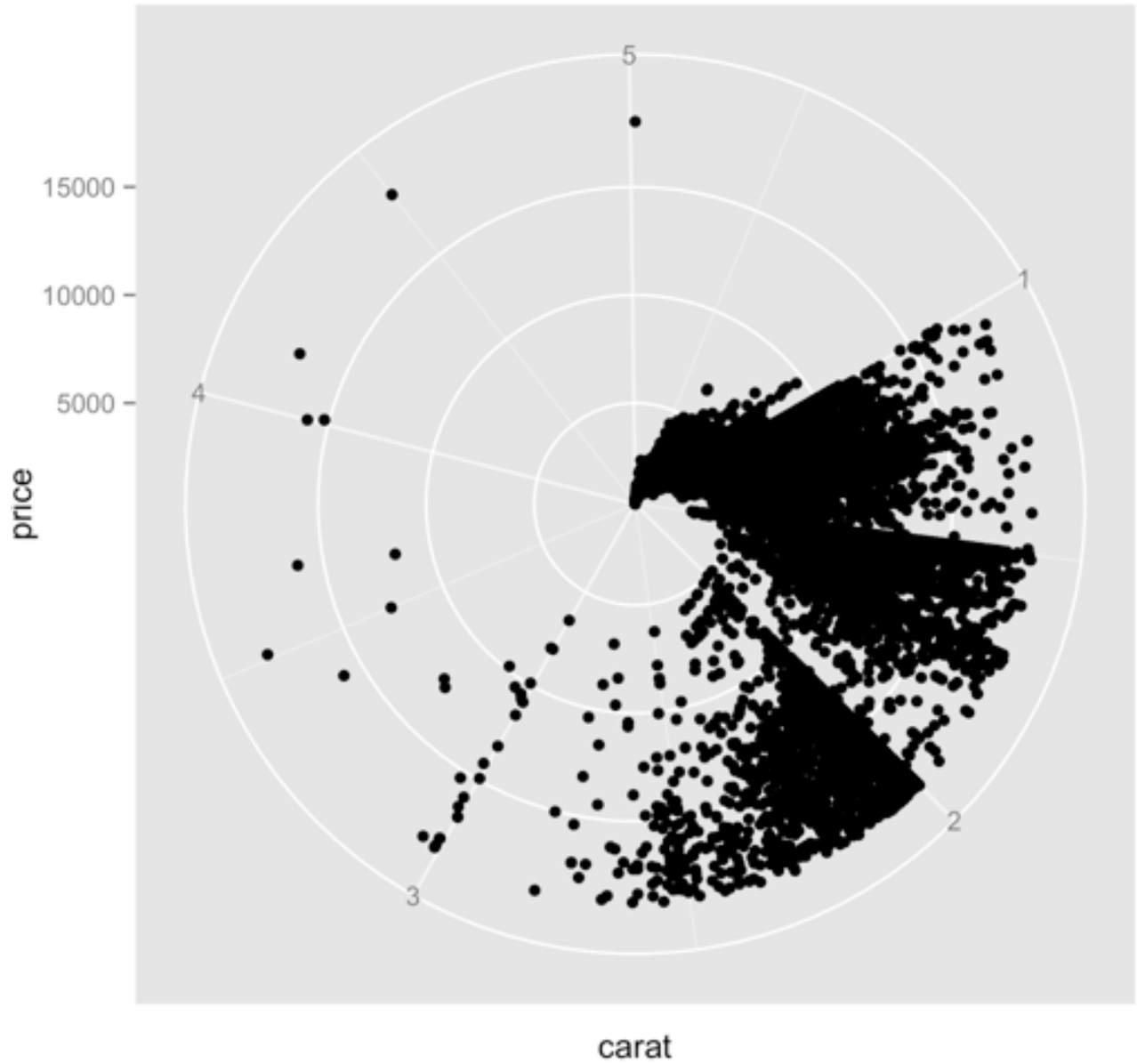
```
cp$coordinates
# $limits
# $limits$x
# NULL

# $limits$y
# NULL

# attr(,"class")
# [1] "cartesian"
"coord"
```

```
cp2$coordinates
# $theta
# [1] "x"

# $r
# [1] "y"

# $start
# [1] 0

# $direction
# [1] 1

# attr(,"class")
# [1] "polar" "coord"
```

polar

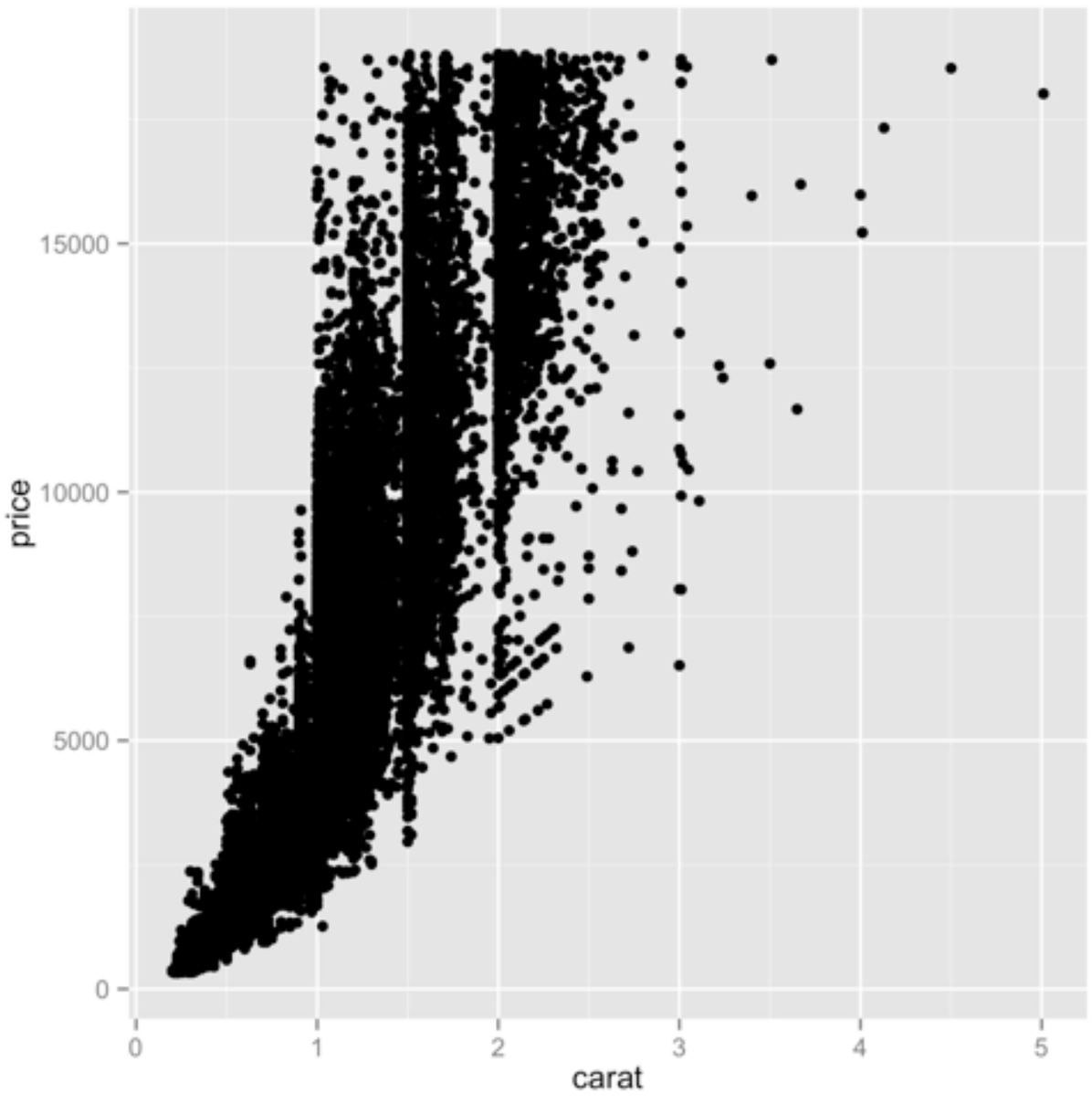


cp

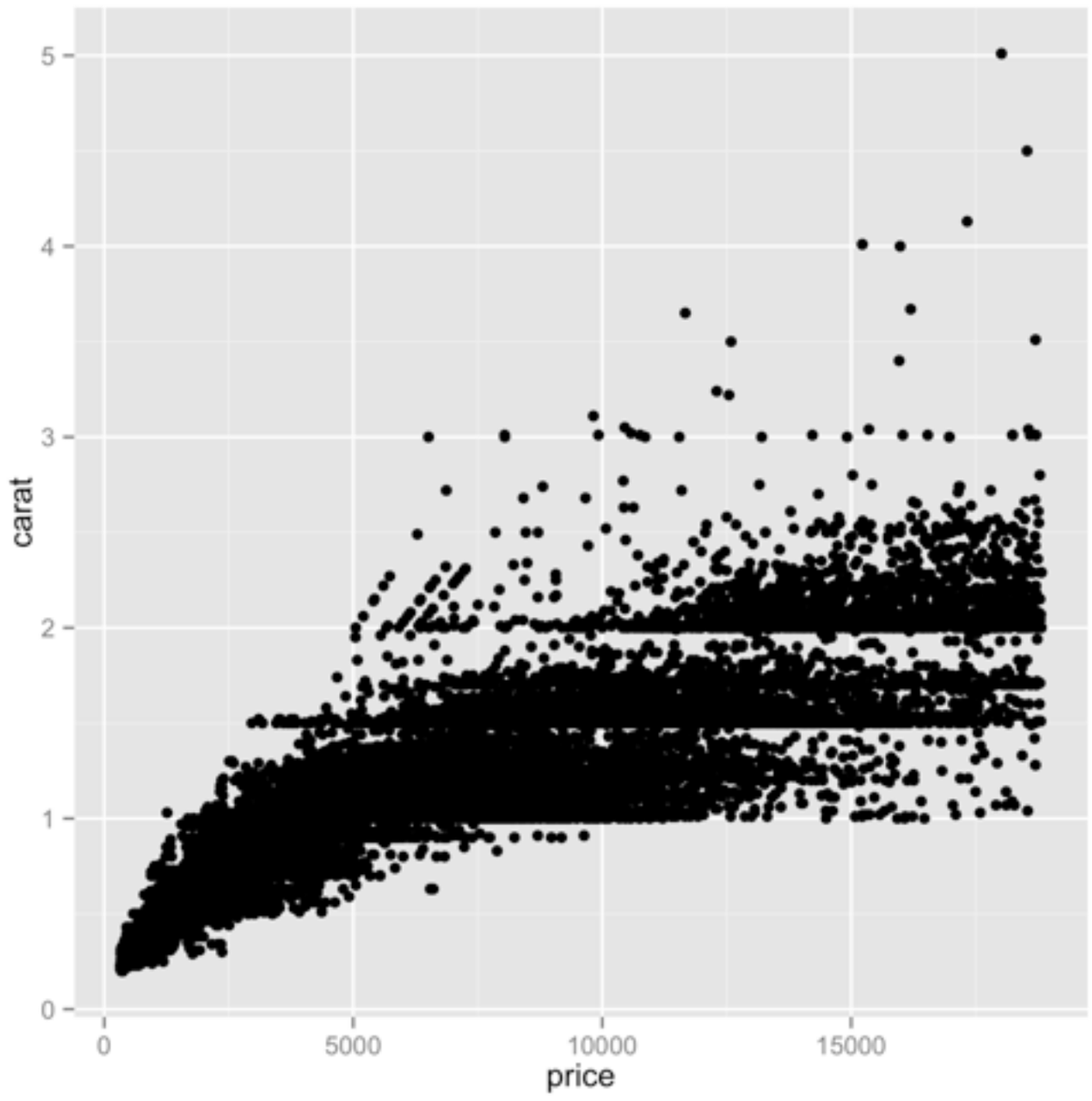


cp + coord_polar()

flip

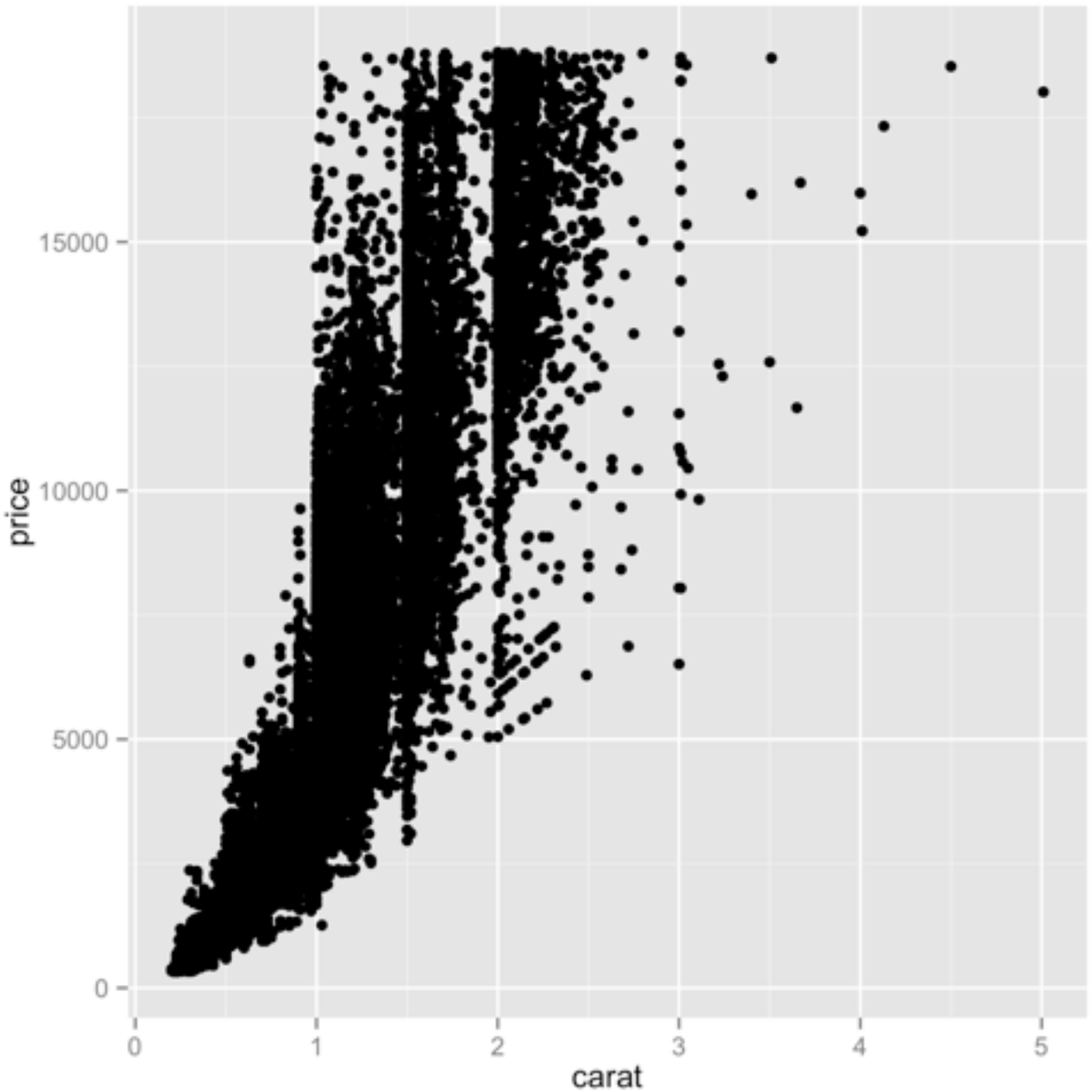


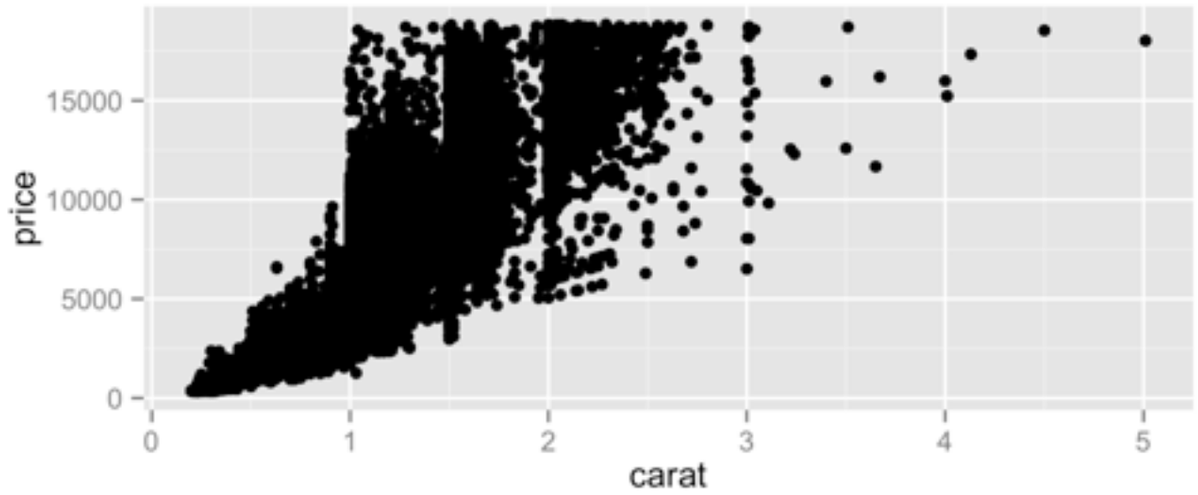
cp




```
cp + coord_flip()
```

fixed

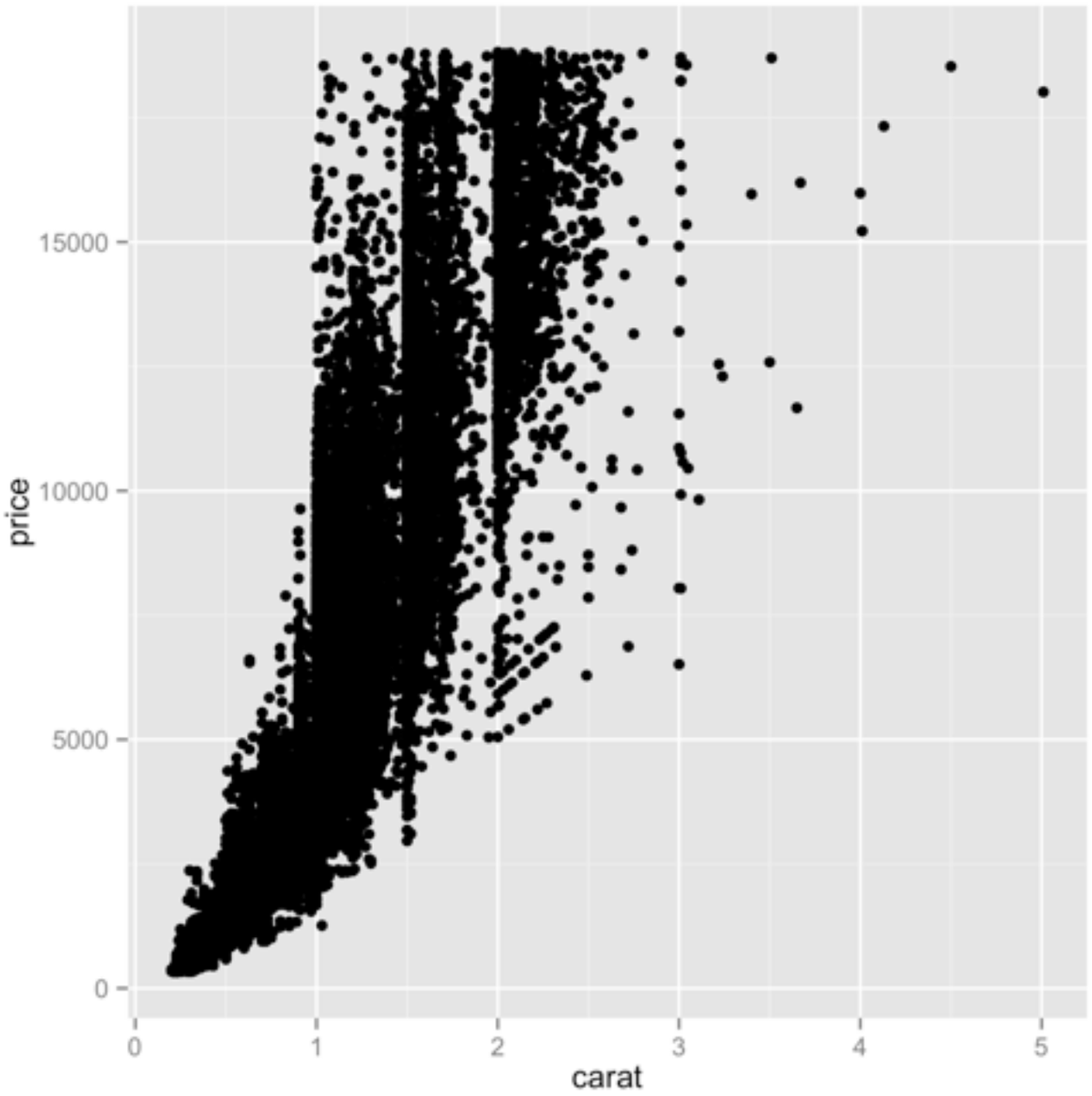


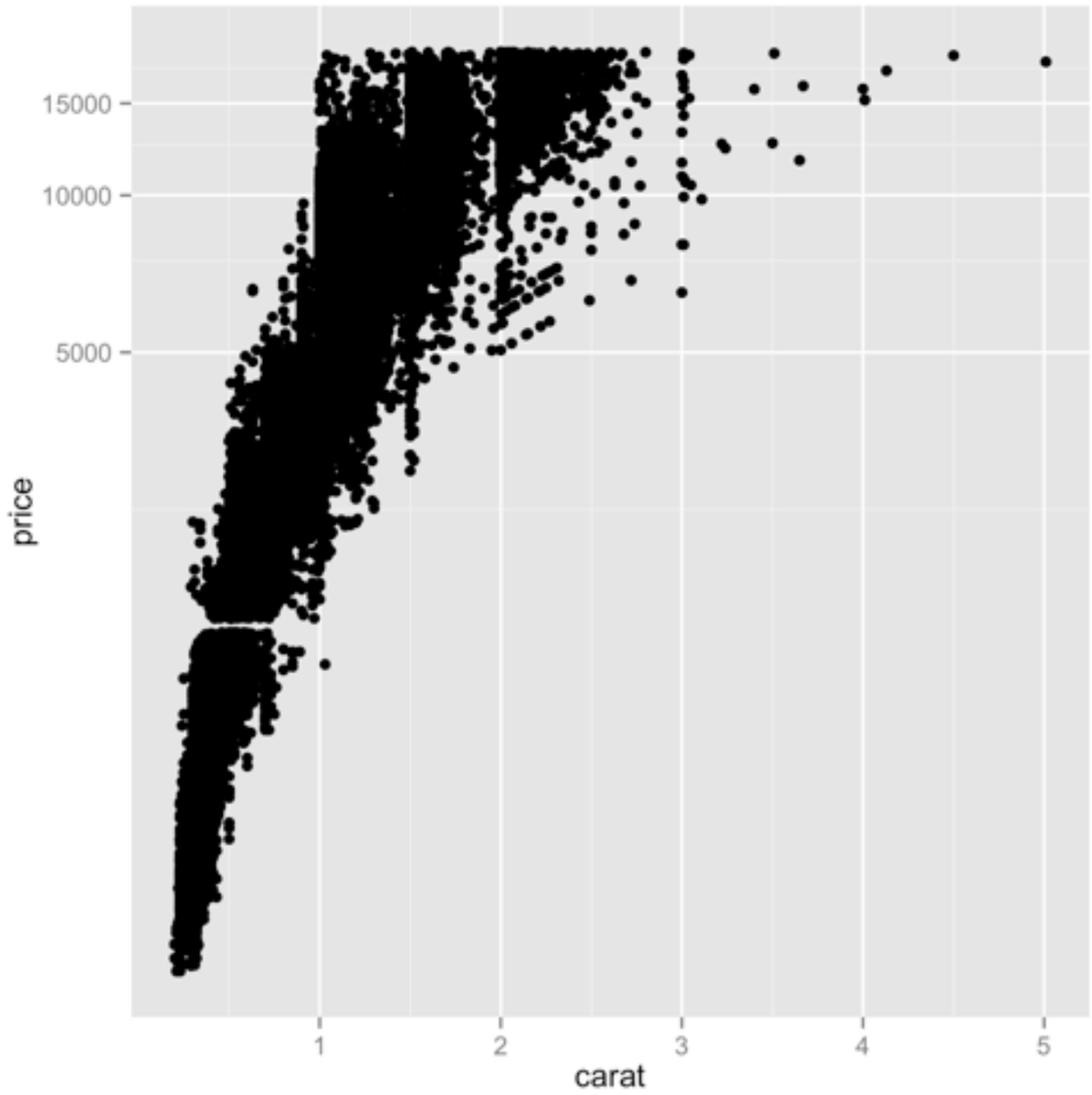



```
cp
```

```
cp + coord_fixed(ratio = 1/10000)
```

trans

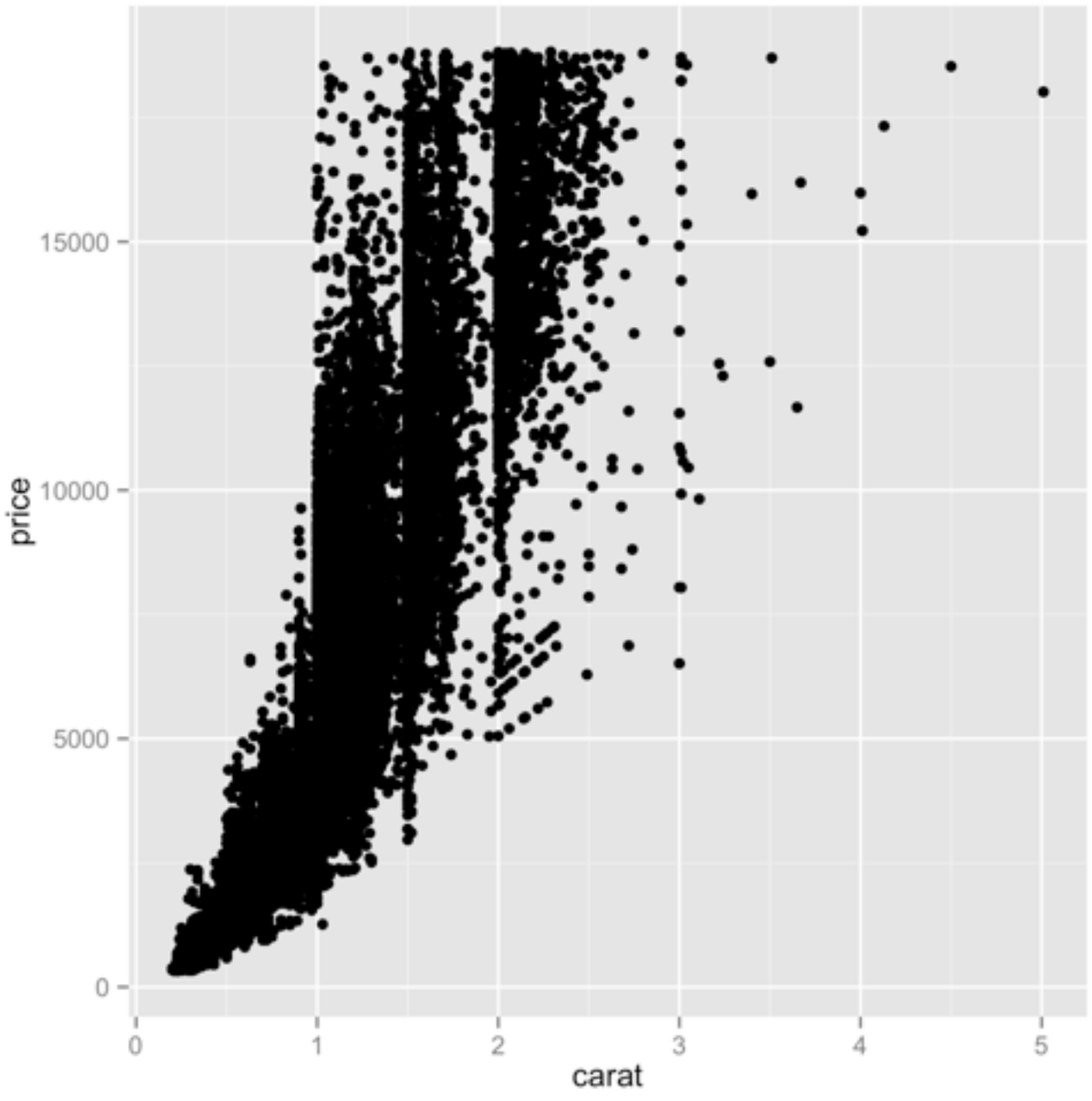


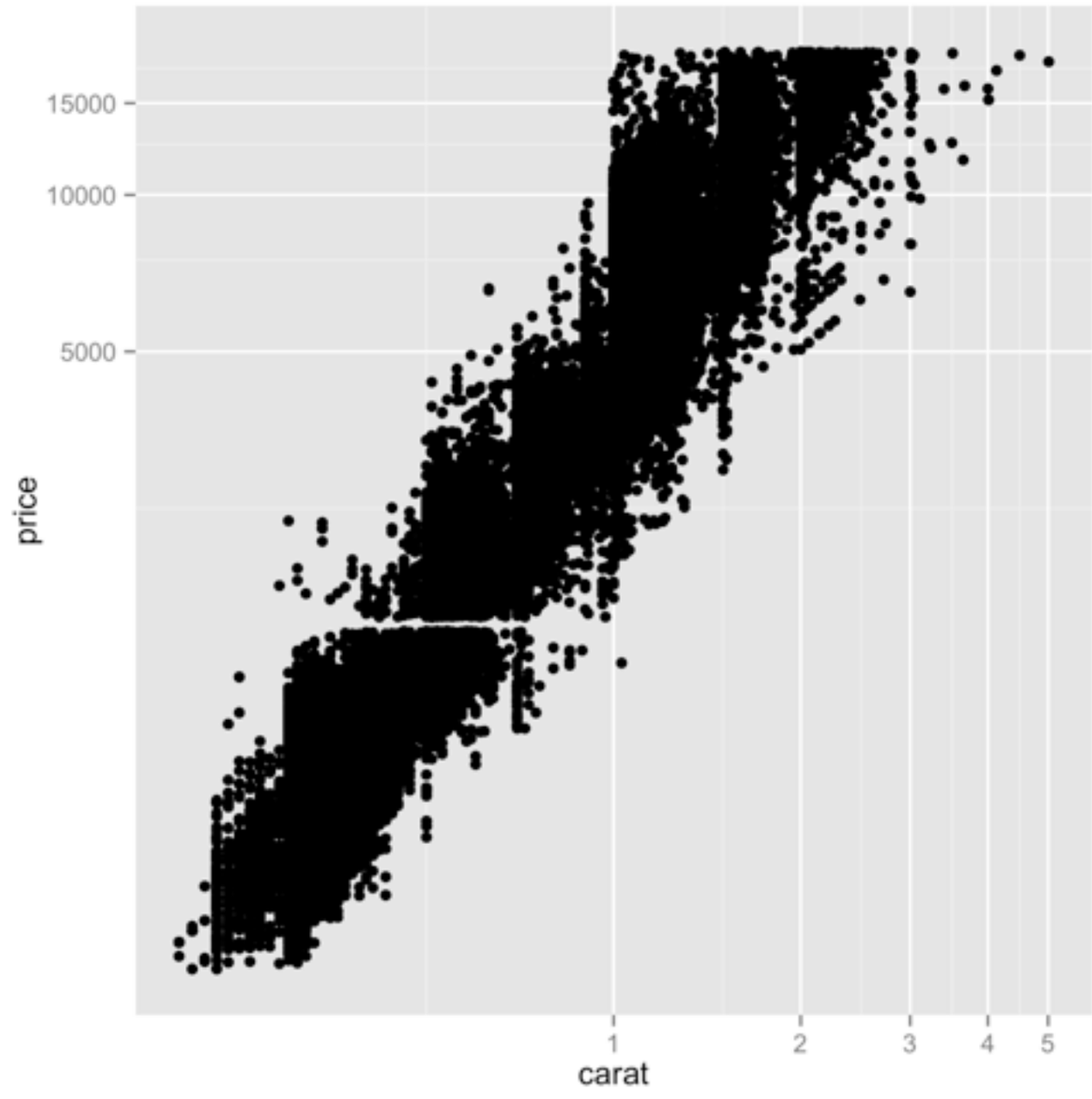



```
cp
```

```
cp + coord_trans(ytrans = "log10")
```

trans

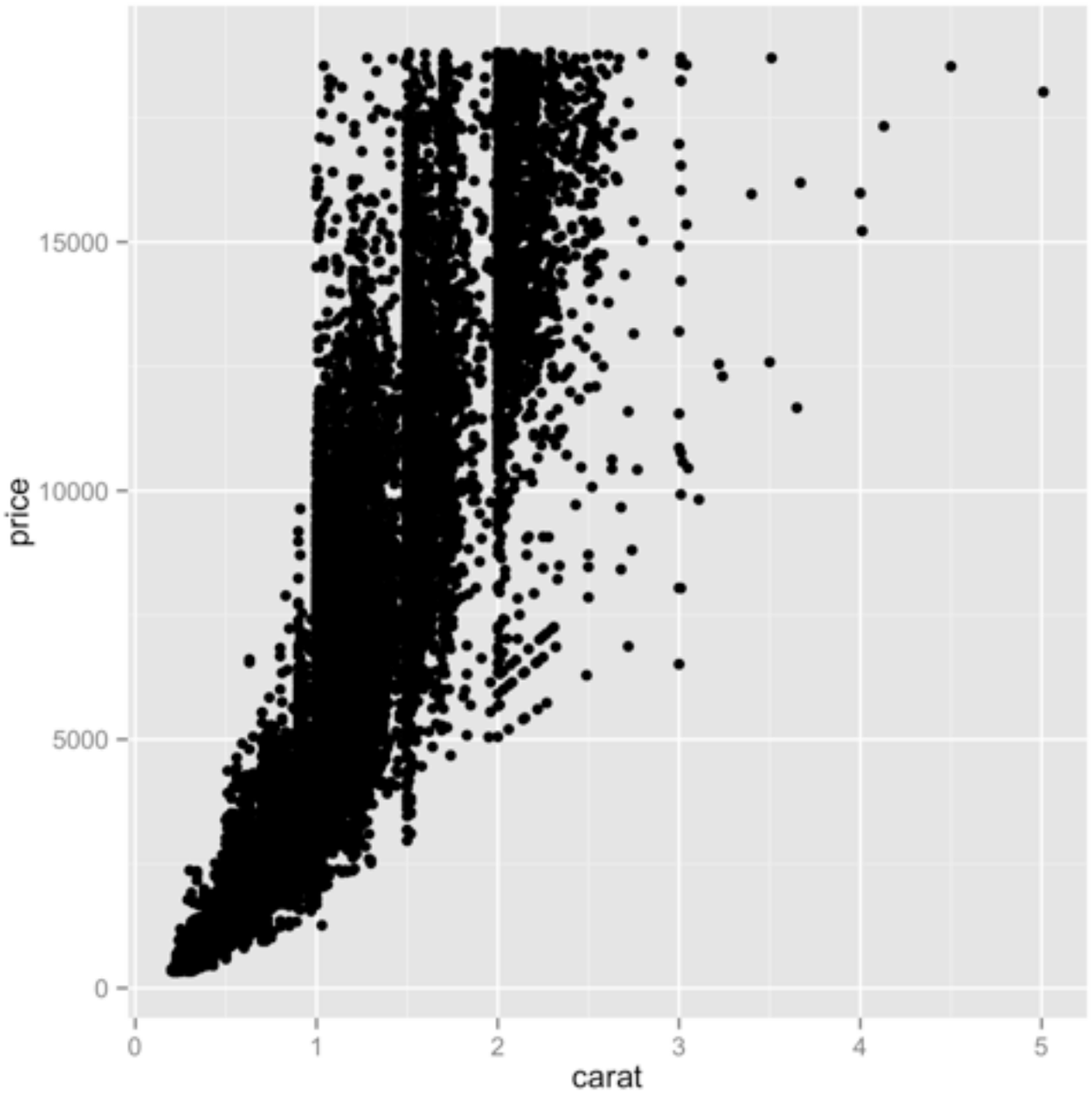




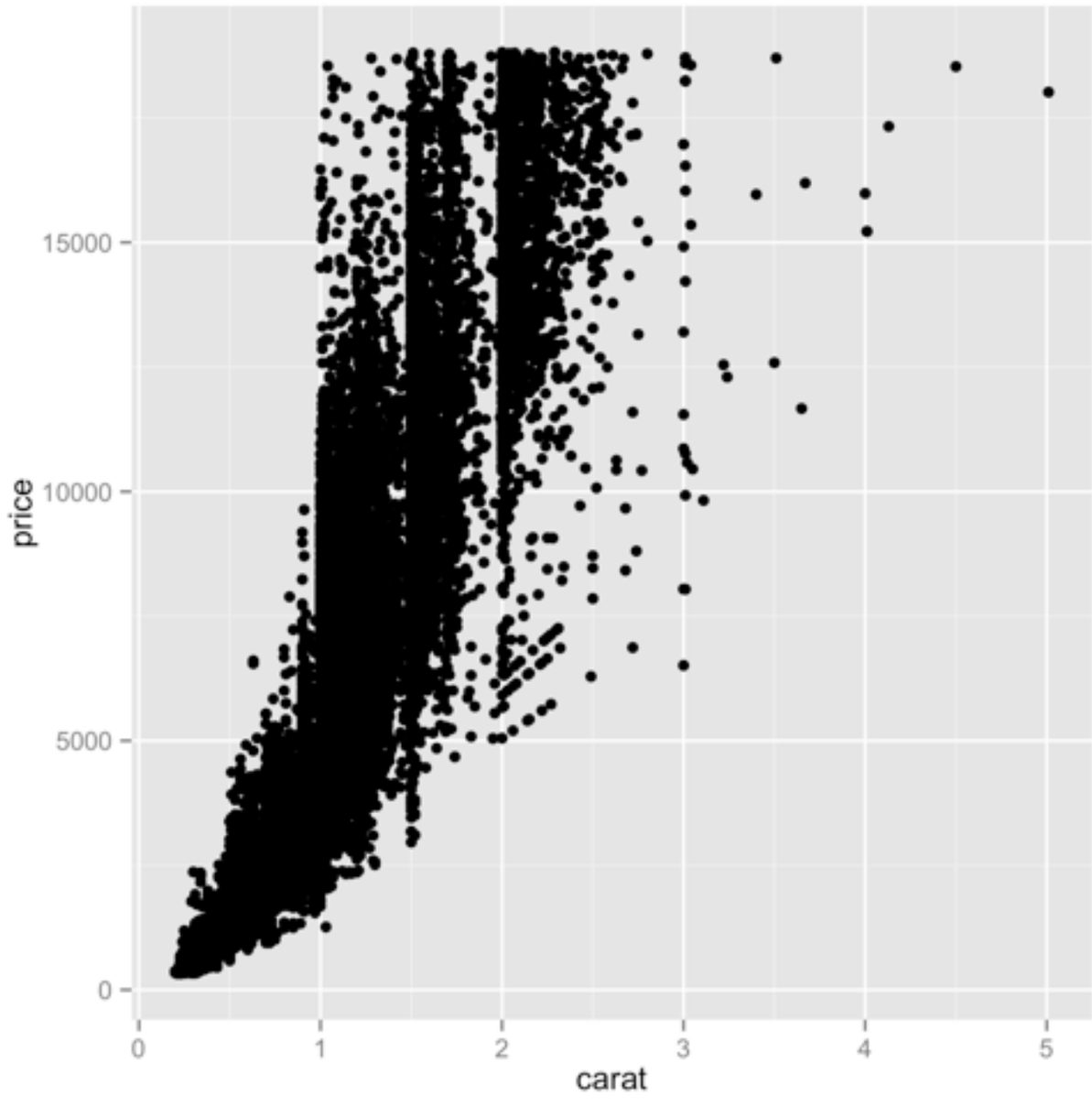

```
cp
```

```
cp + coord_trans(ytrans = "log10",
                 xtrans = "log10")
```

cartesian (default)

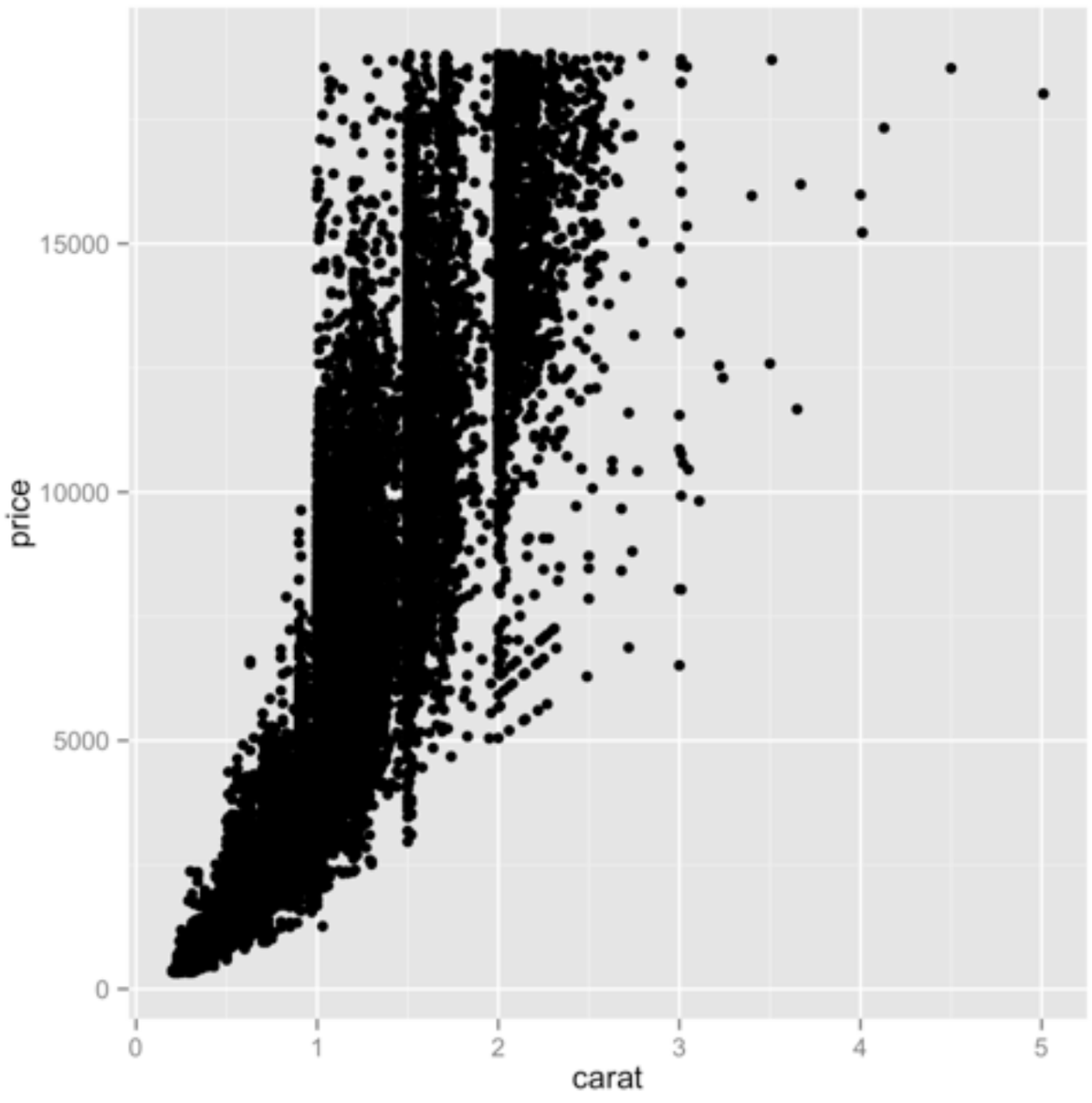


cp

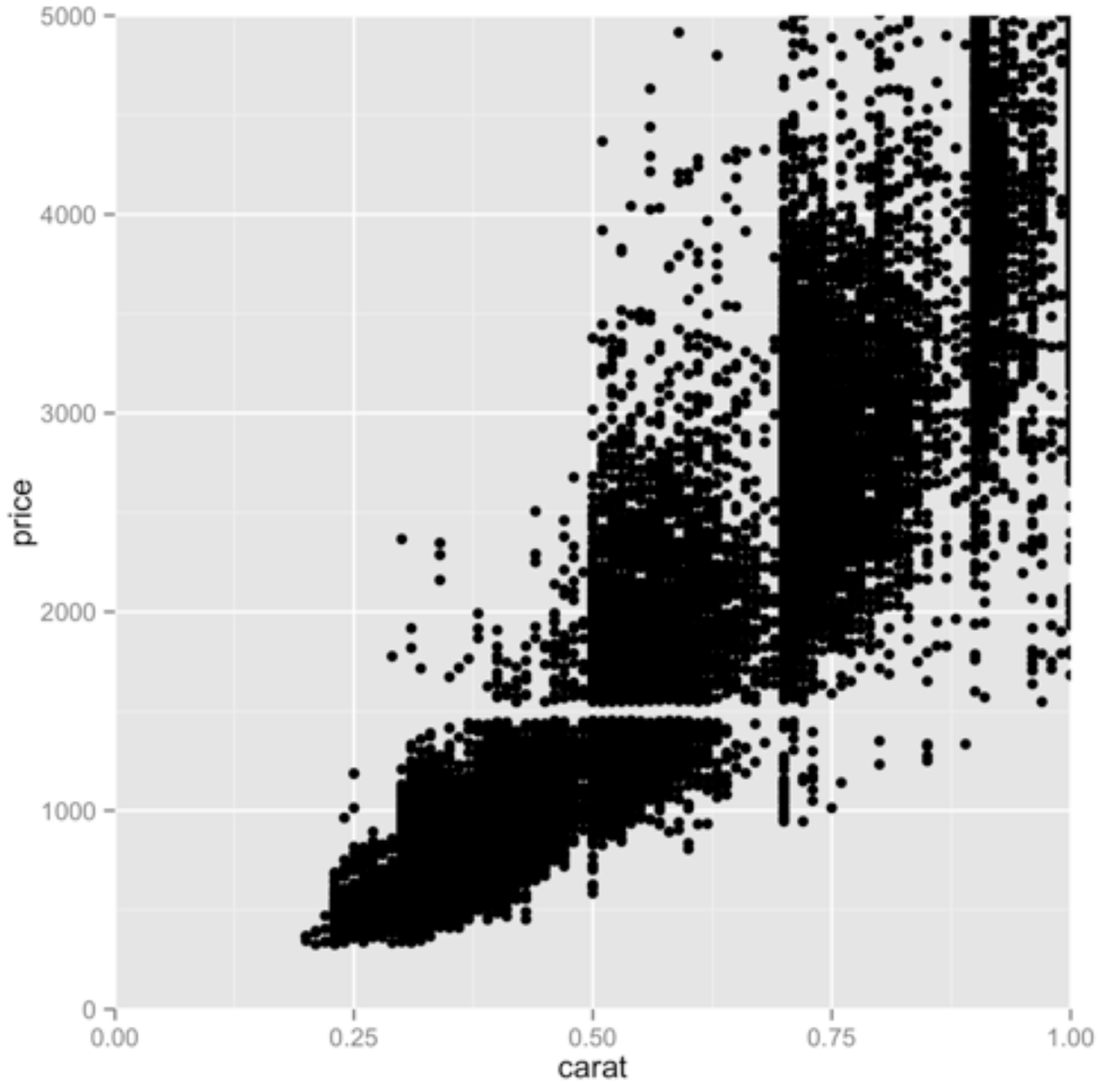



```
cp + coord_cartesian()
```

cartesian (to zoom)



cp

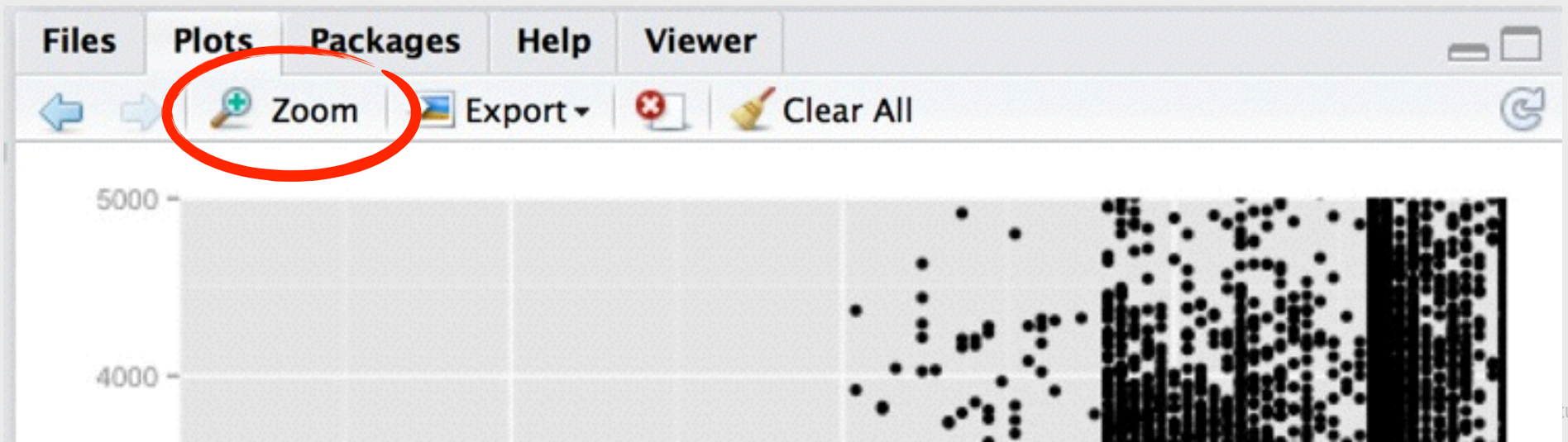



```
cp + coord_cartesian(
  ylim = c(0,5000), xlim = c(0, 1))
```

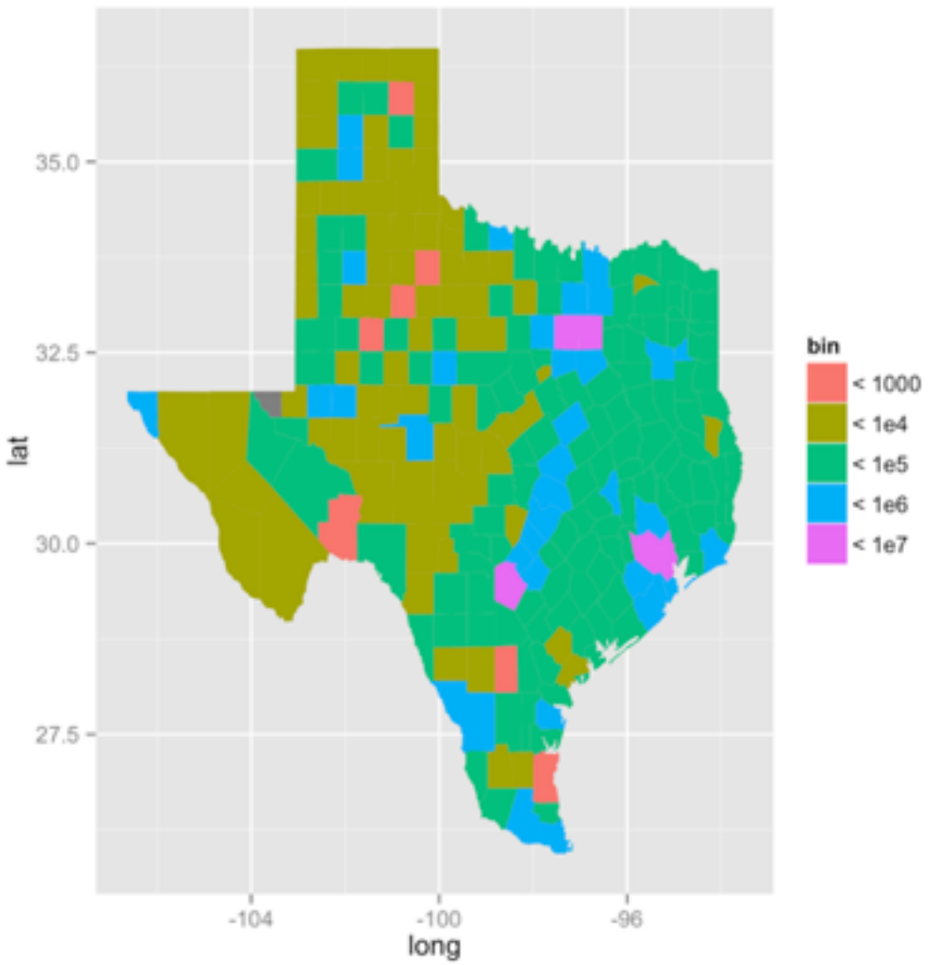
Your Turn

There is also a `coord_map`. Add it to `tx` and then open the plot in a new window by hitting zoom.

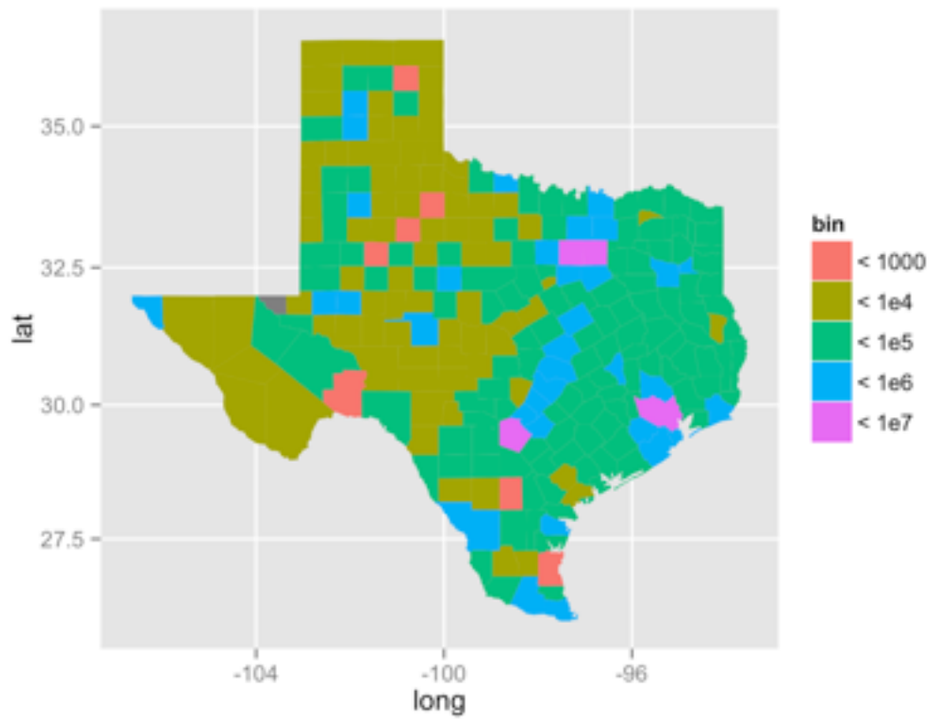
Try rescaling the window. Can you tell what `coord_map` does?



A plot followed by coord_map will always display in a mercator projection (or whichever projection you set).



tx

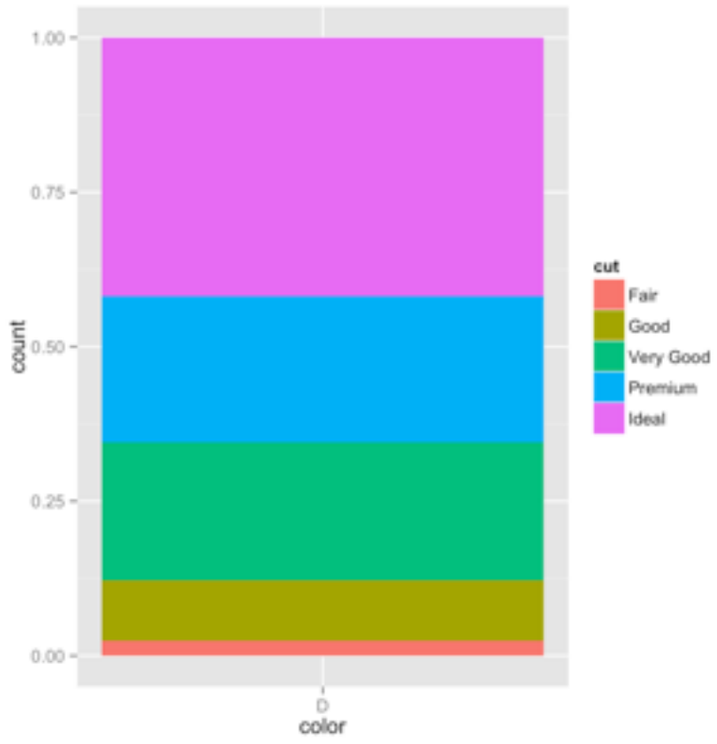


tx + coord_map()

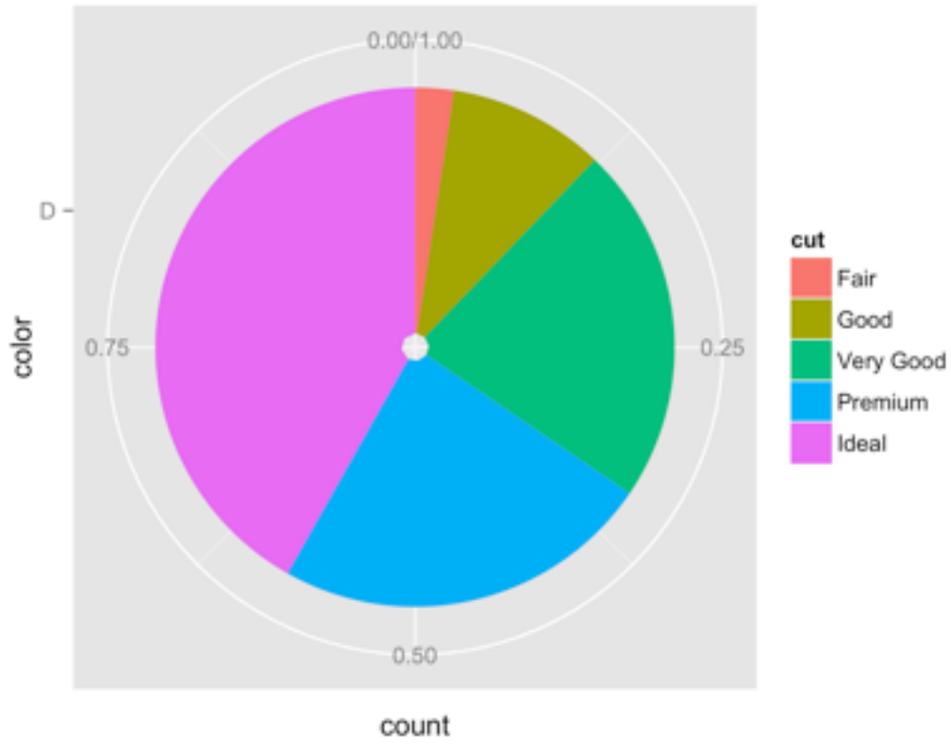
Pie charts

Aside: In the grammar of graphics, a pie chart is a stacked bar graph in polar coordinates.

```
d2 <- subset(diamonds, color == "D")
cc <- qplot(color, data = d2, fill = cut, position = "fill")
```



cc



cc + coord_polar(theta = "y")

Scales

scales

Determine how data is mapped to an aesthetic

```
hd <- qplot(displ, hwy, data =mpg,
      color = class, shape = class)
```

Aesthetic mapping

What variable to map to color

Scale

How to map the variable to color

Always begins with scale_

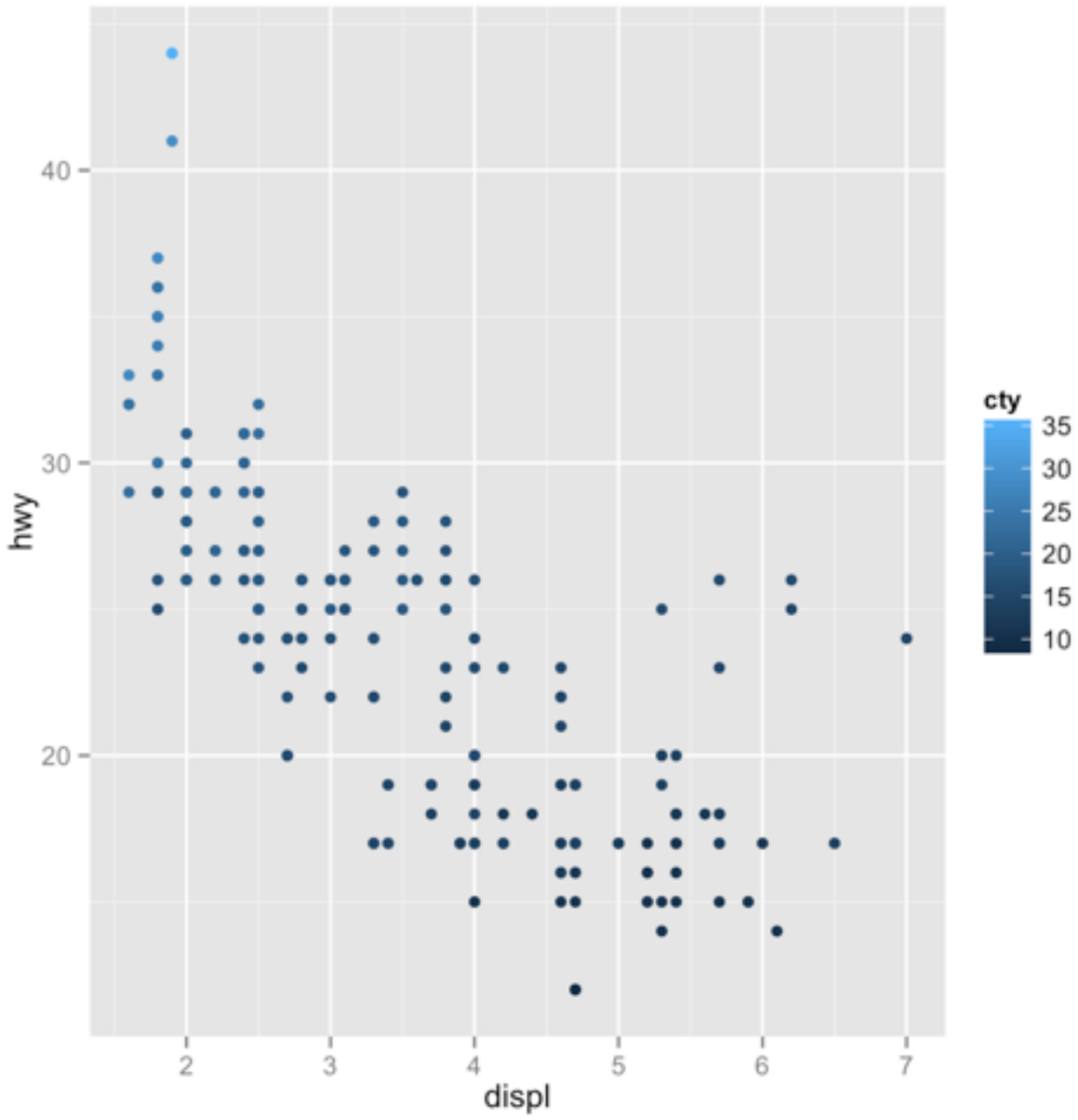
aesthetic to adjust

name of a scale object in ggplot2

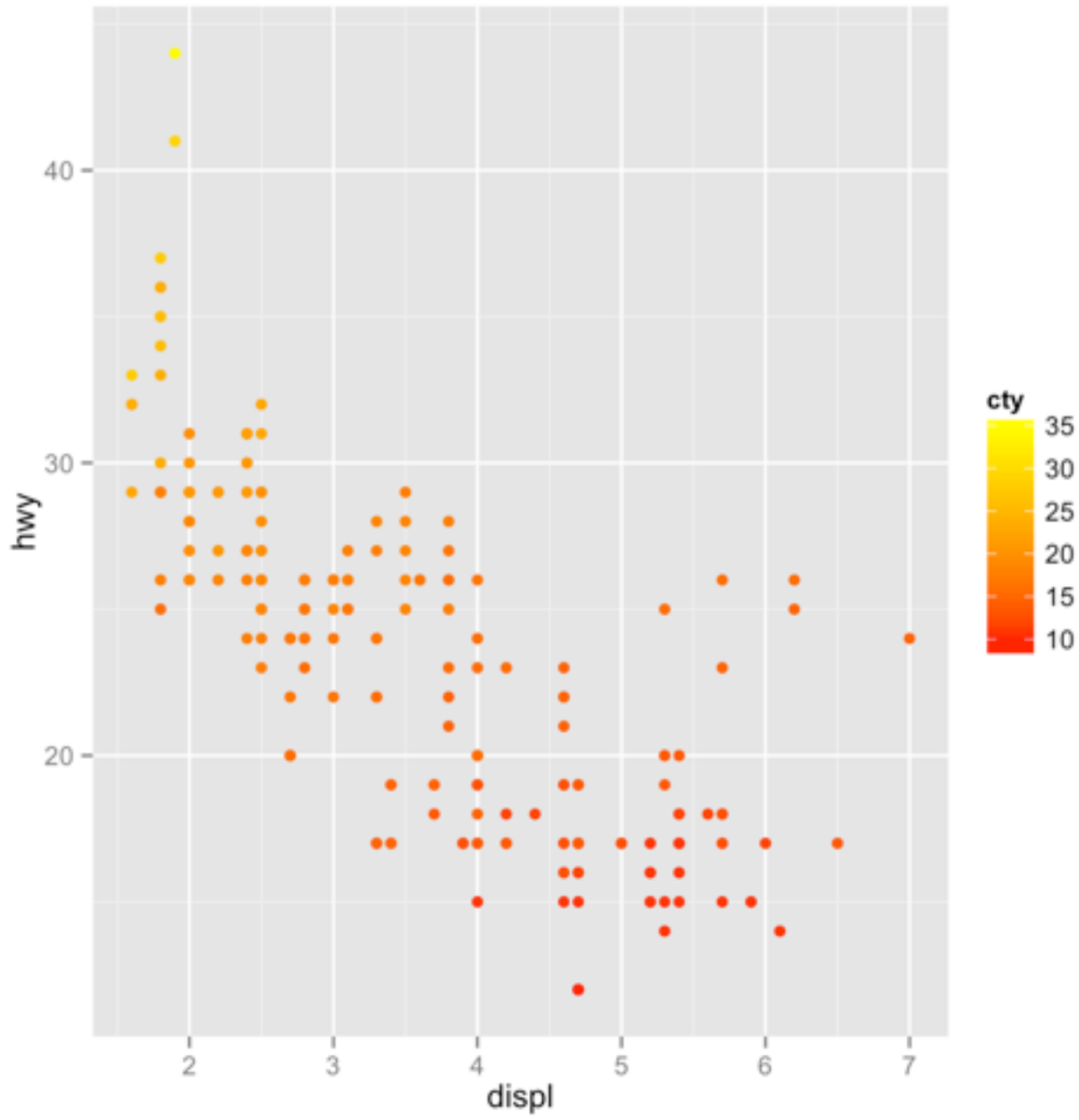
open and closed parentheses

```
scale_aesthetic_name()
```

scale_color_gradient

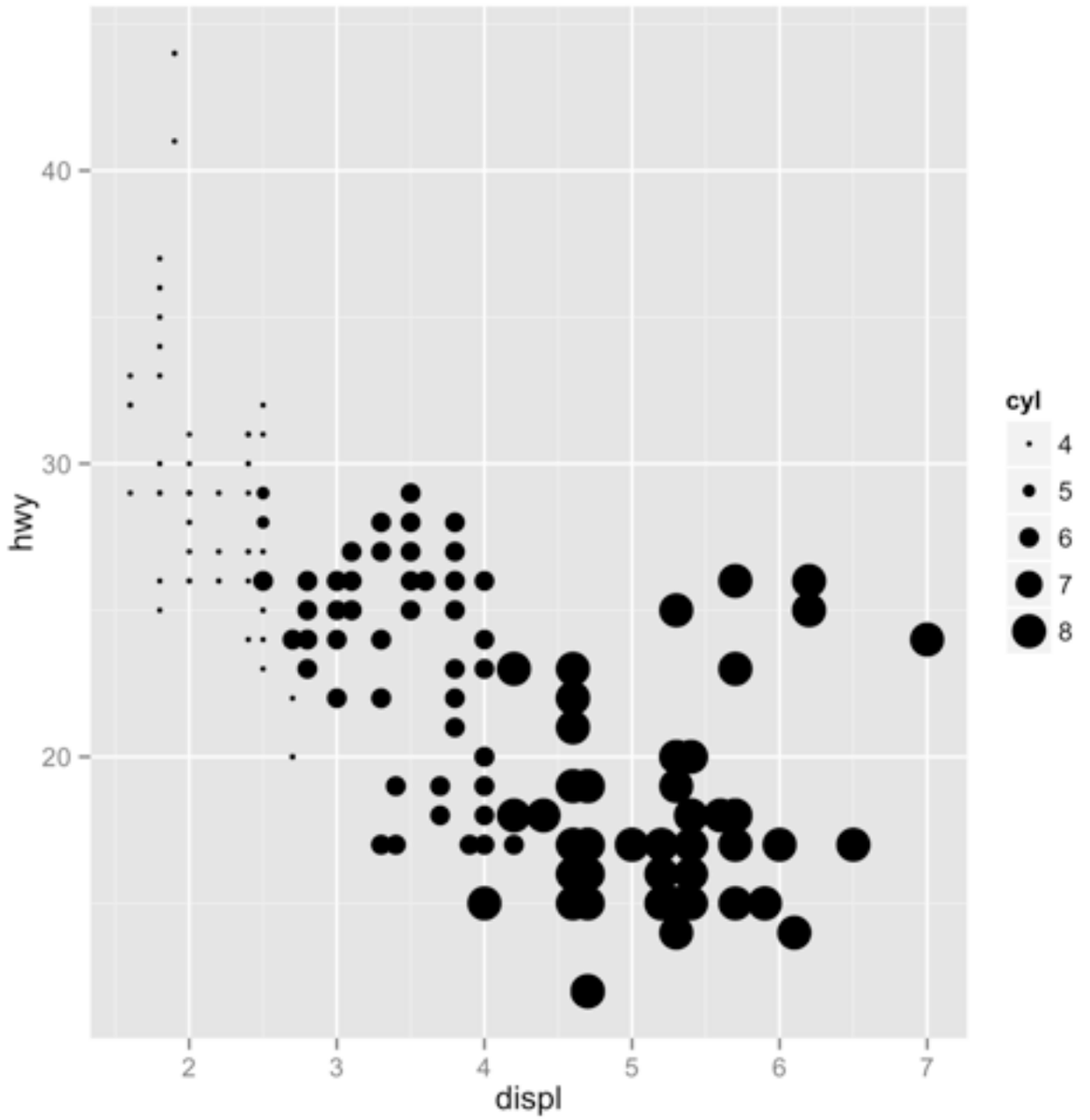



```
qplot(displ, hwy, data = mpg,
  color = cty)
```

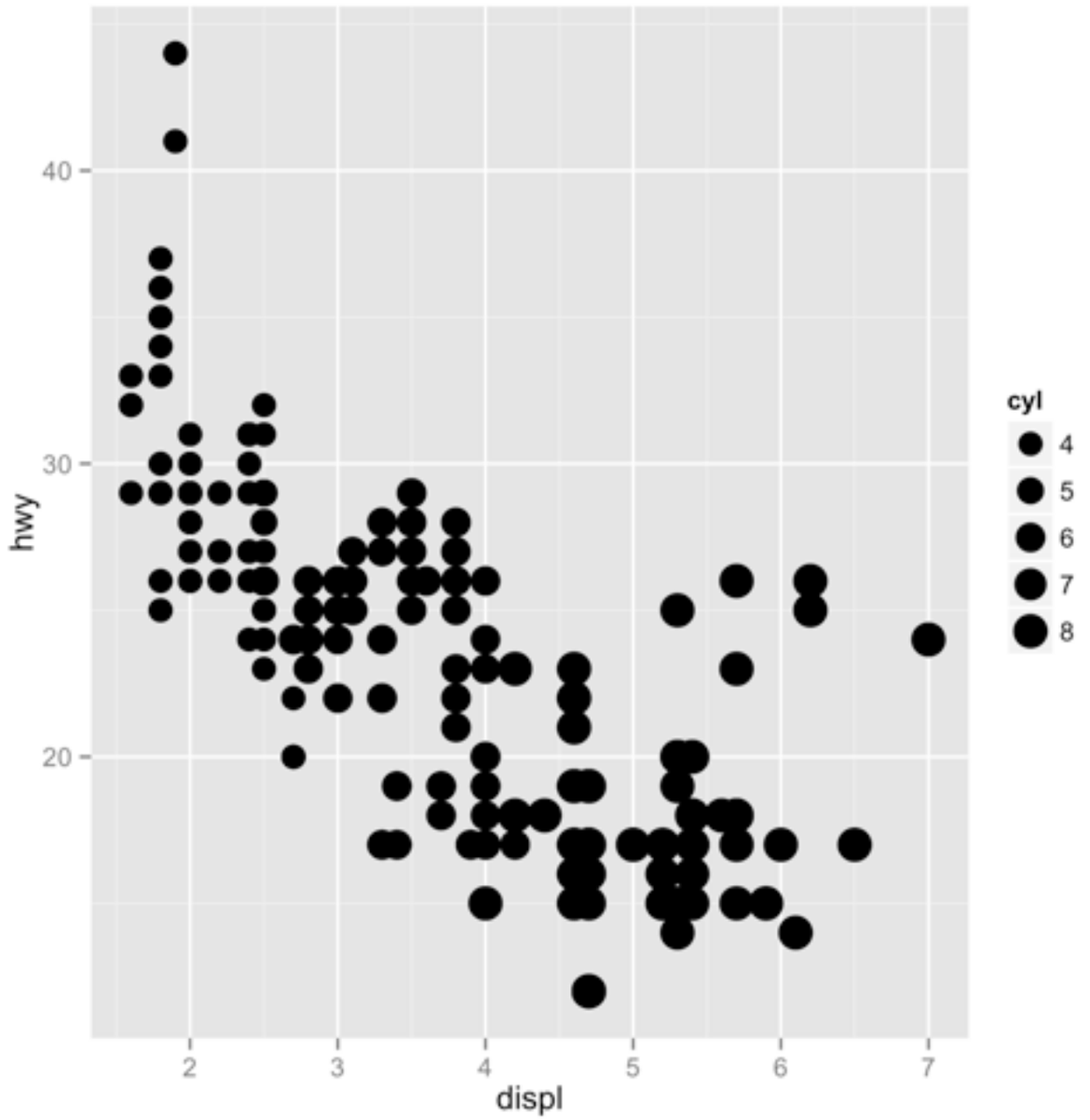



```
same + scale_color_gradient(
  low = "red", high = "yellow")
```

scale_size_area

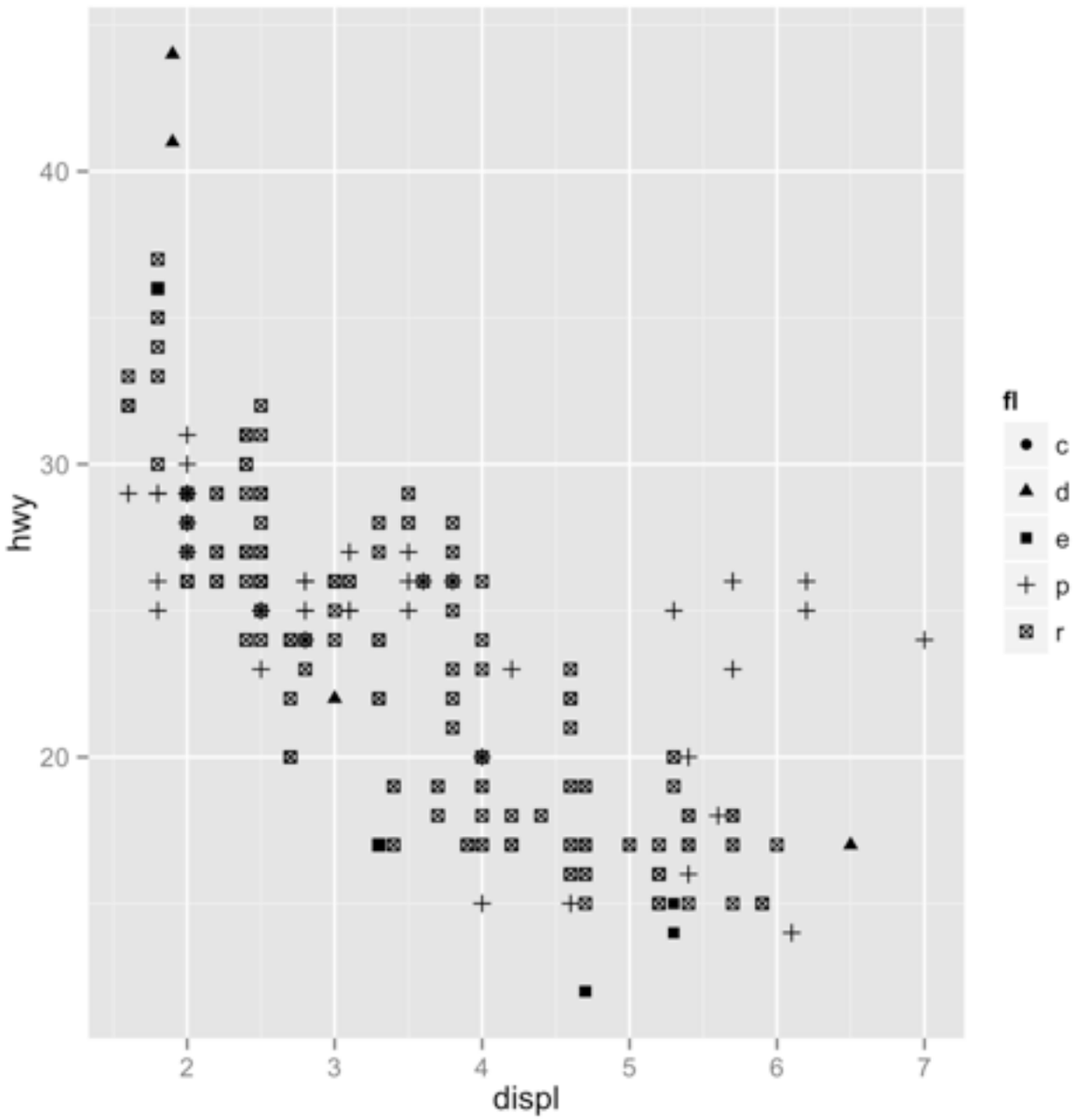



```
qplot(displ, hwy, data = mpg,
  size = cyl)
```

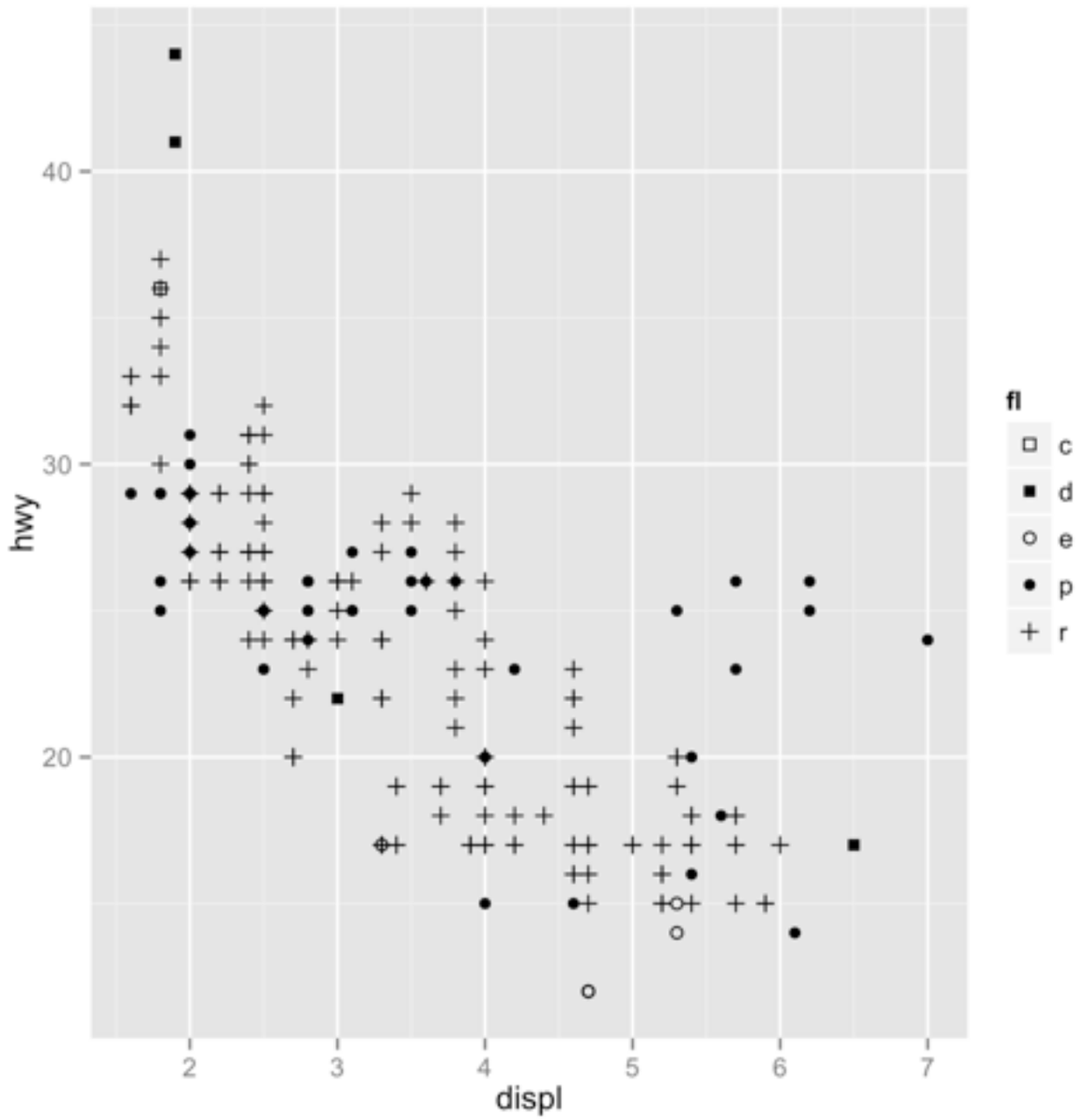



```
same + scale_size_area()
```

scale_shape_manual



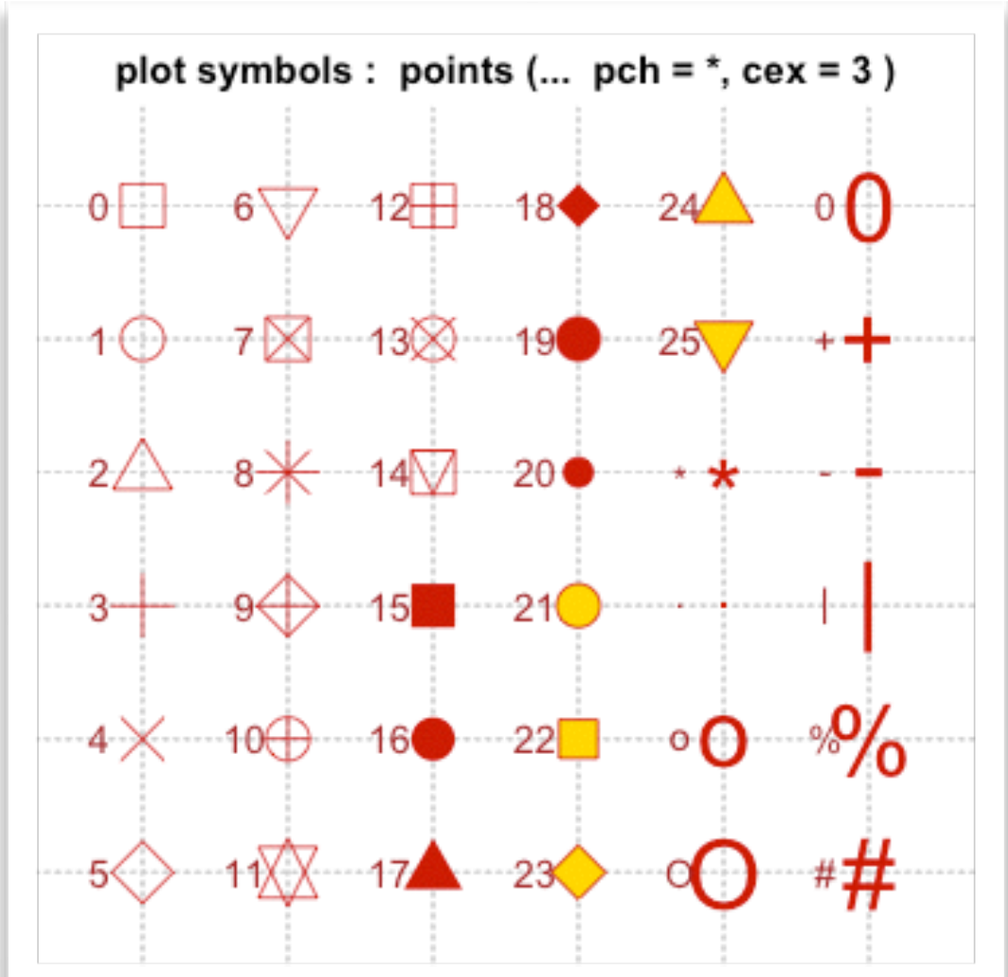

```
qplot(displ, hwy, data = mpg,
  shape = fl)
```




```
same + scale_shape_manual(
  values = c(0, 15, 1, 16, 3))
```

Aside: shapes

```
r <- qplot(displ, cty, colour = drv, shape = fl,
  data = mpg)

# Specify the shapes manually
r + scale_shape_manual(
  values = c(0, 15, 1, 16, 3))

?pch
```

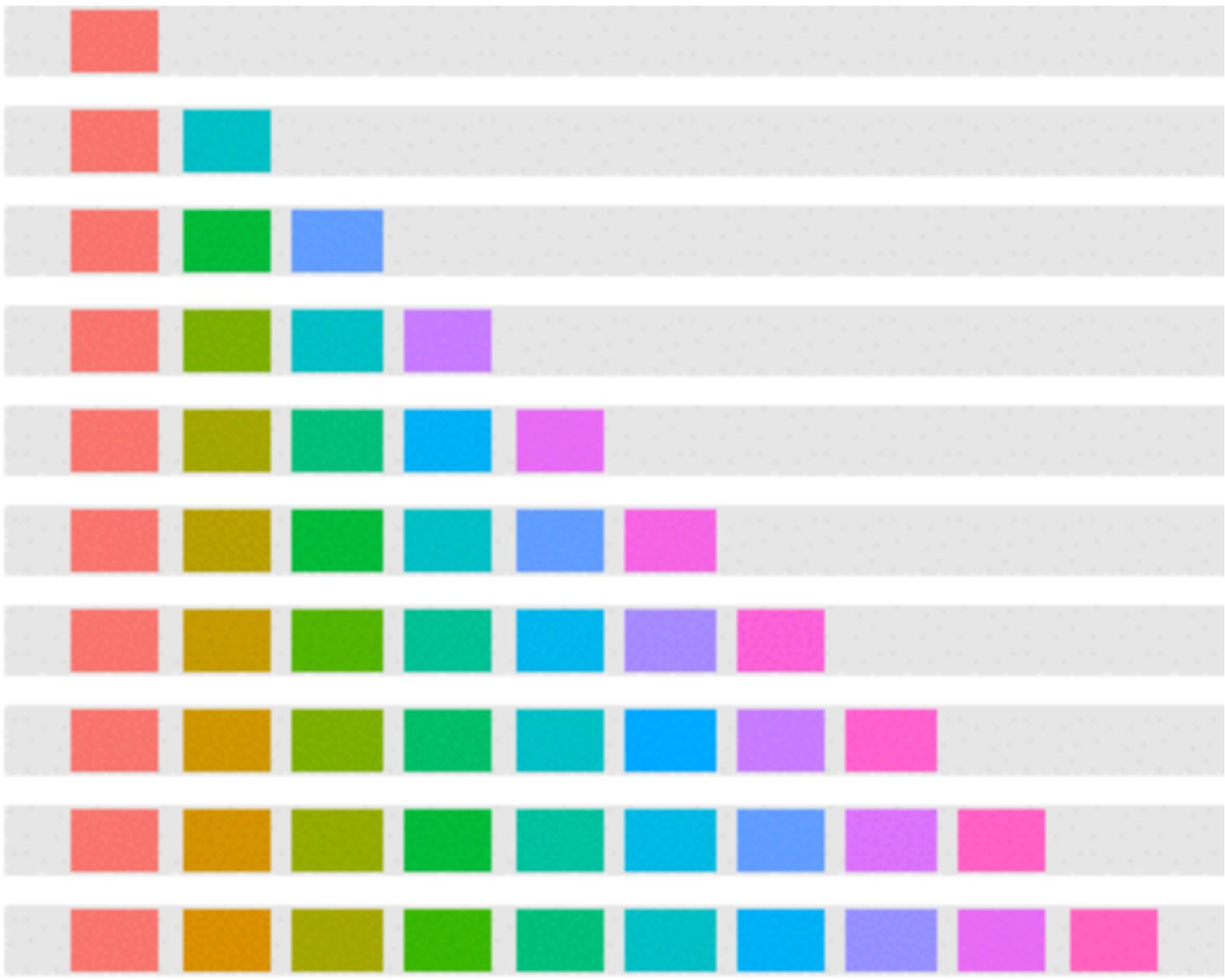


Colors

Colour is the most popular aesthetic after position. It is also the easiest to misuse.

ggplot2's default discrete palettes

(depends on how many colors are needed)



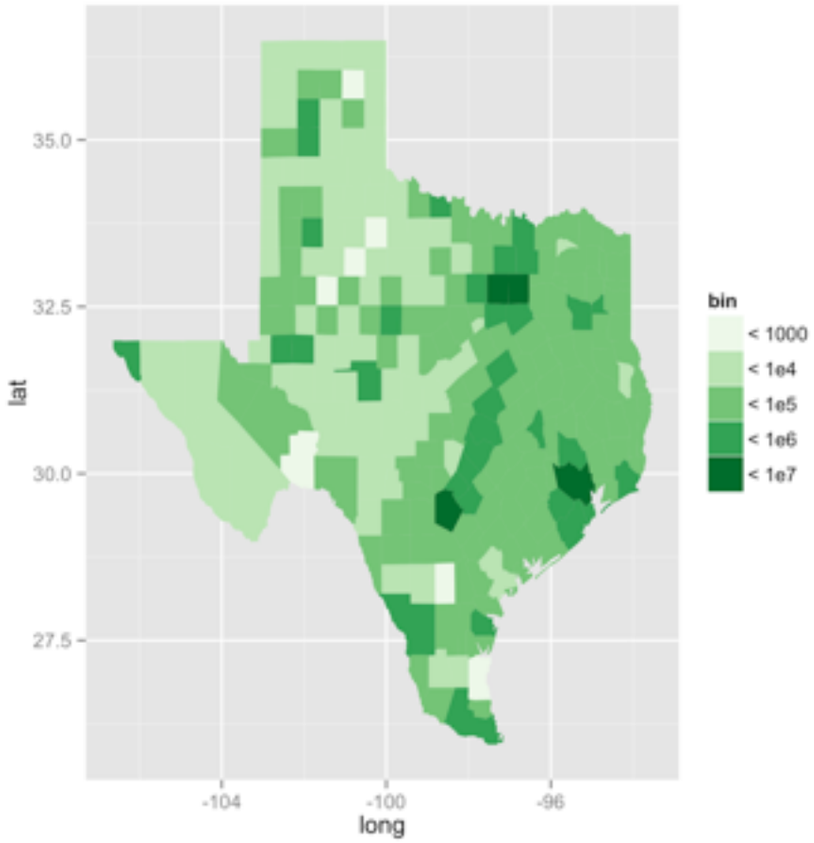
Default continuous palette

But what if you want something different?

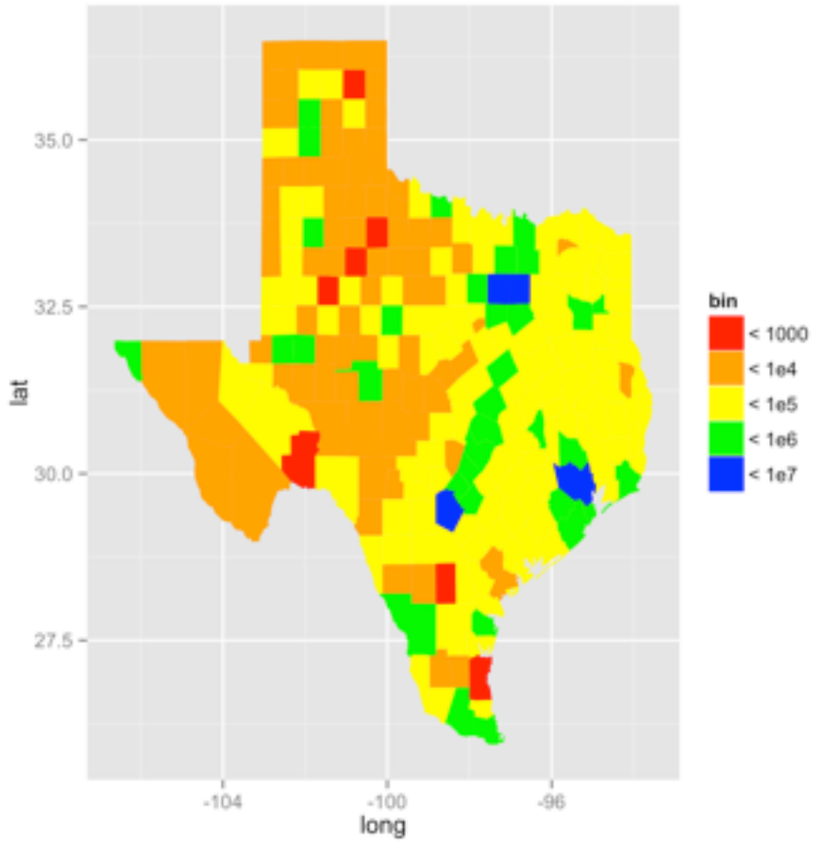
Color scales

scale	data	creates	further customize with these arguments
scale_*_brewer	discrete	nice looking color scheme	palette
scale_*_manual	discrete	the exact colors you list	values
scale_*_hue	discrete	hues evenly spaced throughout HCL color space	h, c, l
scale_*_grey	discrete or continuous	greyscale values	
scale_*_gradient	continuous	gradient between a low color and a high color	low, high
scale_*_gradient2	continuous	gradient with neutral midrange and opposite colored extremes	low, mid, high, midpoint
scale_*_gradientn	continuous	gradient that moves through range of colors	colours

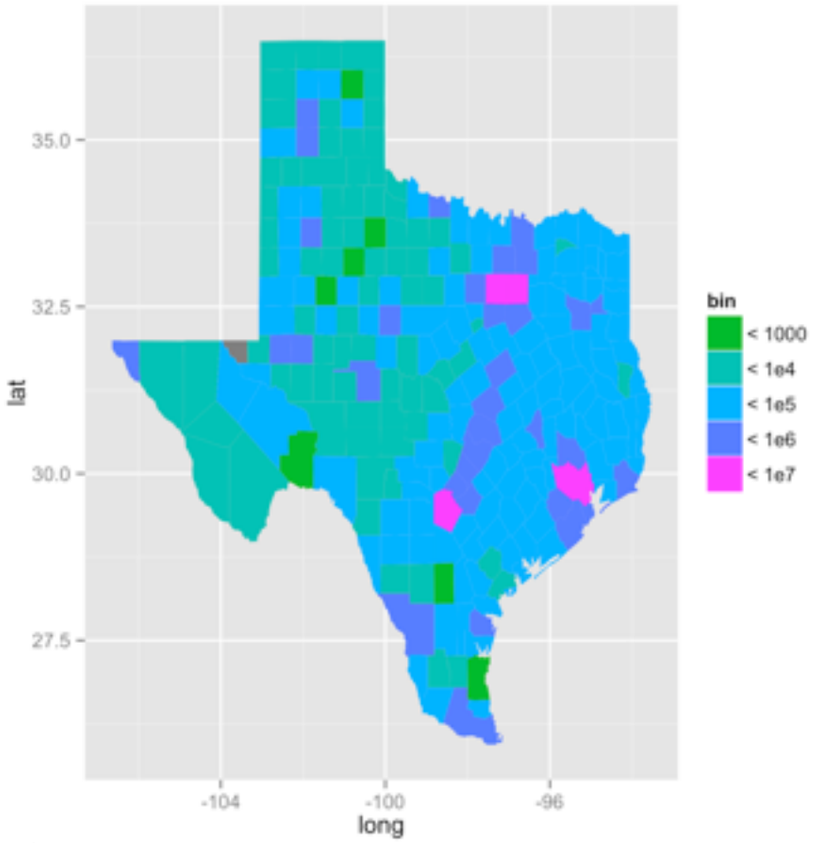
custom != pretty




```
tx +
  scale_fill_brewer(palette = "Greens")
```




```
tx +
  scale_fill_manual(values = c("red",
  "orange", "yellow", "green", "blue"))
```



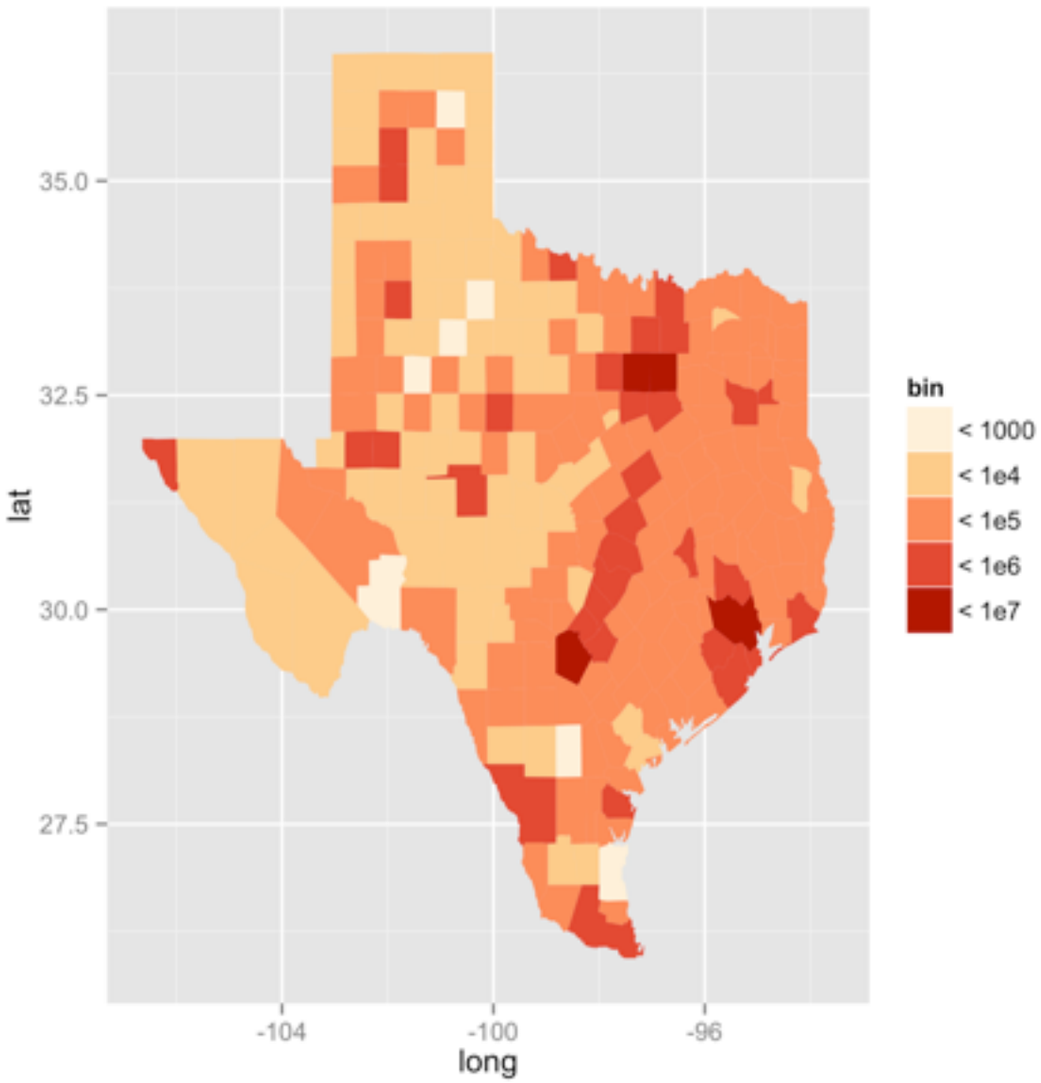

```
tx +
  scale_fill_hue(h = c(150, 300),
  c = 150, l = 60)
```

The brewer scale

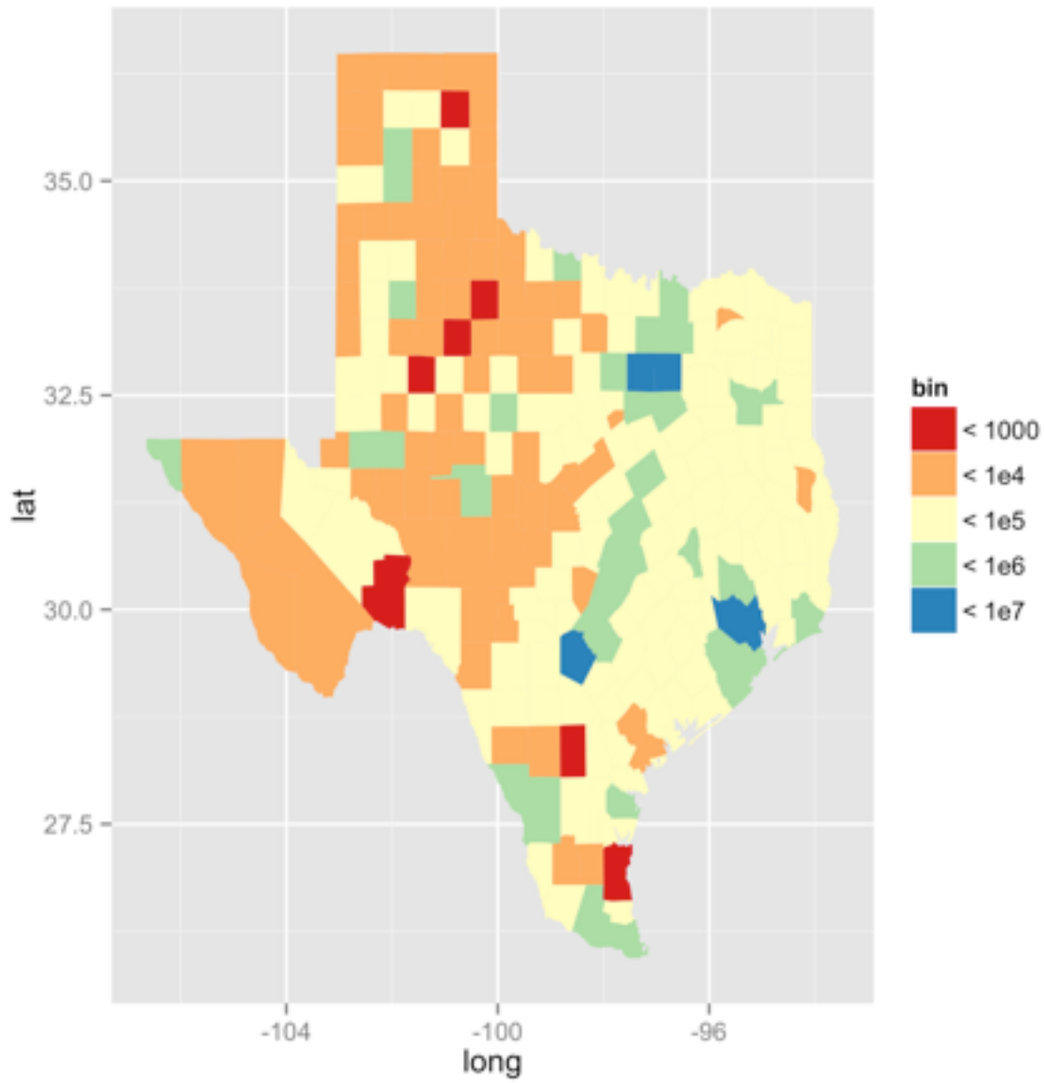
`scale_color_brewer`

`scale_fill_brewer`

new parameter	controls
palette	name of a palette in the RColorBrewer package




```
tx + scale_fill_brewer(
    palette = "OrRd")
```




```
tx + scale_fill_brewer(
    palette = "Spectral")
```

Color brewer

Cynthia Brewer developed useful, pleasing palettes, particularly tailored for maps: http://colorbrewer2.org

To see a list of each brewer palette, run the following command in the RColorBrewer library.

```
library(RColorBrewer)
display.brewer.all()


tx + scale_fill_brewer(palette="OrRd")
```

Your turn

```
display.brewer.all()
```

Experiment with the different palettes available for the brewer scale. Add a color scheme that you like to `tx`.

There are scales for every aesthetic ggplot2 uses, not just color. To see complete list with examples, visit

http://docs.ggplot2.org/current

Scales

Scales control the mapping between data and aesthetics.

- `expand_limits`
  Expand the plot limits with data.
- `guides`
  Set guides for each scale.
- `guide_legend`
  Legend guide.
- `guide_colourbar` (guide_colorbar)
  Contiuous colour bar guide.
- `scale_alpha` (scale_alpha_continuous, scale_alpha_discrete)
  Alpha scales.

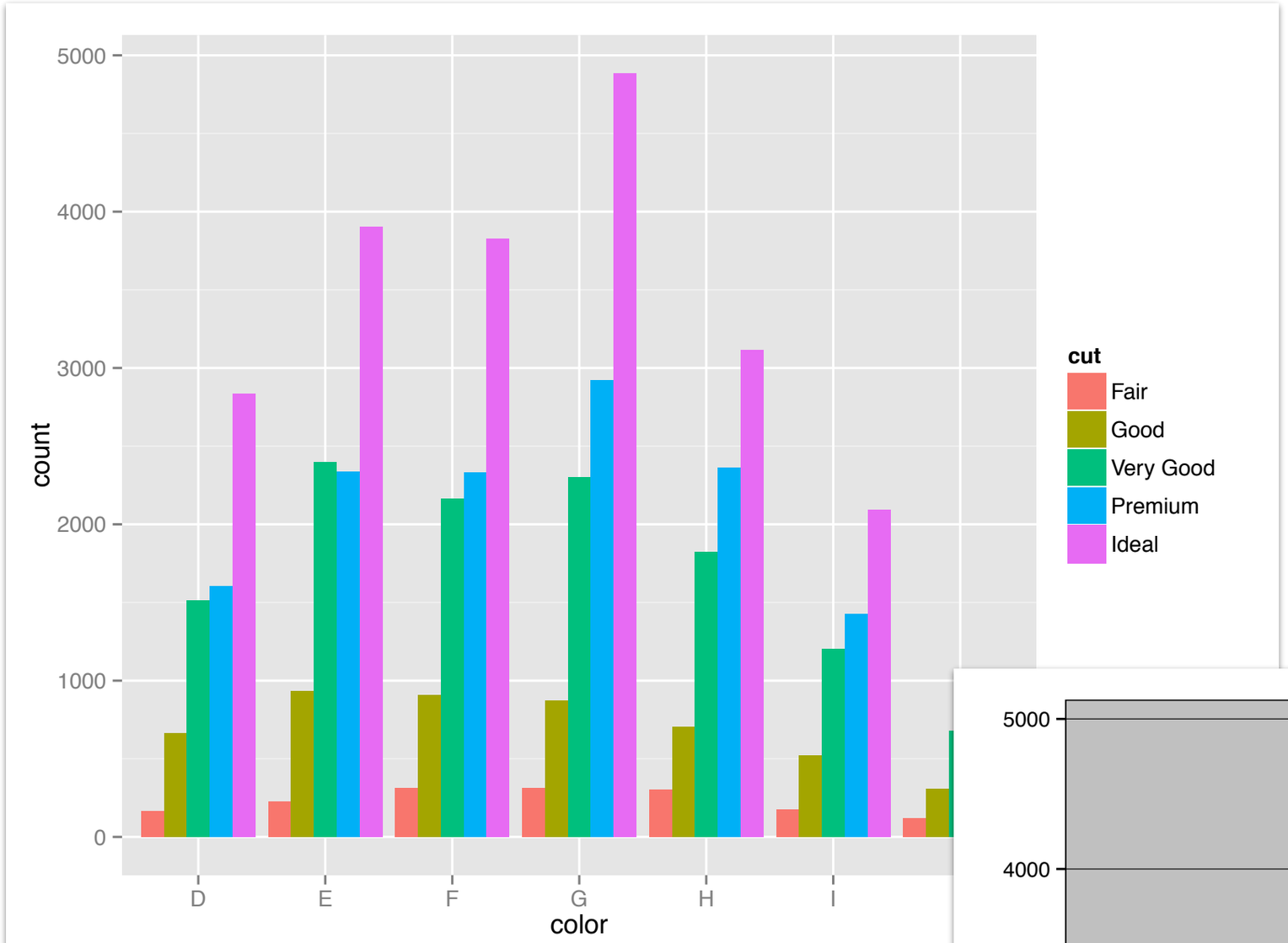
- `scale_area`
  Scale area instead of radius (for size).
- `scale_colour_brewer` (scale_color_brewer, scale_fill_brewer)
  Sequential, diverging and qualitative colour scales from colorbrewer.org
- `scale_colour_gradient` (scale_color_continuous, scale_color_gradient, scale_colour_continuous, scale_fill_continuous, scale_fill_gradient)
  Smooth gradient between two colours

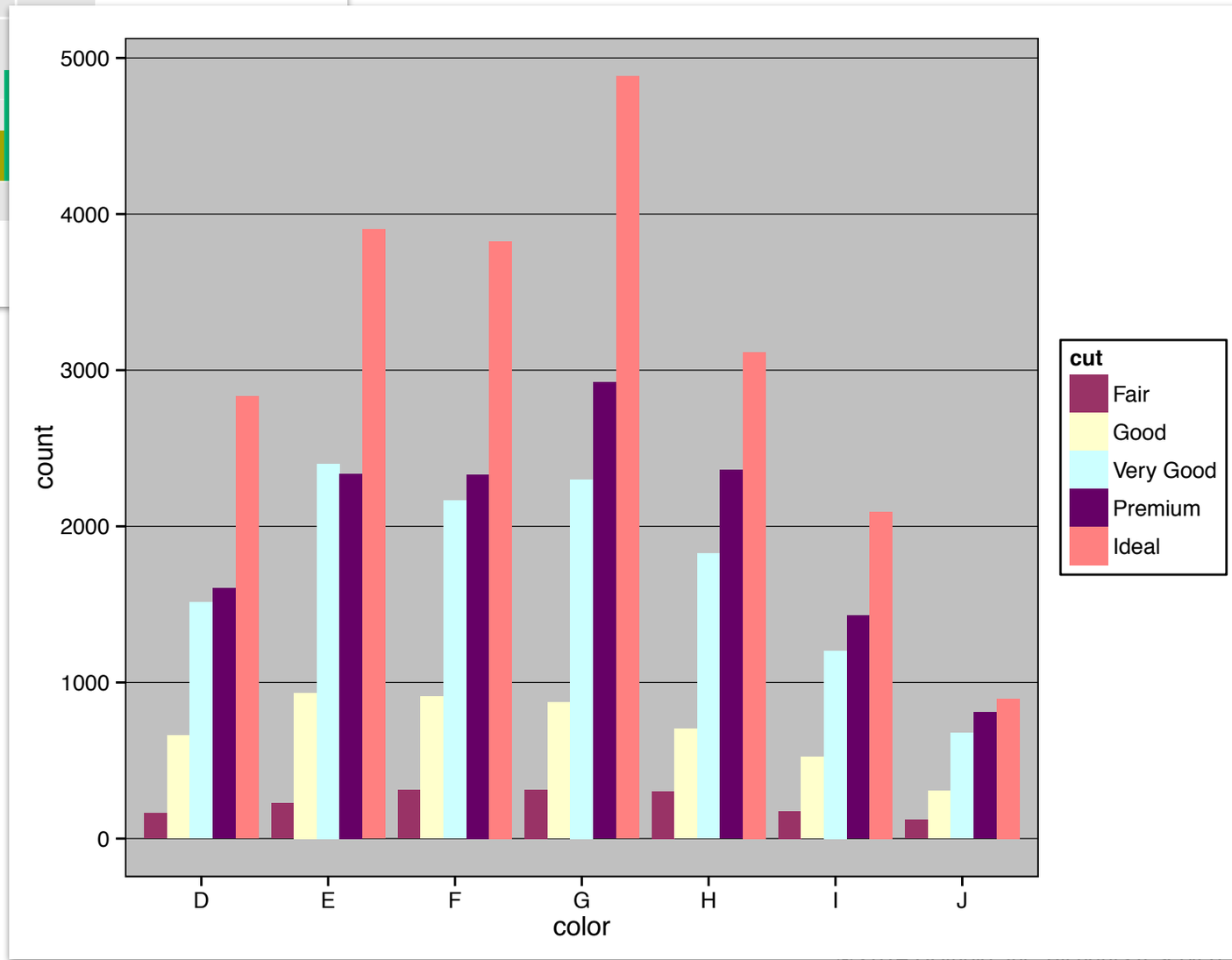
- `scale_colour_gradient2` (scale_color_gradient2, scale_fill_gradient2)
  Diverging colour gradient
- `scale_colour_gradientn` (scale_color_gradientn, scale_fill_gradientn)
  Smooth colour gradient between n colours

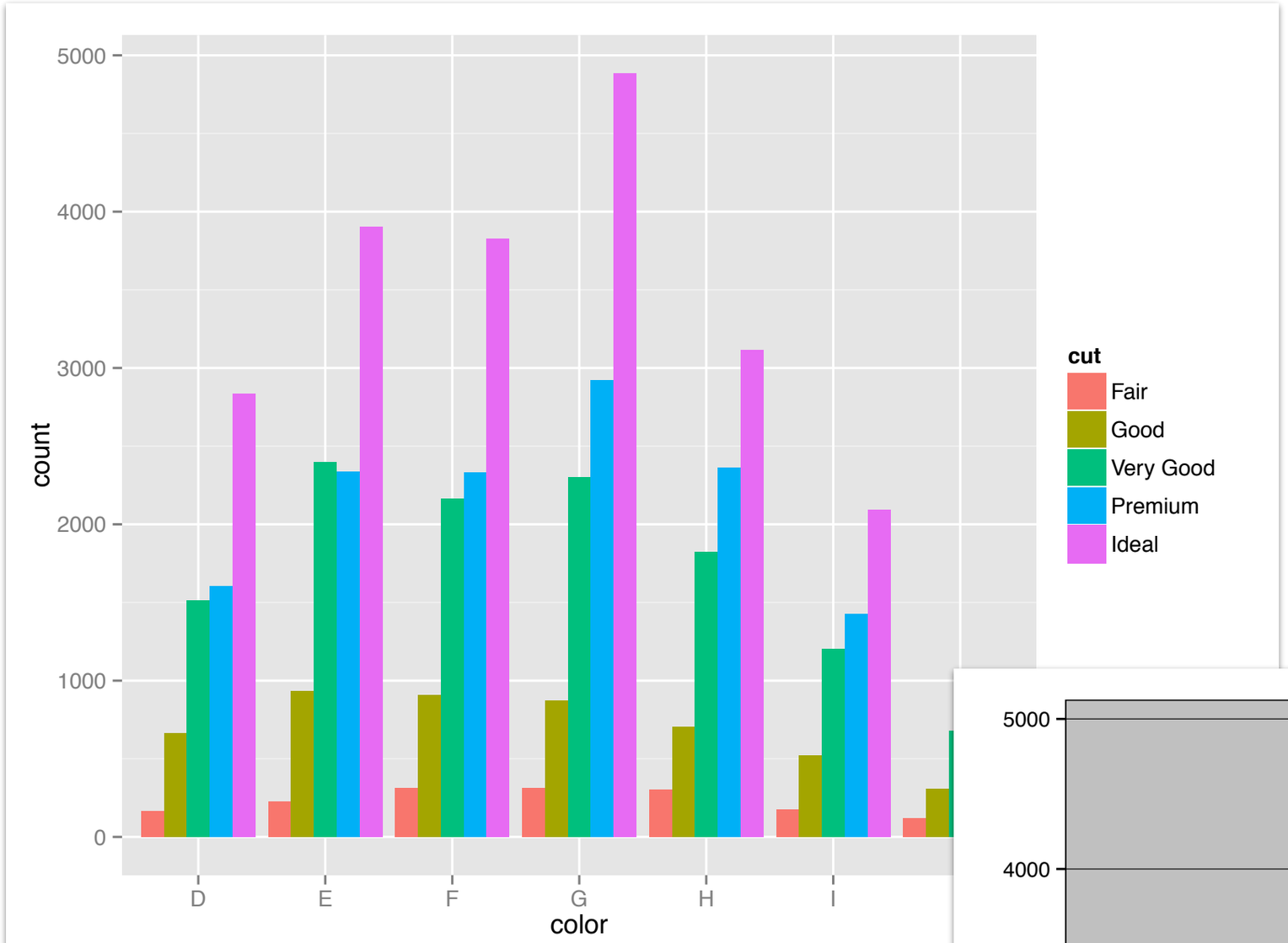
Other aesthetics

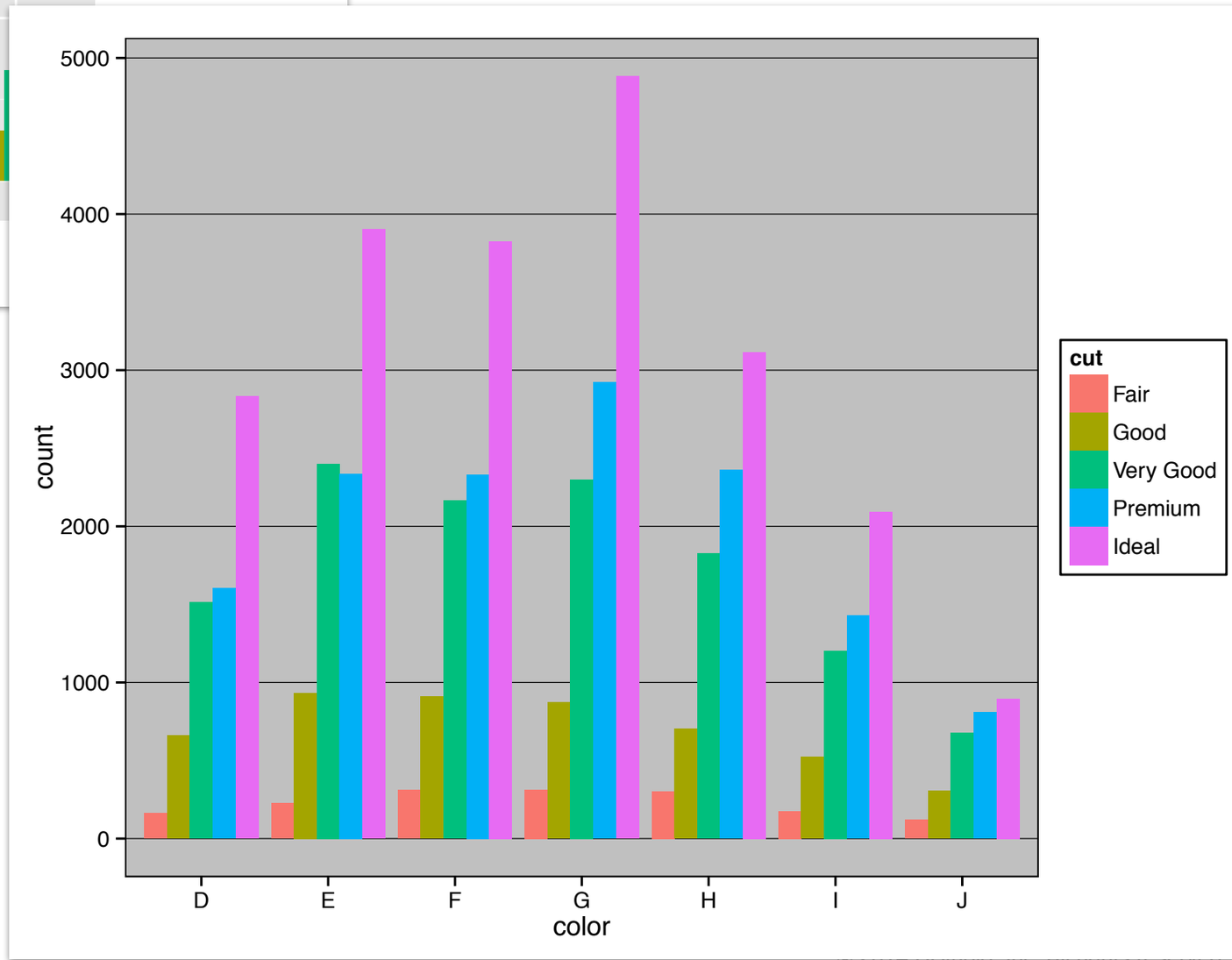
You can manage the relationship used for each aesthetic mapping in your plot the same way: select a scale that applies to that mapping (using the three part scale name convention), and then set additional arguments in the scale function if desired.

Themes









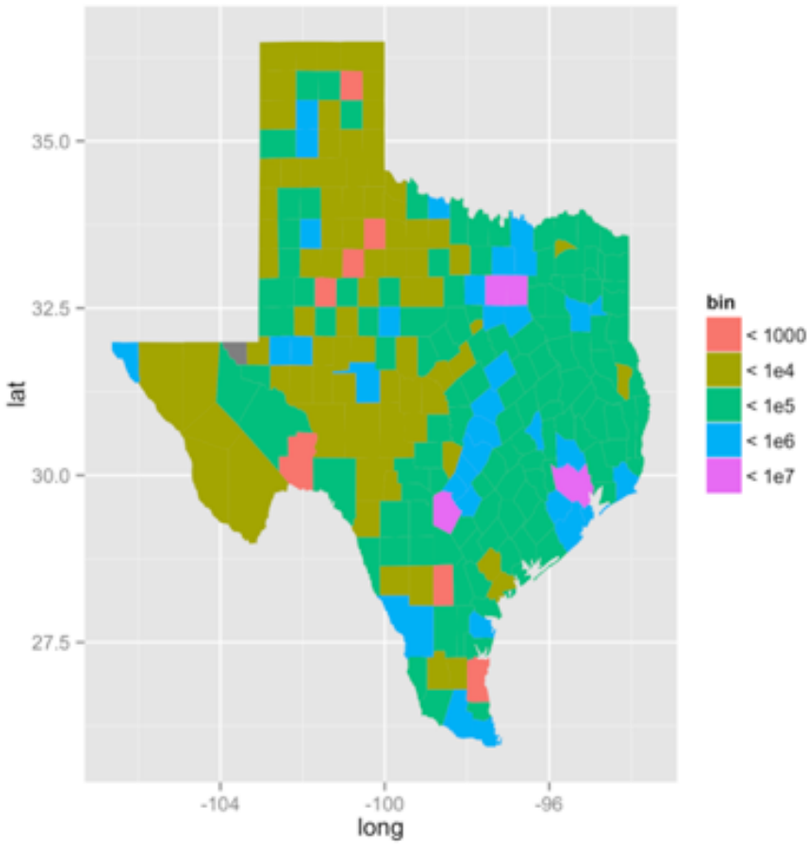
Visual appearance

The theme controls the appearance of the non-geom parts of the plot.

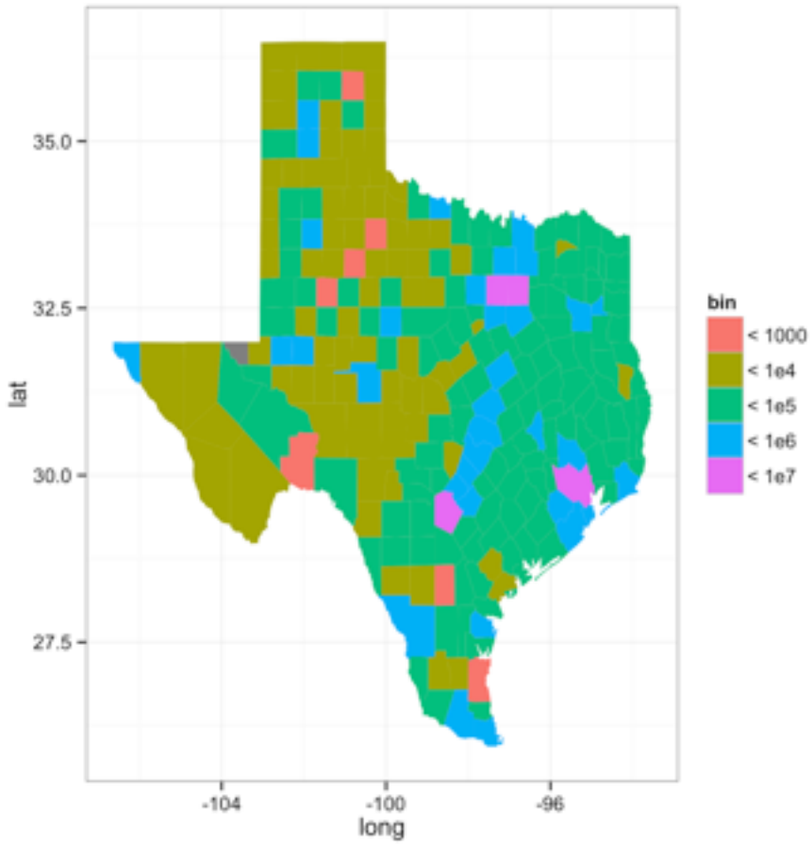
In other words, the theme determines how the plot looks.

themes

ggplot2 comes with two pre-loaded themes that control the appearance of non-data elements



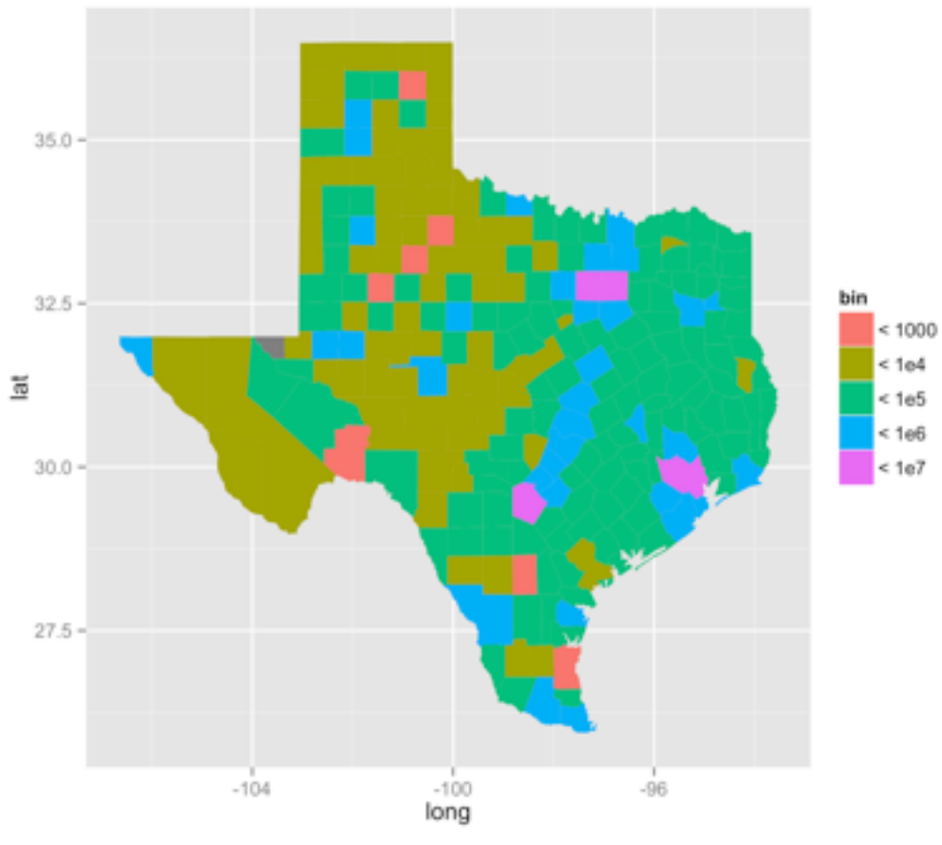

```
tx + theme_grey()
```



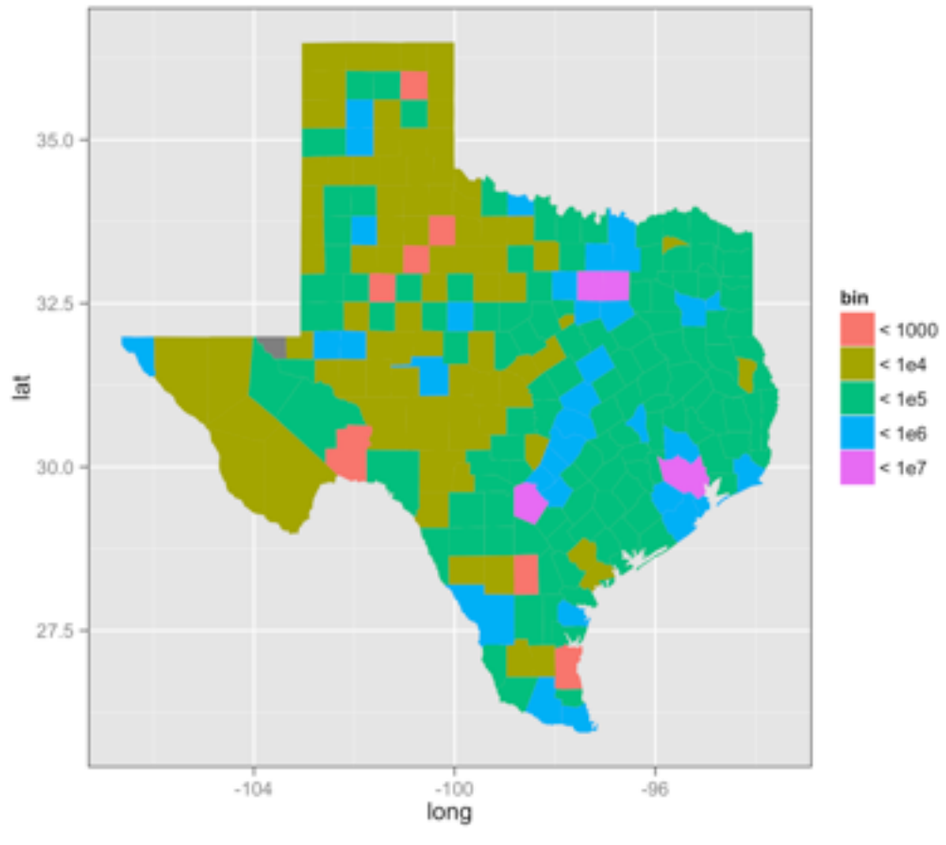

```
tx + theme_bw()
```

Customization

You can modify individual details of the current theme with `theme()` or even create your own theme.




```
tx
```




```
tx + theme(panel.border =
element_rect(colour = "black", fill = NA))
```

Elements

Axis: axis.line, axis.text.x, axis.text.y, axis.ticks, axis.title.x, axis.title.y

Legend: legend.background, legend.key, legend.text, legend.title

Panel: panel.background, panel.border, panel.grid.major, panel.grid.minor

Strip: strip.background, strip.text.x, strip.text.y

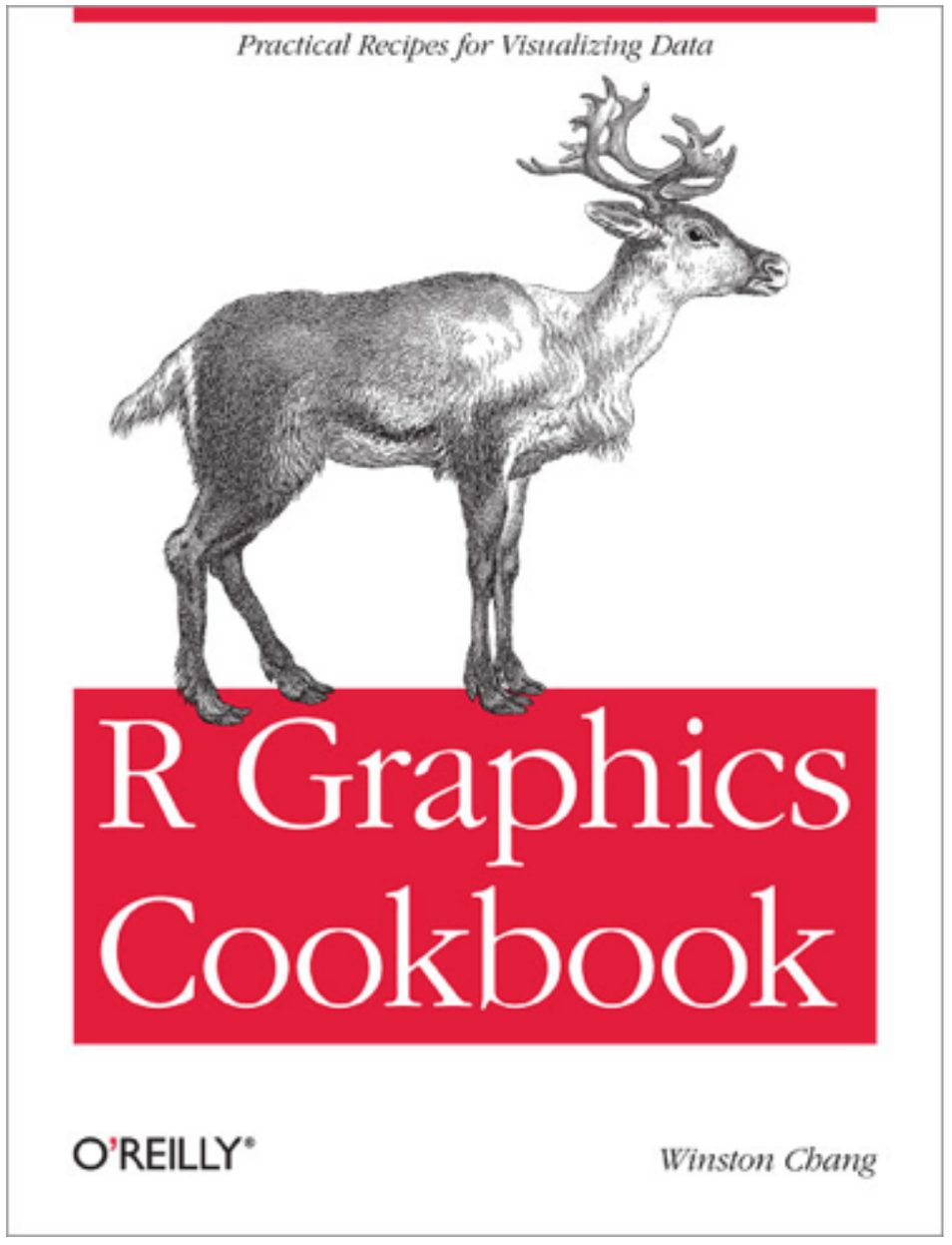
see ?theme

https://github.com/wch/ggplot2/wiki/New-theme-system

themes

To learn how to change individual elements of a theme, I recommend the R Graphics Cookbook by Winston Chang

http://shop.oreilly.com/product/0636920023135.do#



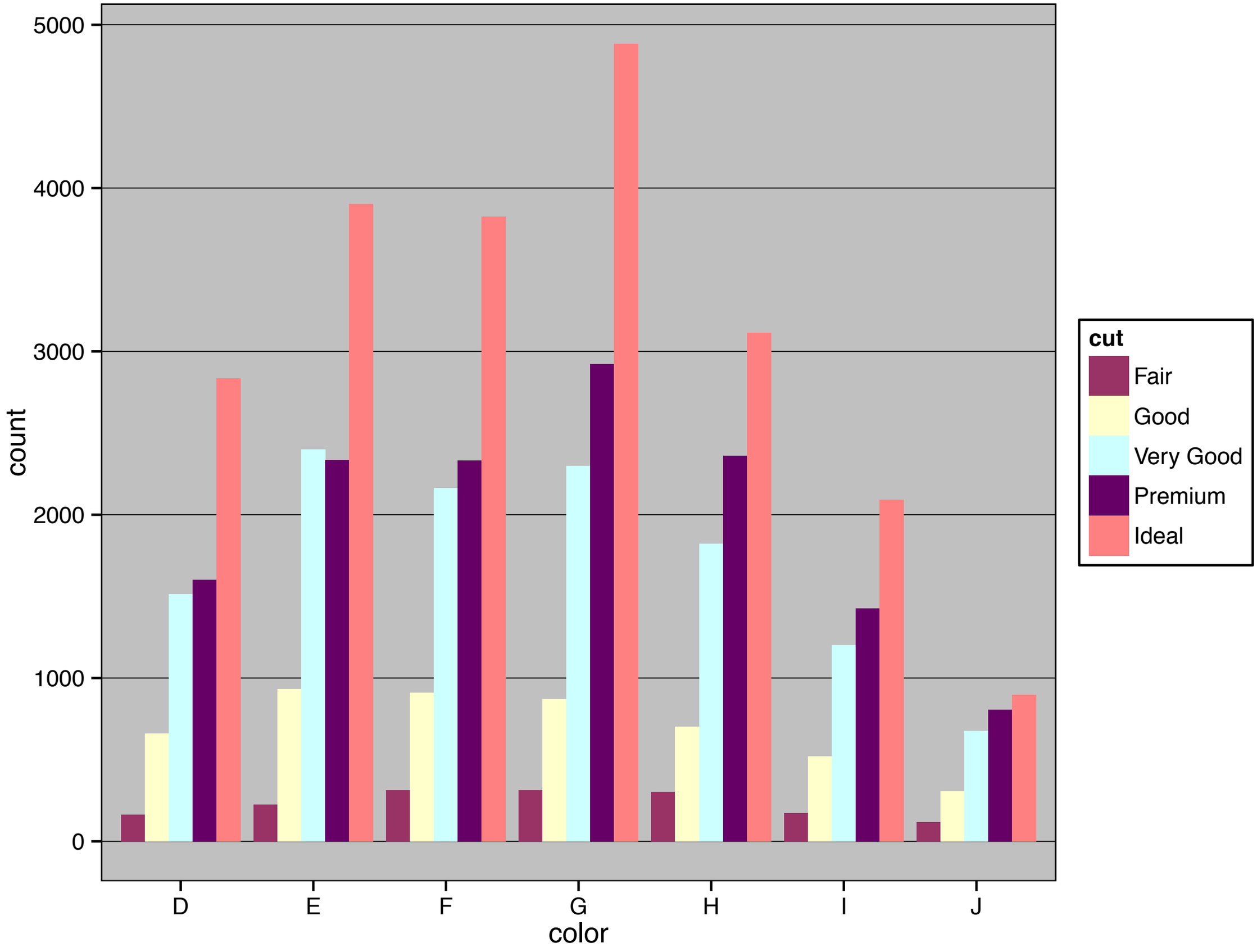
pre-made themes

ggthemes

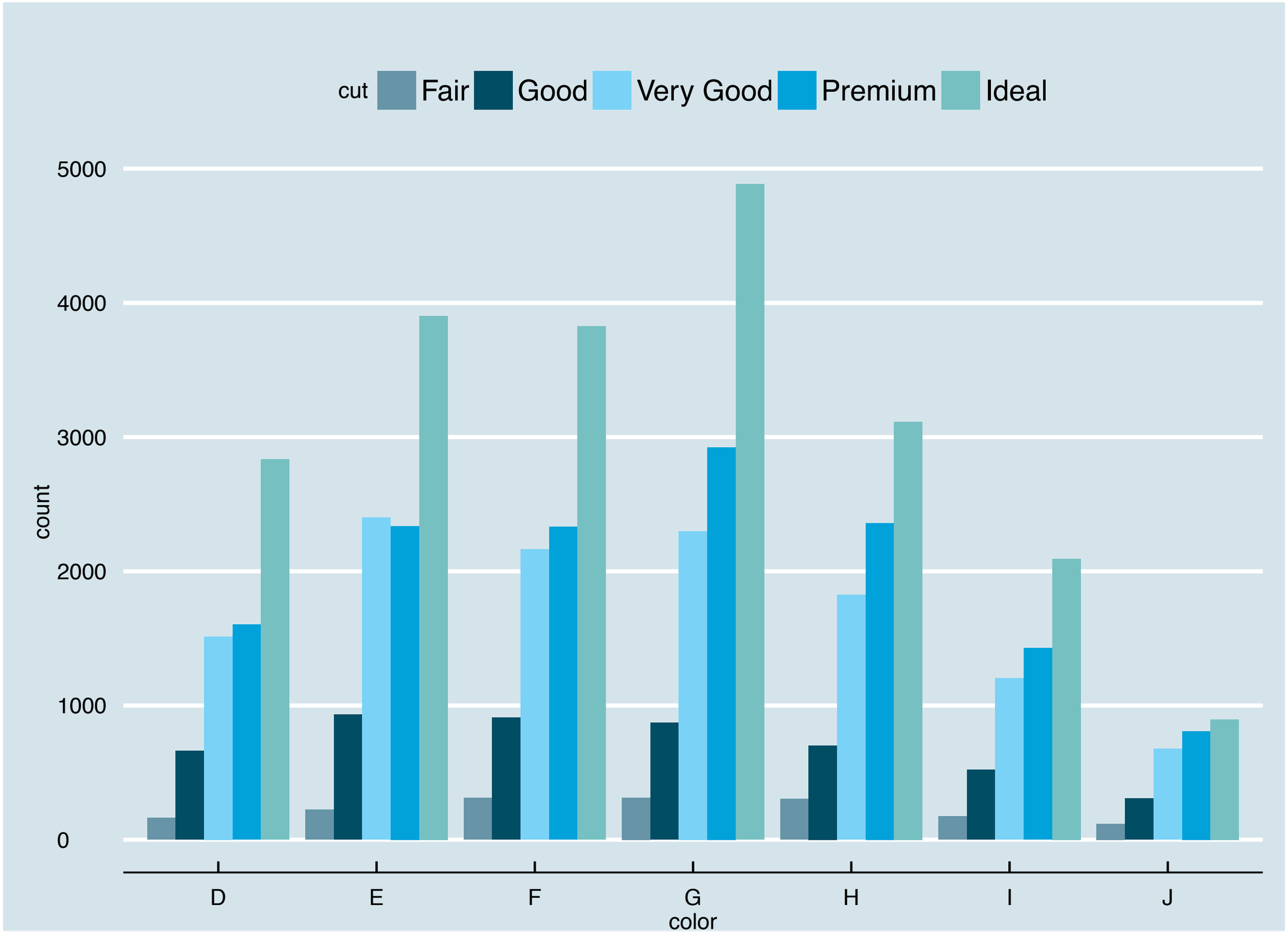
```
install.packages("ggthemes")
```

```
library(ggthemes)
p <- qplot(color, data = diamonds,
   fill = cut, position = "dodge")
```

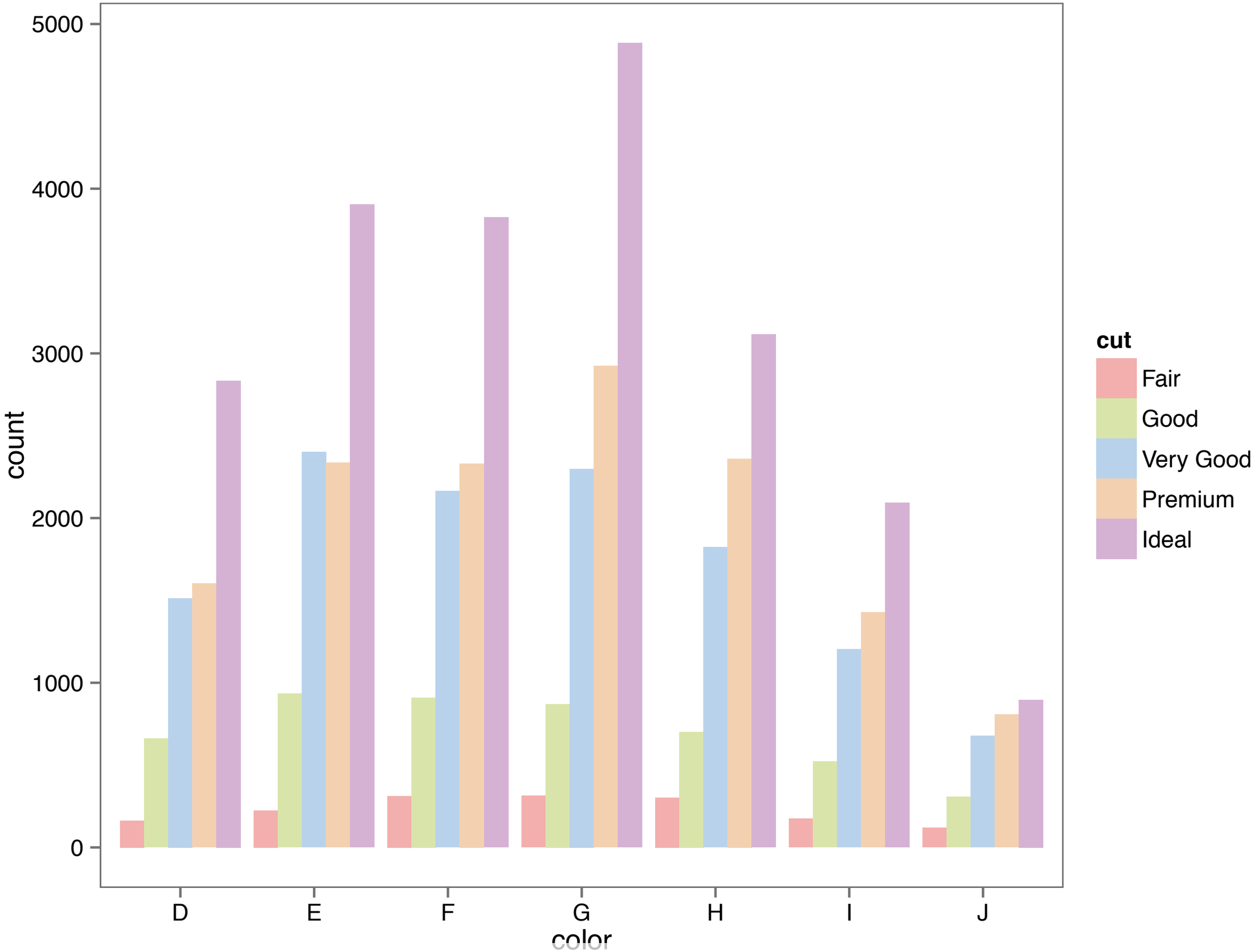
https://github.com/jrnold/ggthemes



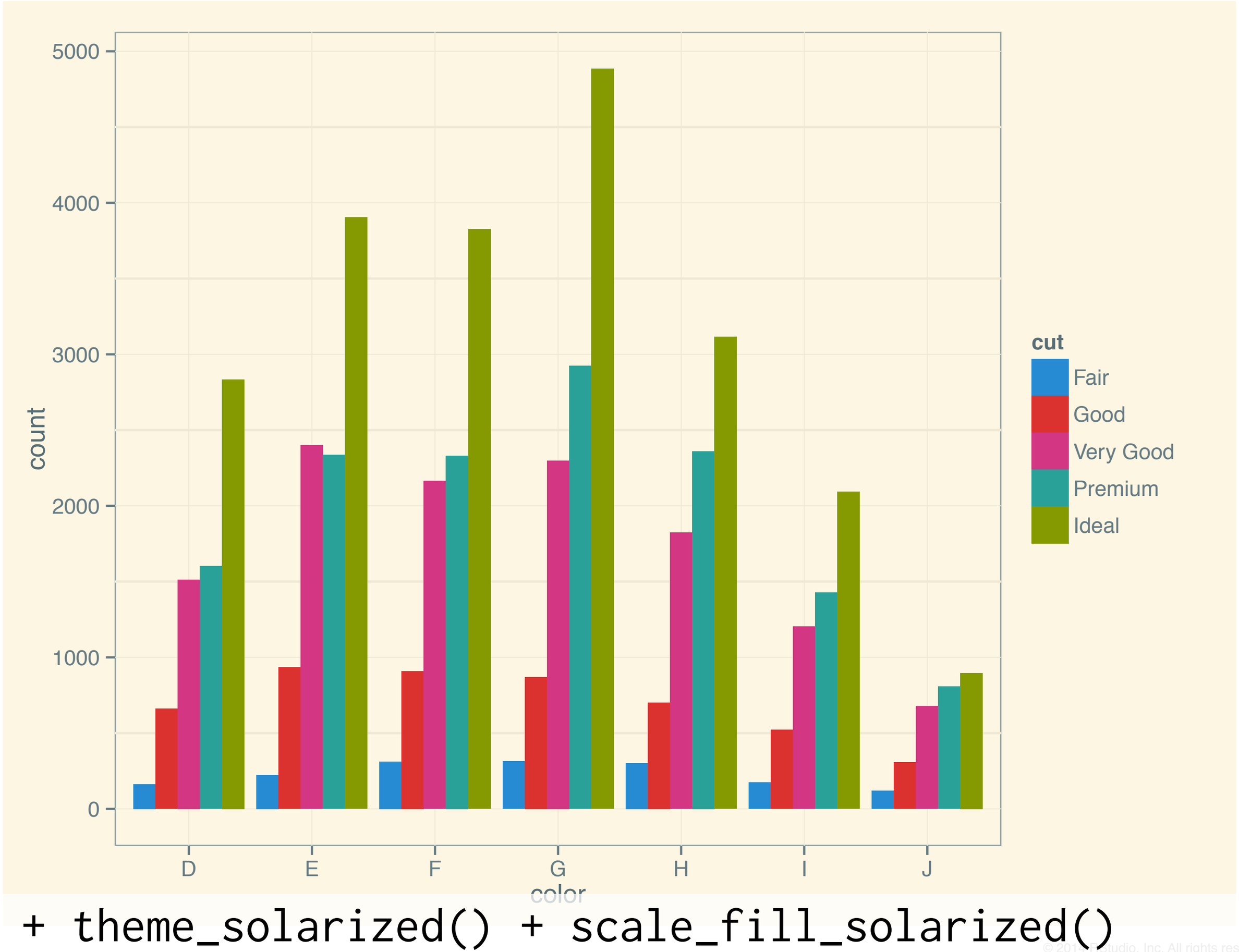

```
p + theme_excel() + scale_fill_excel()
```



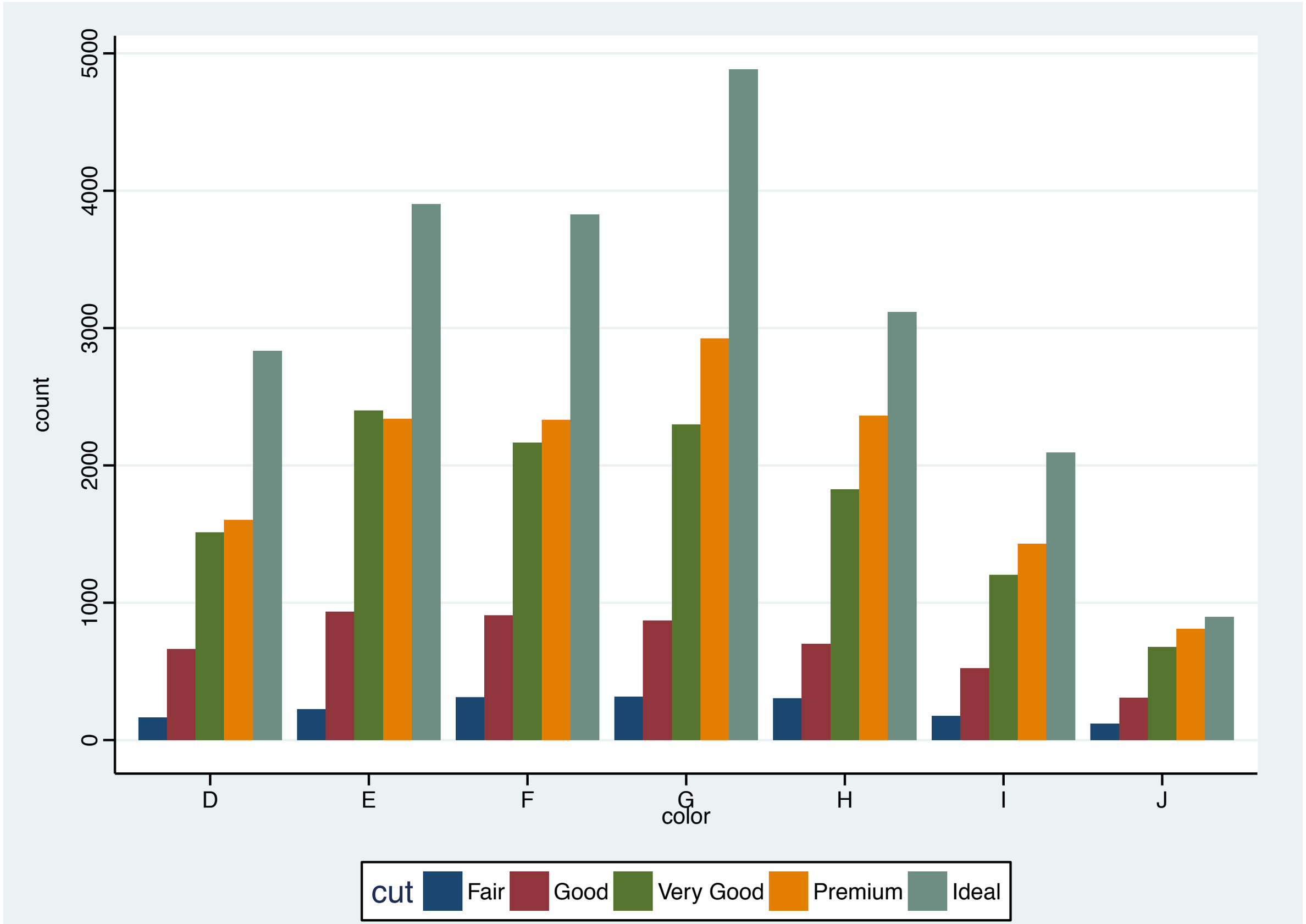

```
p + theme_economist() + scale_fill_economist()
```



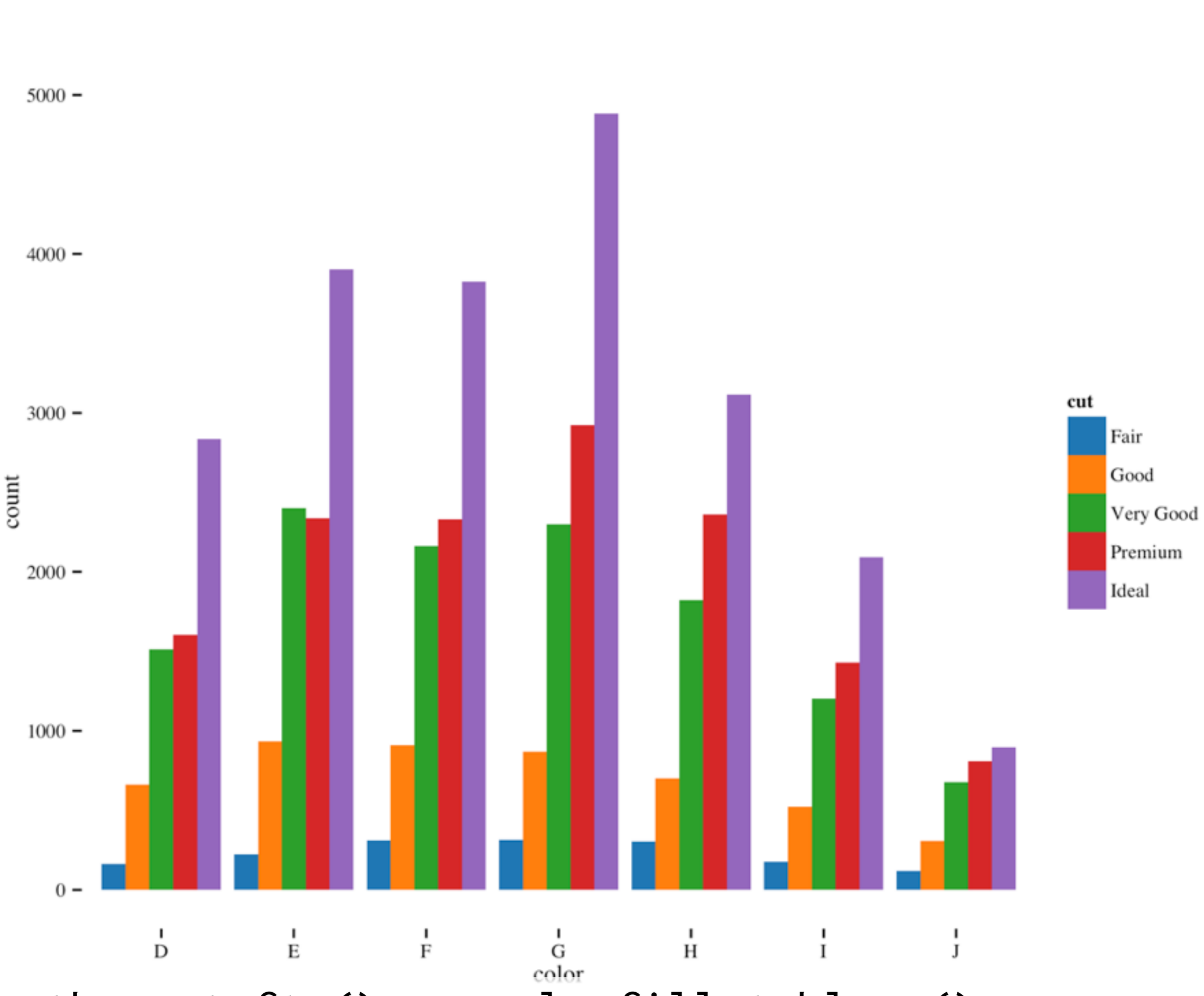

```
p + theme_few() + scale_fill_few()
```



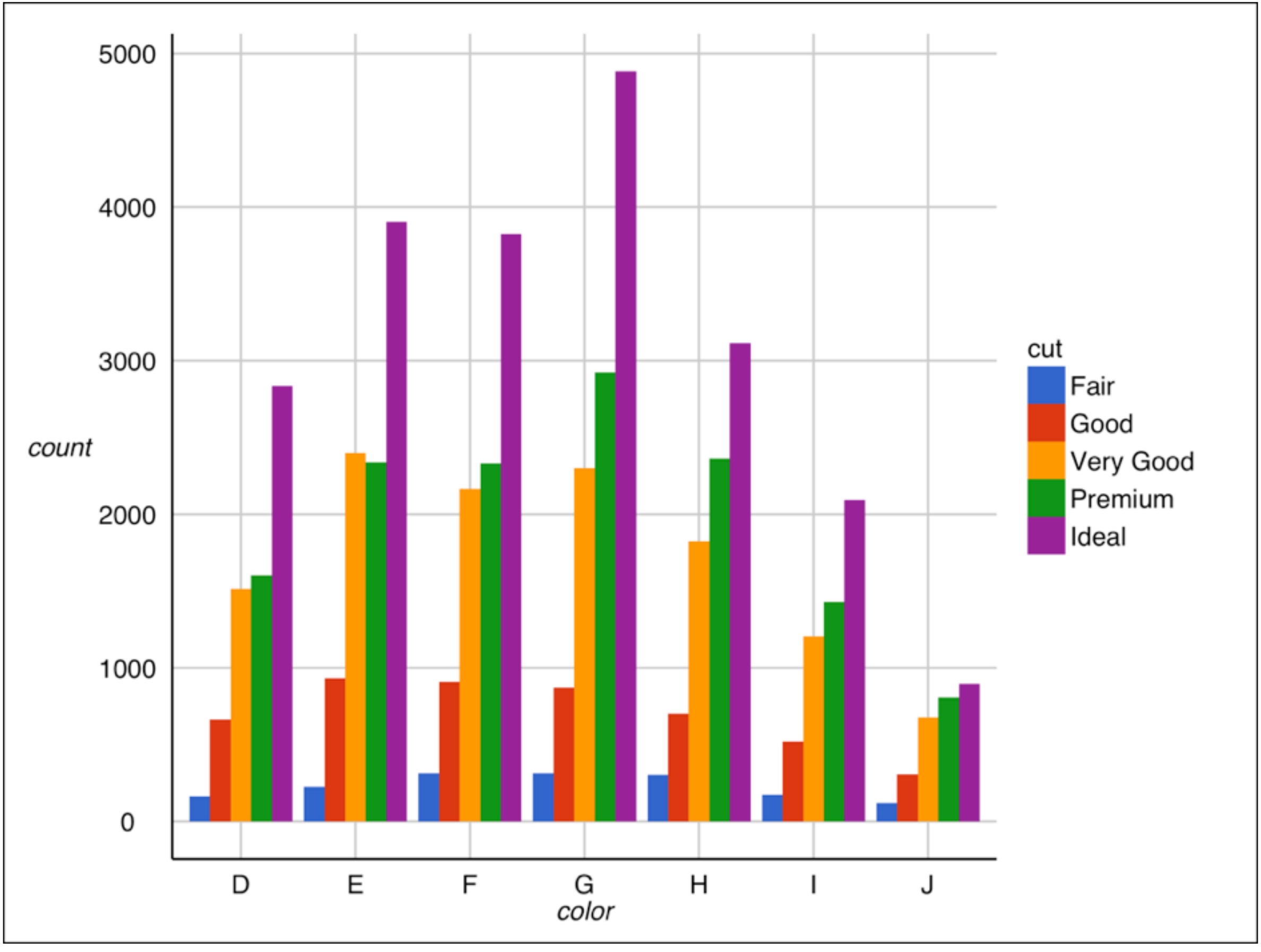

```
p + theme_solarized() + scale_fill_solarized()
```



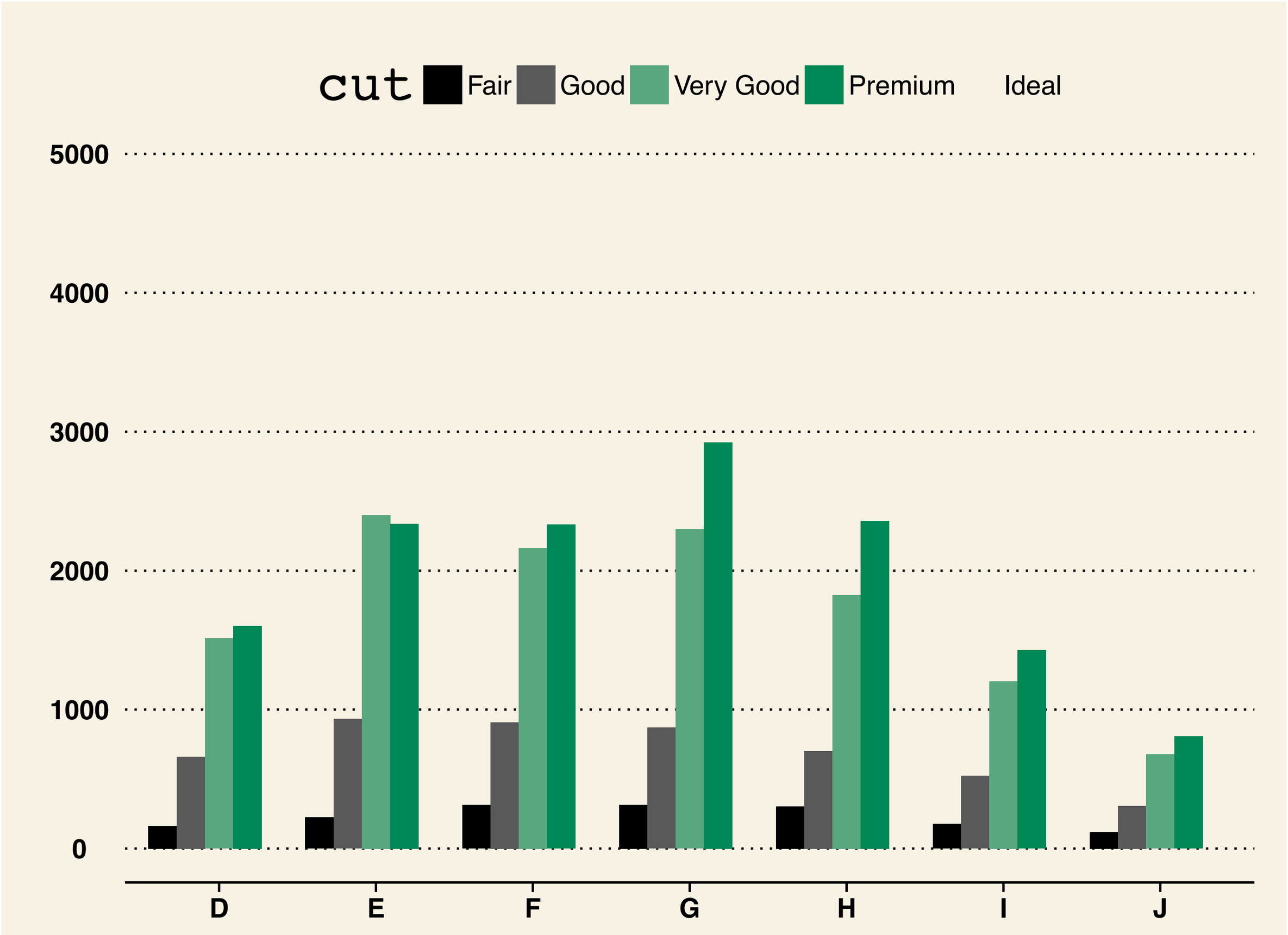

```
p + theme_stata() + scale_fill_stata()
```




```
p + theme_tufte() + scale_fill_tableau()
```




```
p + theme_gdocs() + scale_fill_gdocs()
```



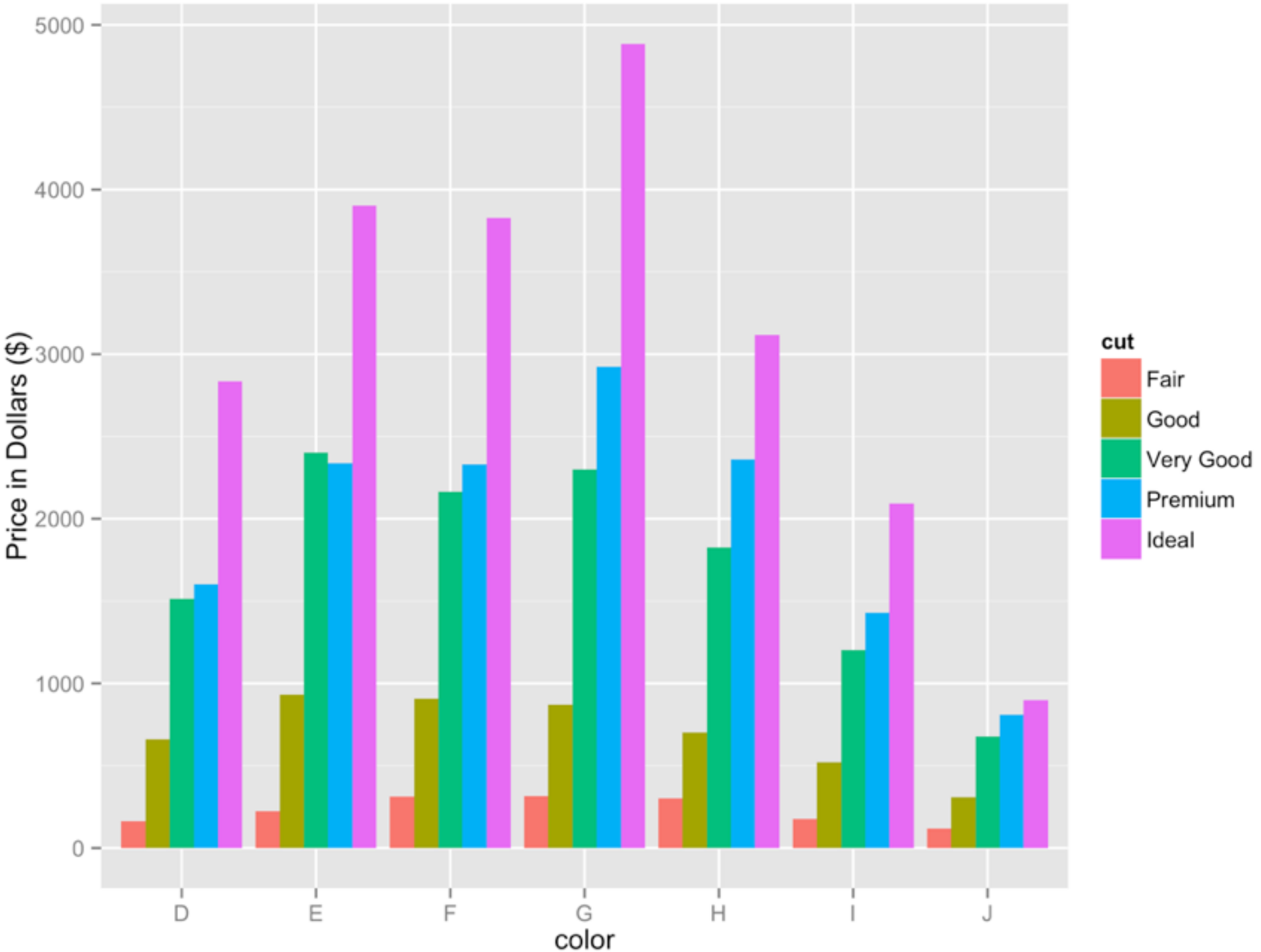

```
p + theme_wsj() + scale_fill_wsj(palette = "black_green")
```

Axis labels

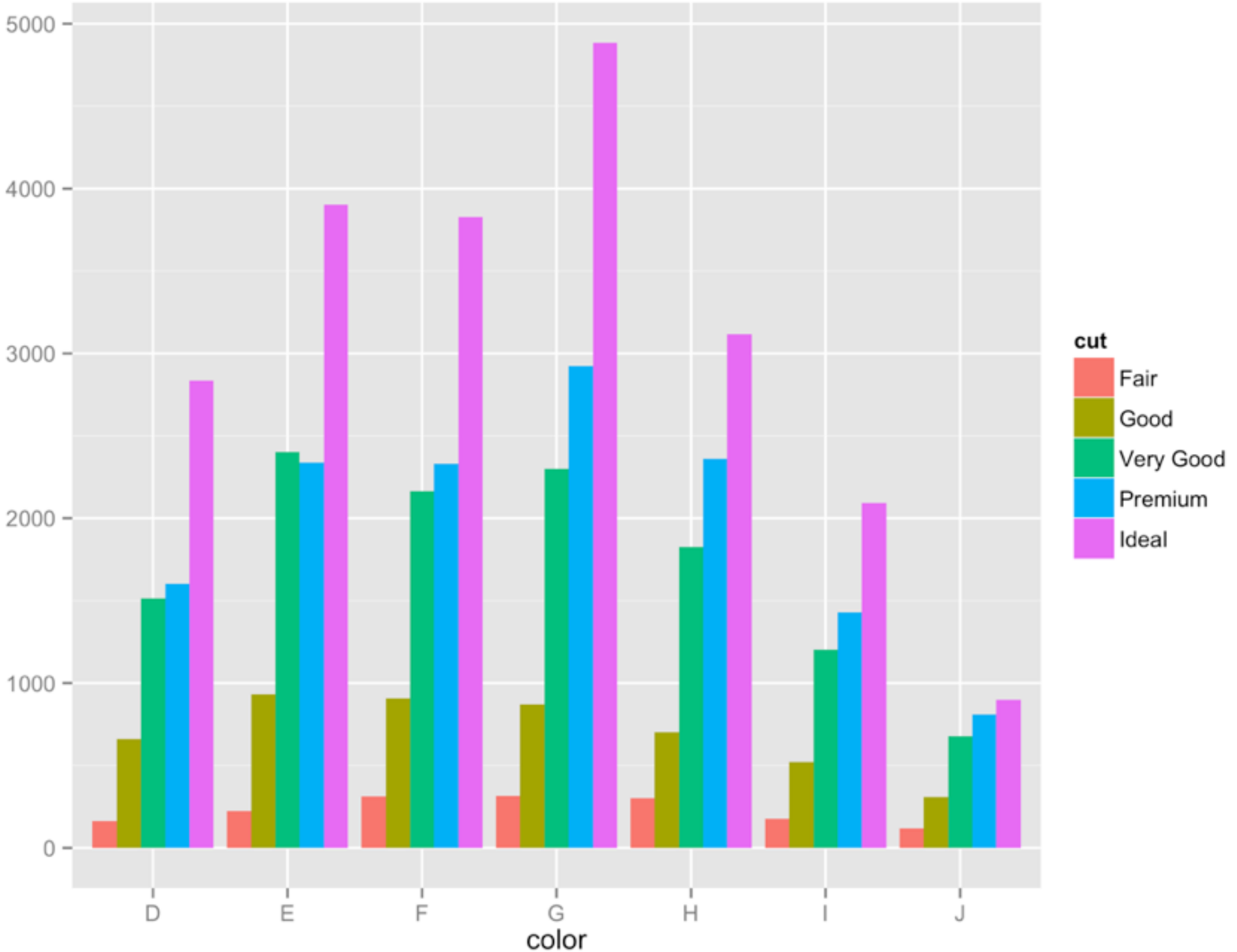
Axis labels

Modify axis labels with `xlab` and `ylab`

```
p + ylab("Price in Dollars ($)")
```




```
p + ylab("Price in Dollars ($)")
```



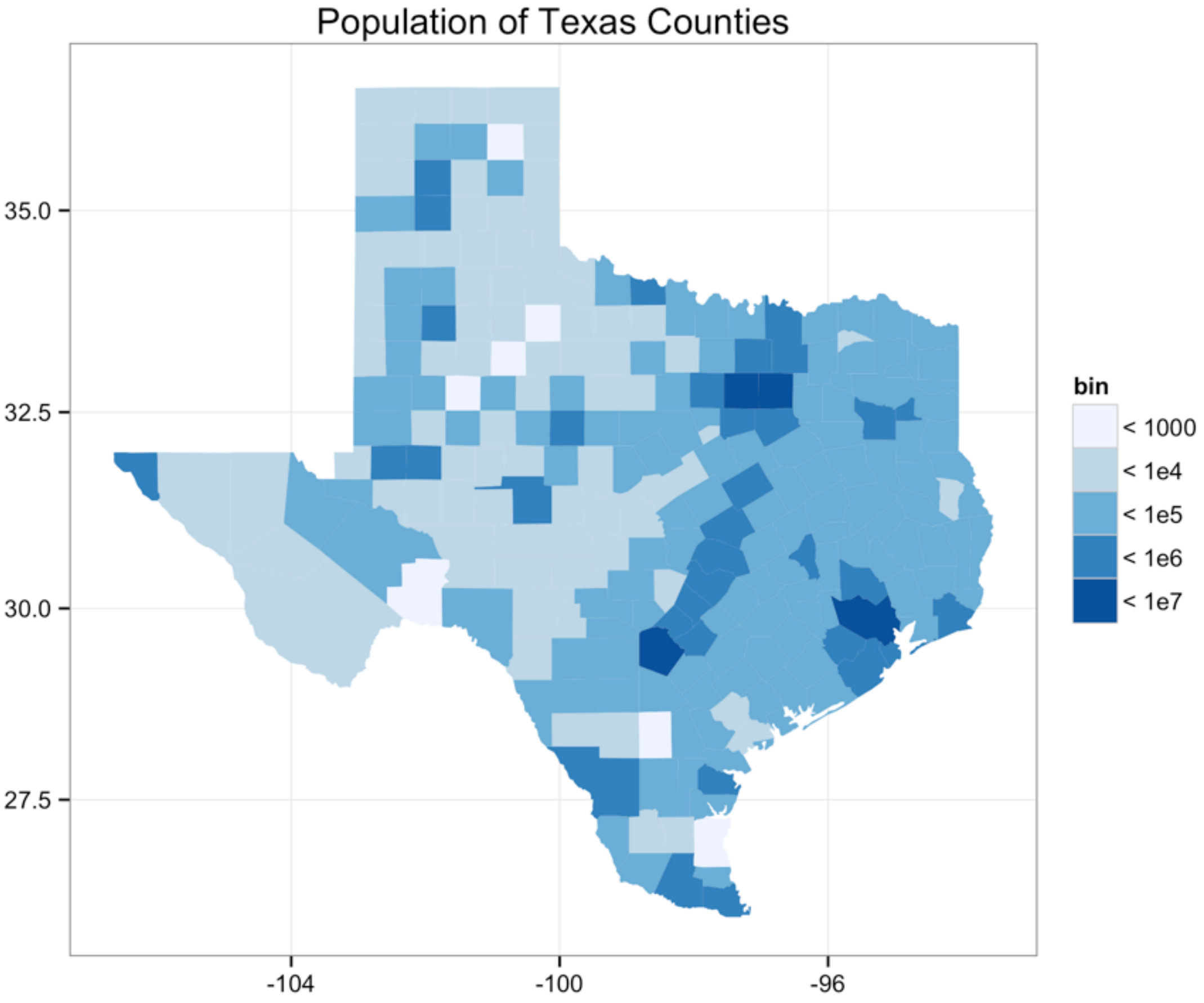

```
p + ylab("")
```

Your turn

Practice what you have learned so far on `tx`

1. remove the `long` and `lat` axis labels
2. add a white background
3. add a title
4. add a brewer color scale
5. an appropriate coordinate system

```
tx + scale_fill_brewer(palette = "Blues") +
  xlab("") +
  ylab("") +
  theme_bw() +
  coord_map() +
  ggtitle("Population of Texas Counties")
```

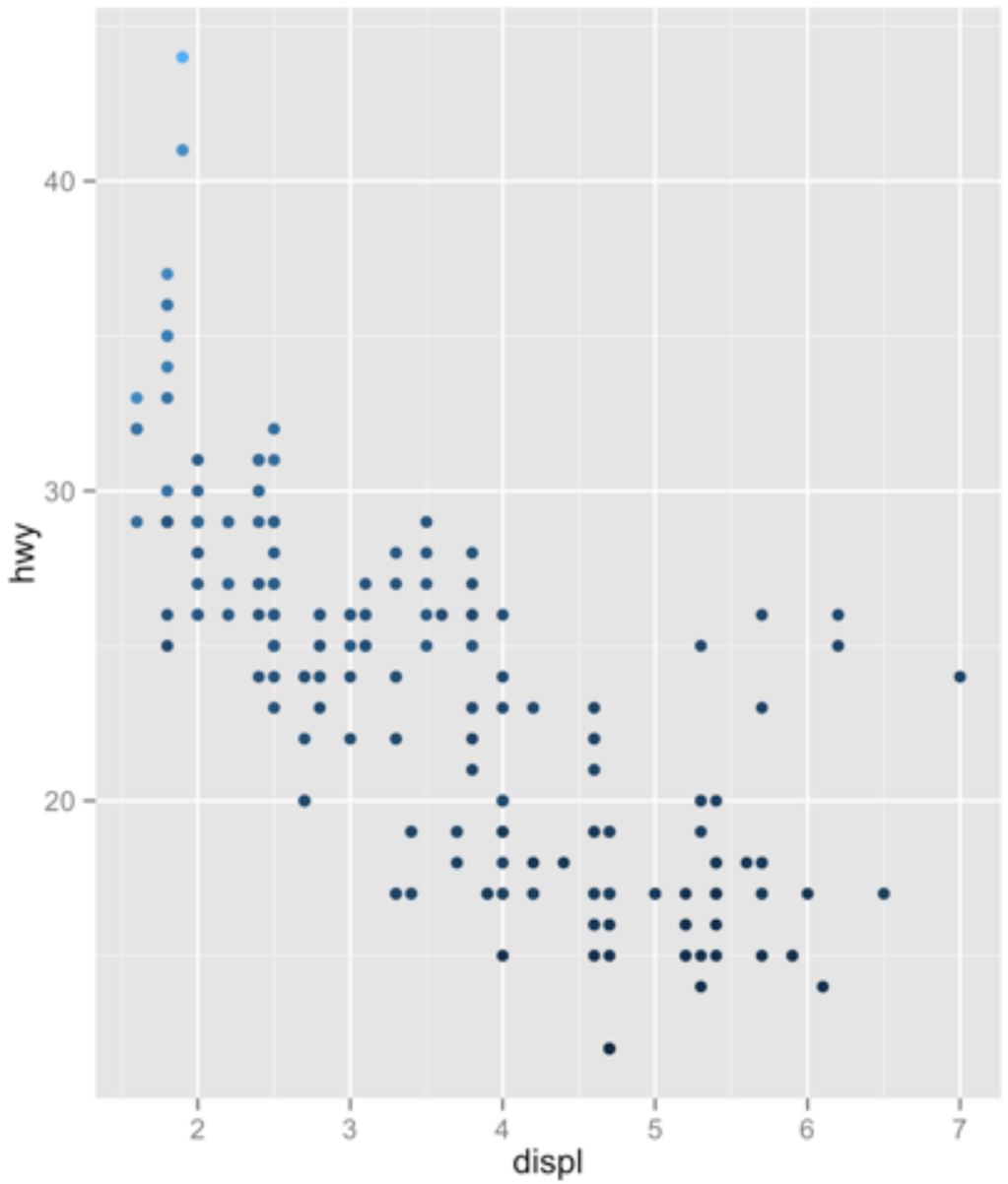


Legends

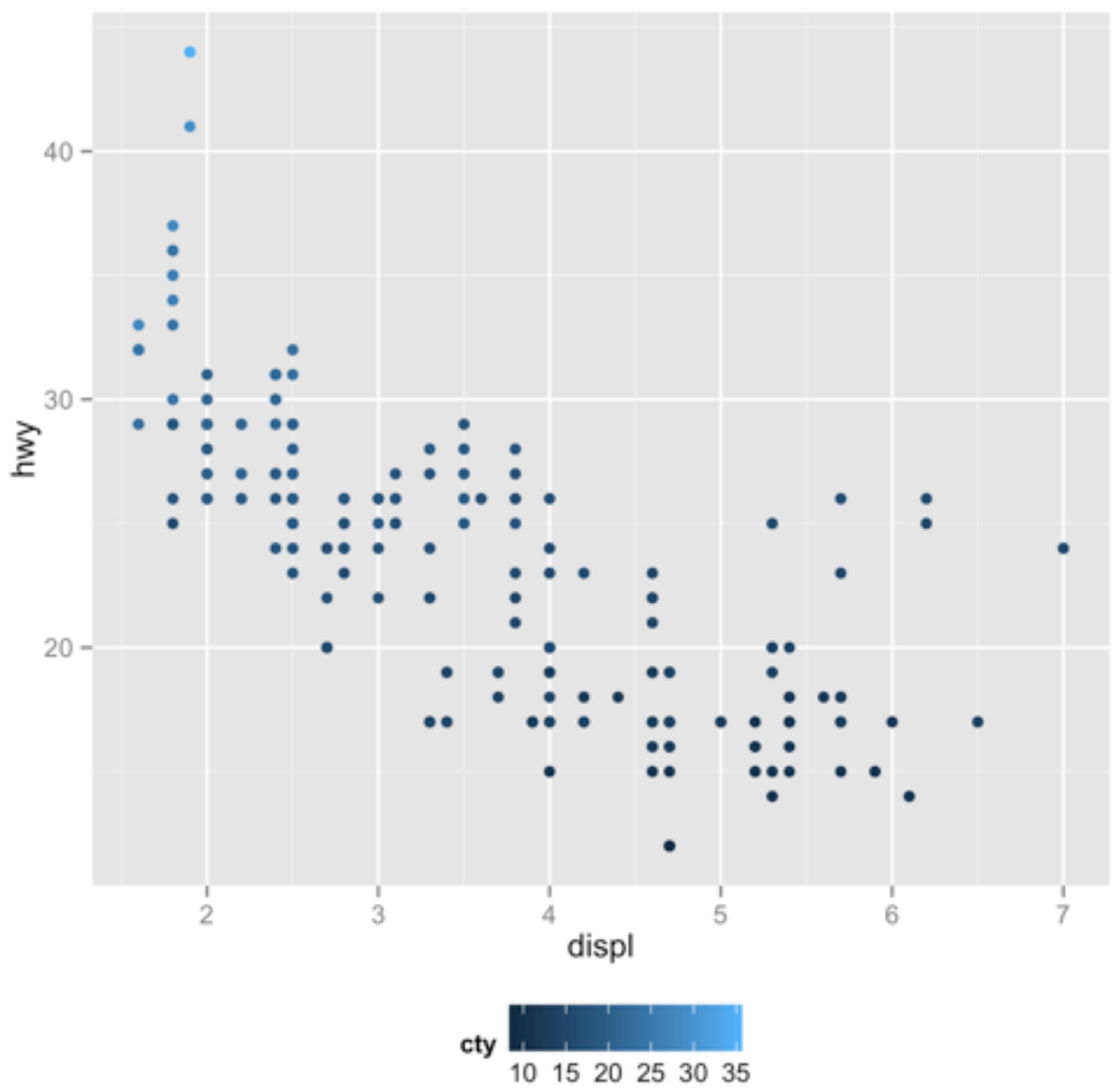
Control over the legends is decentralized.

```
q <- qplot(displ, hwy, data = mpg, color = cty)
```

Change position with `theme`



q




```
q + theme(legend.position = "bottom")
```

Legend position

Choose the position of legend with theme

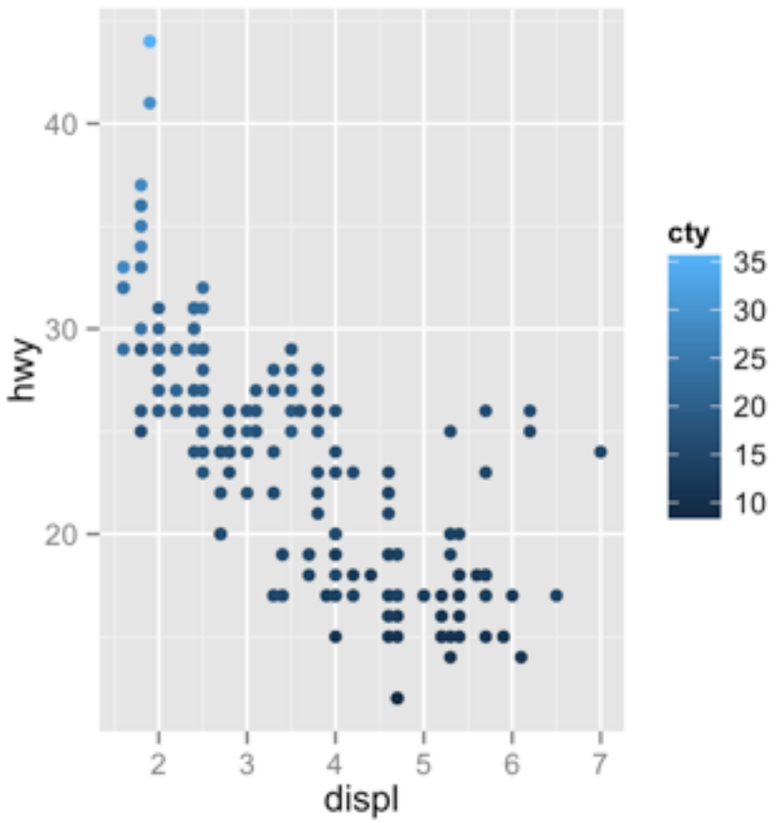
```
q + theme(legend.position = "bottom")
```

guides

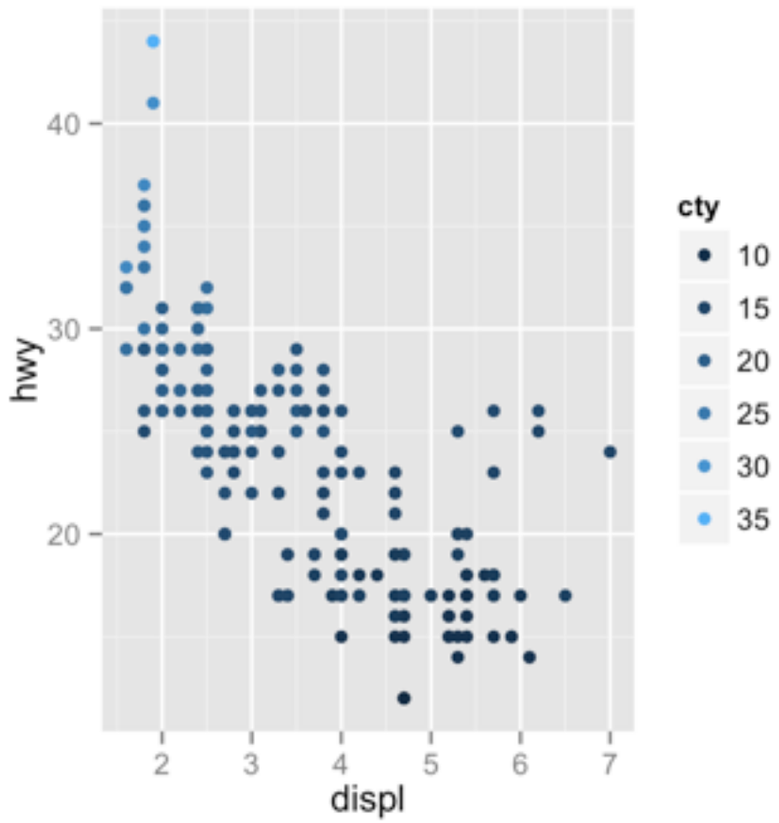
aesthetic

one of “bottom”, “top”, or “left”, or “right”

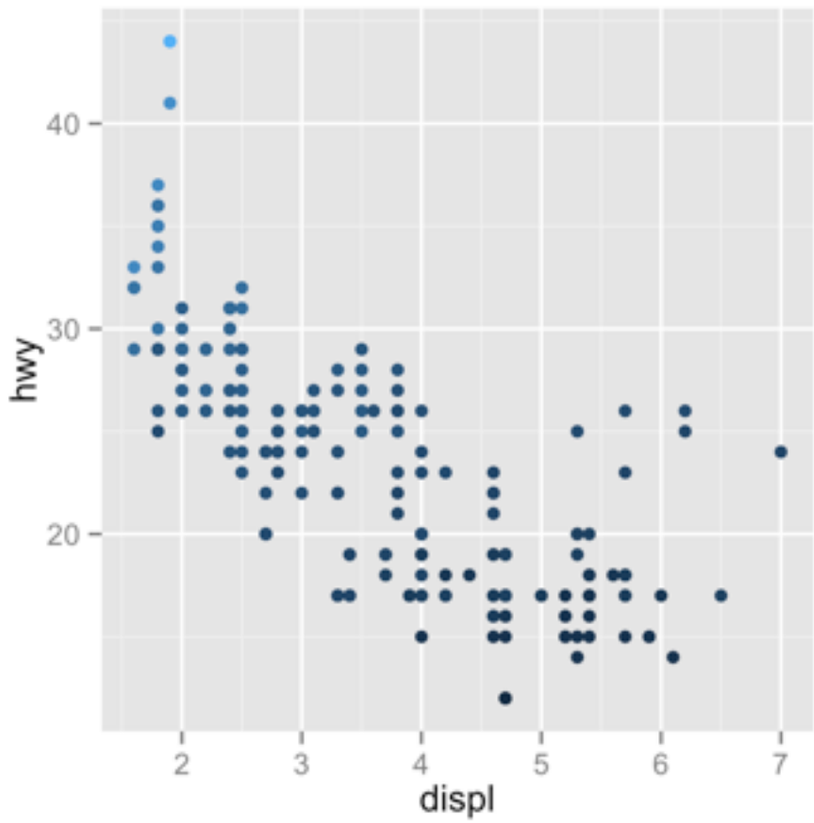
Change type with `guides`



`q + guides(color = "colorbar")`
(continuous values only)



`q + guides(color = "legend")`



`q + guides(color = "none")`

Legend type

Choose the type of legend for each aesthetic with guides

```
q + guides(color = "legend")
```

guides

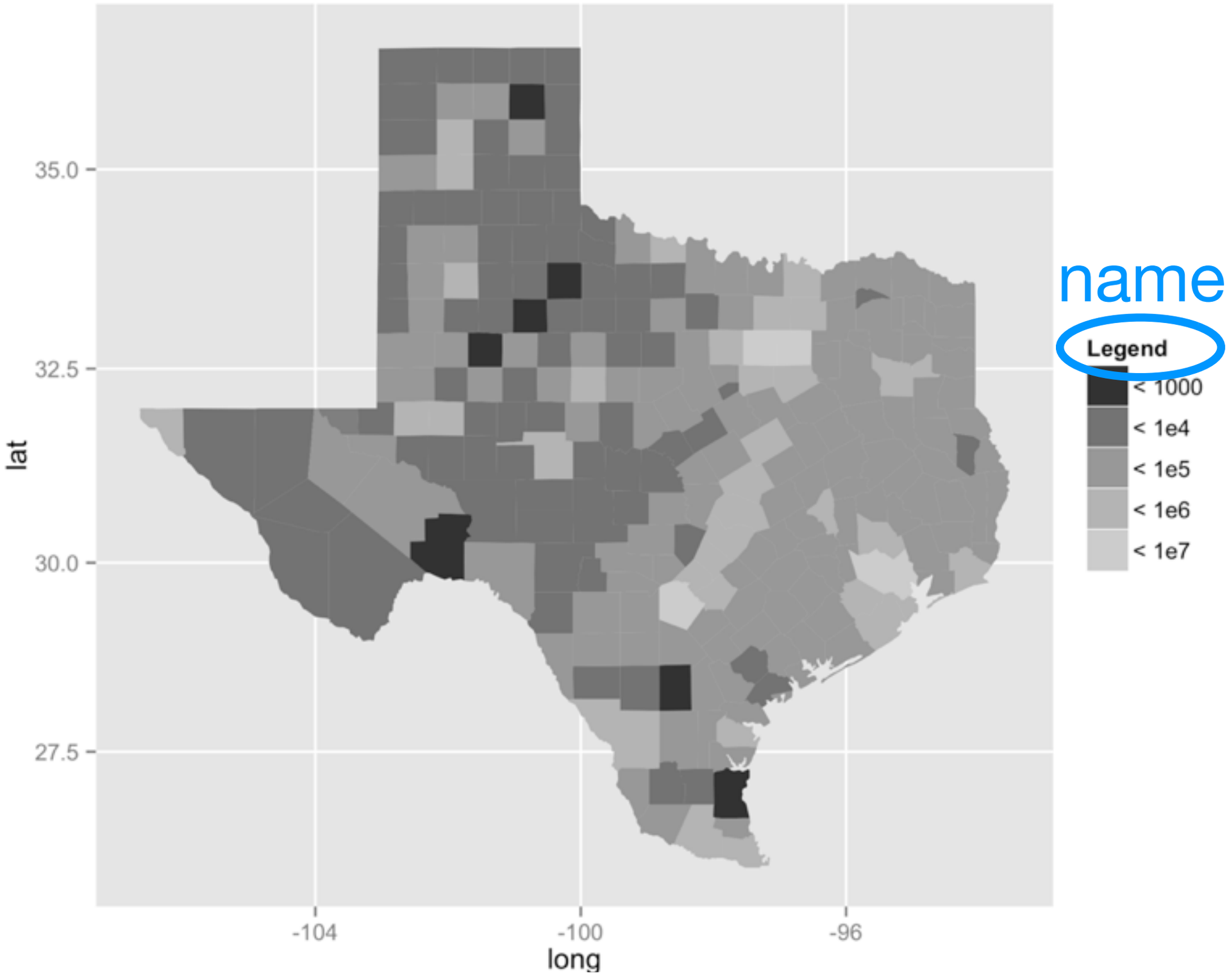
aesthetic

one of “legend”, “none”, or “colorbar” (for continuous values only)

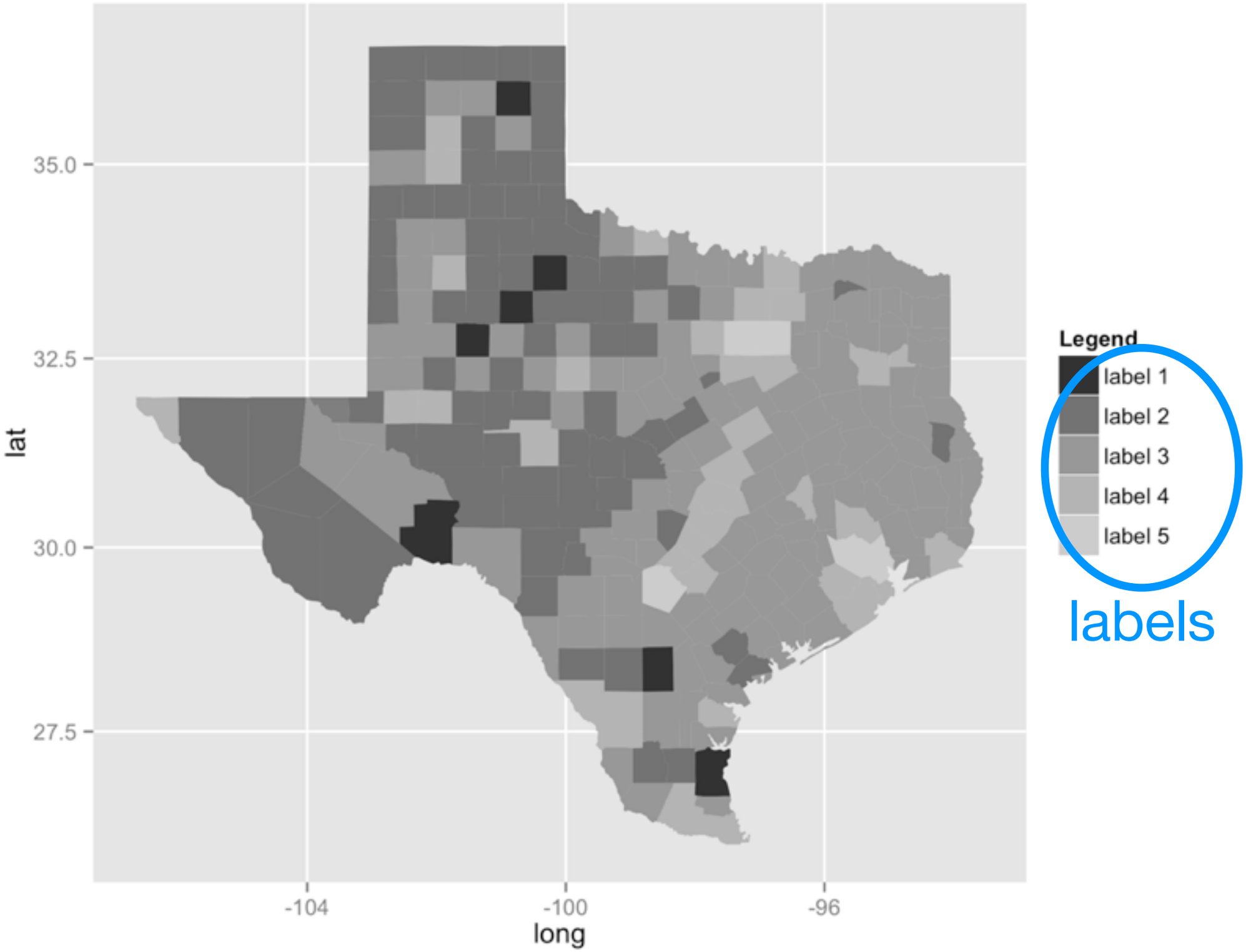
Change labels with `scales`

Every scale uses the following arguments

argument	controls
name	Title of legend (or axis label)
labels	Labels inside legend (or tick labels on axis) * must be a vector with one element for each label or tick mark




```
tx + scale_fill_grey(name = "Legend")
```



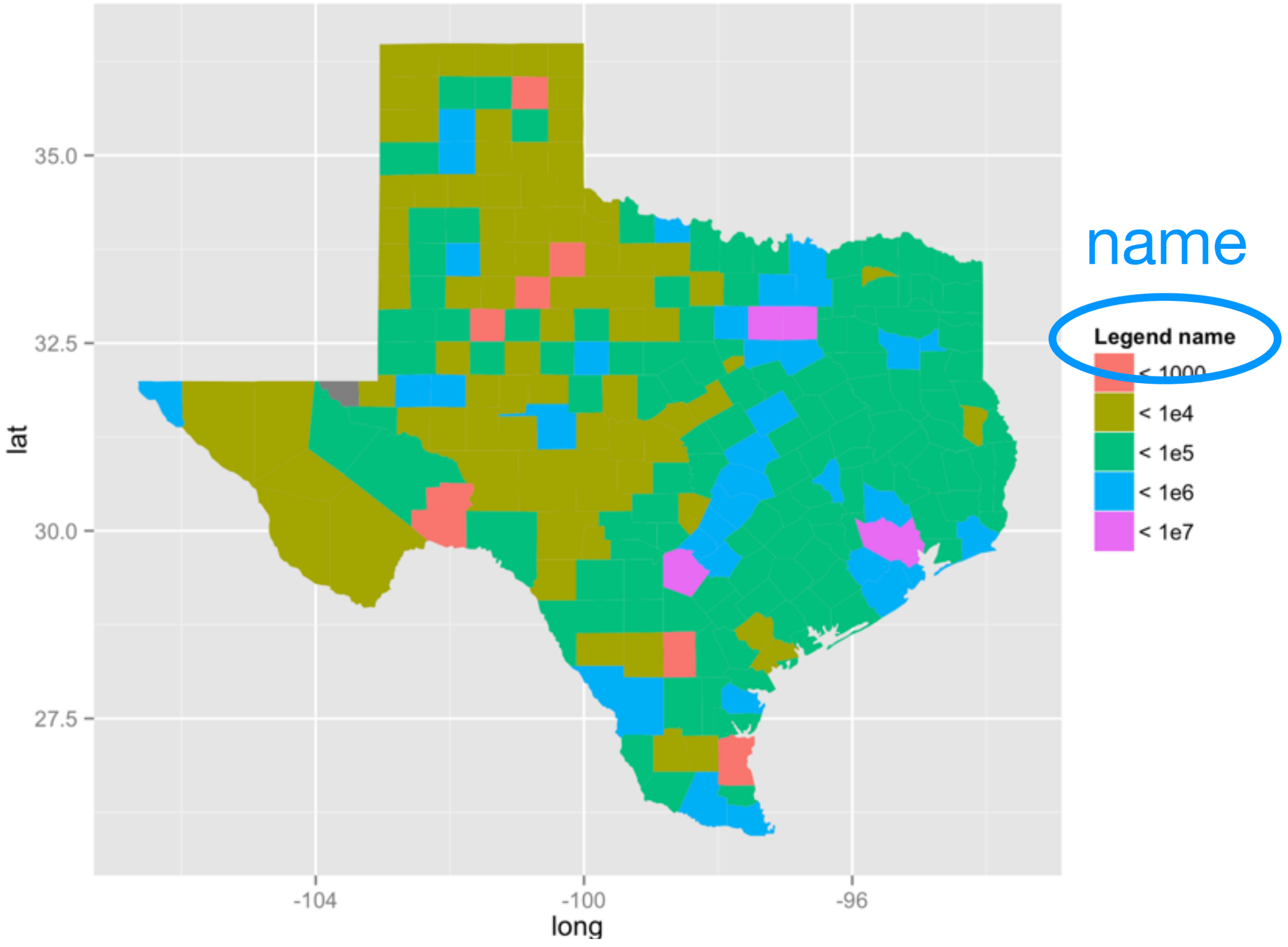

```
tx + scale_fill_grey(
  name = "Legend",
  labels = c("label 1", "label 2", "label 3","label 4", "label 5"))
```

Defaults

What if you want to change the legend, but don't want to switch to a new scale?

Use a continuous or discrete scale (depending on your data). These are the defaults - so they won't change the plot.

- scale_*aesthetic*_**continuous**, or
- scale_*aesthetic*_**discrete**



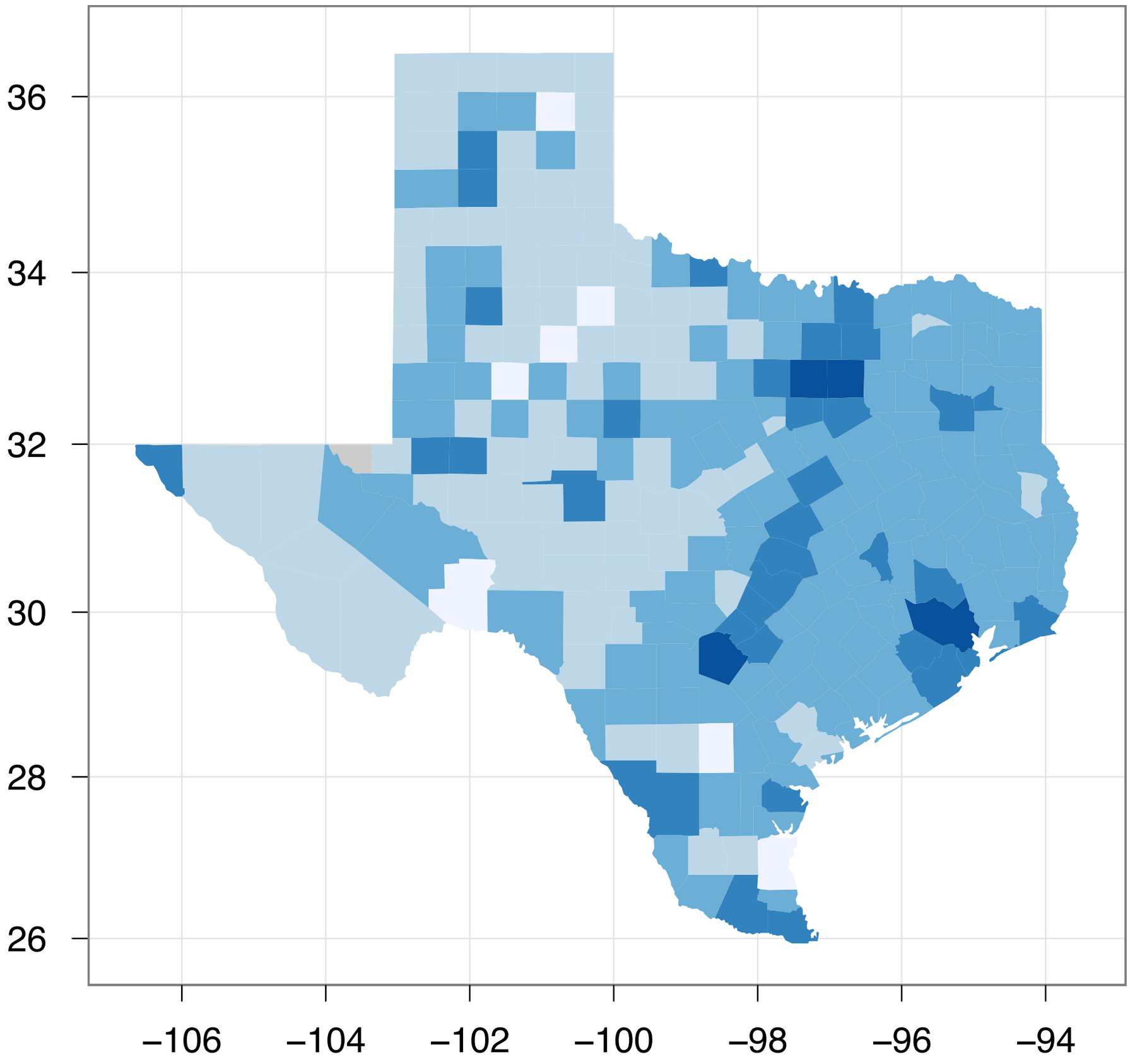

```
tx + scale_fill_discrete(name = "Legend name")
```

Your turn

Complete the Texas map you started in the last your turn by adding an informative title and set of labels to the legend.

```
tx + scale_fill_brewer(
  palette = "Blues",
  name = "Population",
  labels = c("0 - 999", "1,000 - 9,999",
    "10,000 - 99,999", "100,000 - 999,999",
    "1,000,000+")) +
xlab("") +
ylab("") +
theme_bw() +
coord_map() +
ggtitle("Population of Texas Counties")
```

Population of Texas Counties



Population

- 0 – 999
- 1,000 – 9,999
- 10,000 – 99,999
- 100,000 – 999,999
- 1,000,000+

Where to go from here

Index. ggplot2 0.9.2.1

docs.ggplot2.org/current/

ggplot2 0.9.2.1 Index

Help topics

Geoms

Geoms, short for geometric objects, describe the type of plot you will produce.

- `geom_abline`
  Line specified by slope and intercept.
- `geom_area`
  Area plot.
- `geom_bar`
  Bars, rectangles with bases on x-axis
- `geom_bin2d`
  Add heatmap of 2d bin counts.
- `geom_blank`
  Blank, draws nothing.
- `geom_boxplot`
  Box and whiskers plot.
- `geom_contour`
  Display contours of a 3d surface in 2d.
- `geom_crossbar`
  Hollow bar with middle indicated by horizontal line.
- `geom_density`
  Display a smooth density estimate.
- `geom_density2d`
  Contours from a 2d density estimate.
- `geom_dotplot`
  Dot plot

Dependencies

- **Depends**: stats, methods
- **Imports**: plyr, digest, grid, gtable, reshape2, scales, memoise, proto, MASS
- **Suggests**: quantreg, Hmisc, mapproj, maps, hexbin, maptools, multcomp, nlme, testthat
- **Extends**: sp

Learning ggplot2

ggplot2 mailing list
http://groups.google.com/group/ggplot2

stackoverflow
http://stackoverflow.com/tags/ggplot2

Cookbook for common graphics
http://wiki.stdout.org/rcookbook/Graphs/

ggplot2 book
http://amzn.com/0387981403

More useful packages

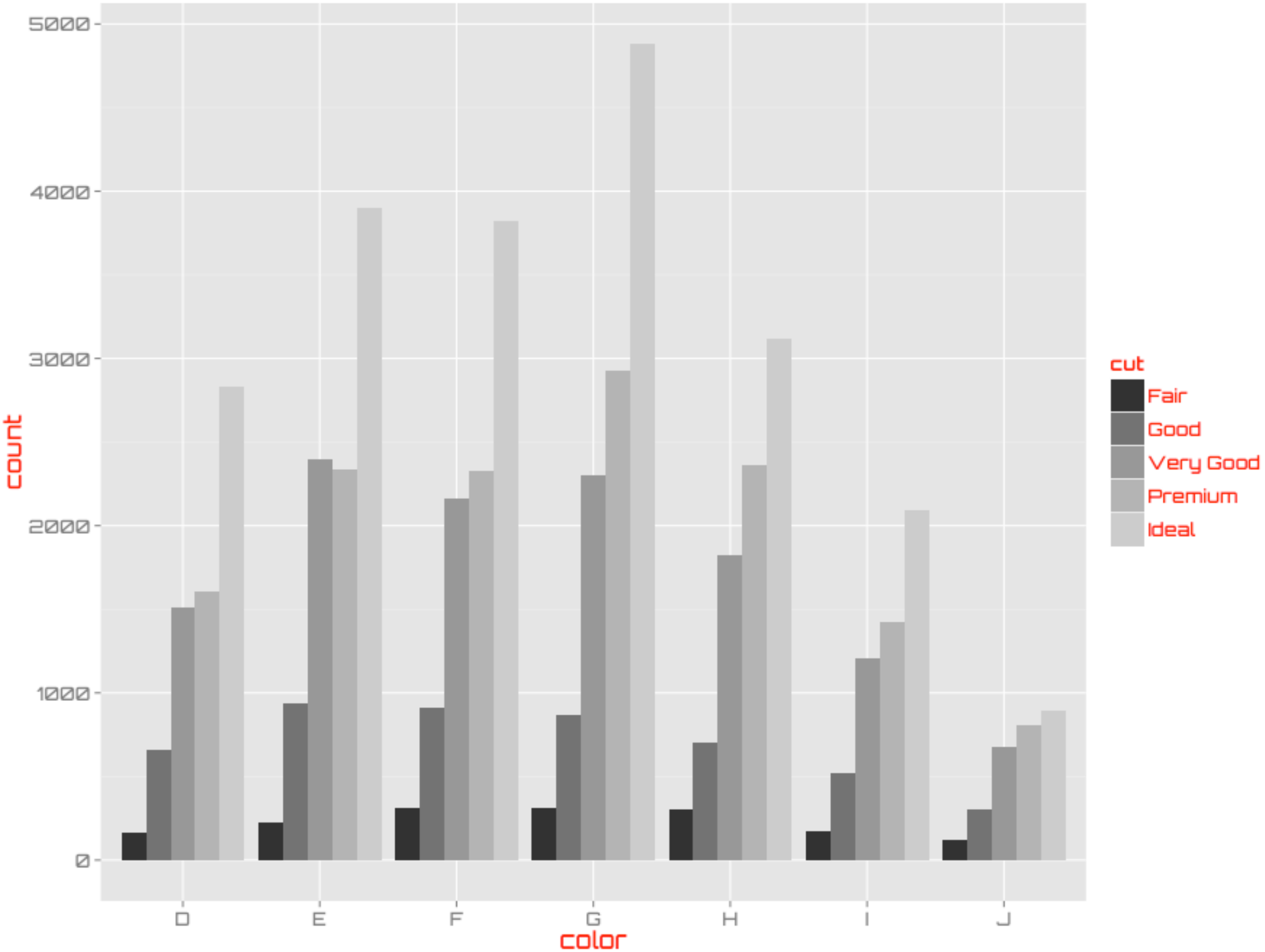
use system fonts in plots

extrafont

```
install.packages("extrafont")
```

https://github.com/wch/extrafont

Made with extrafont



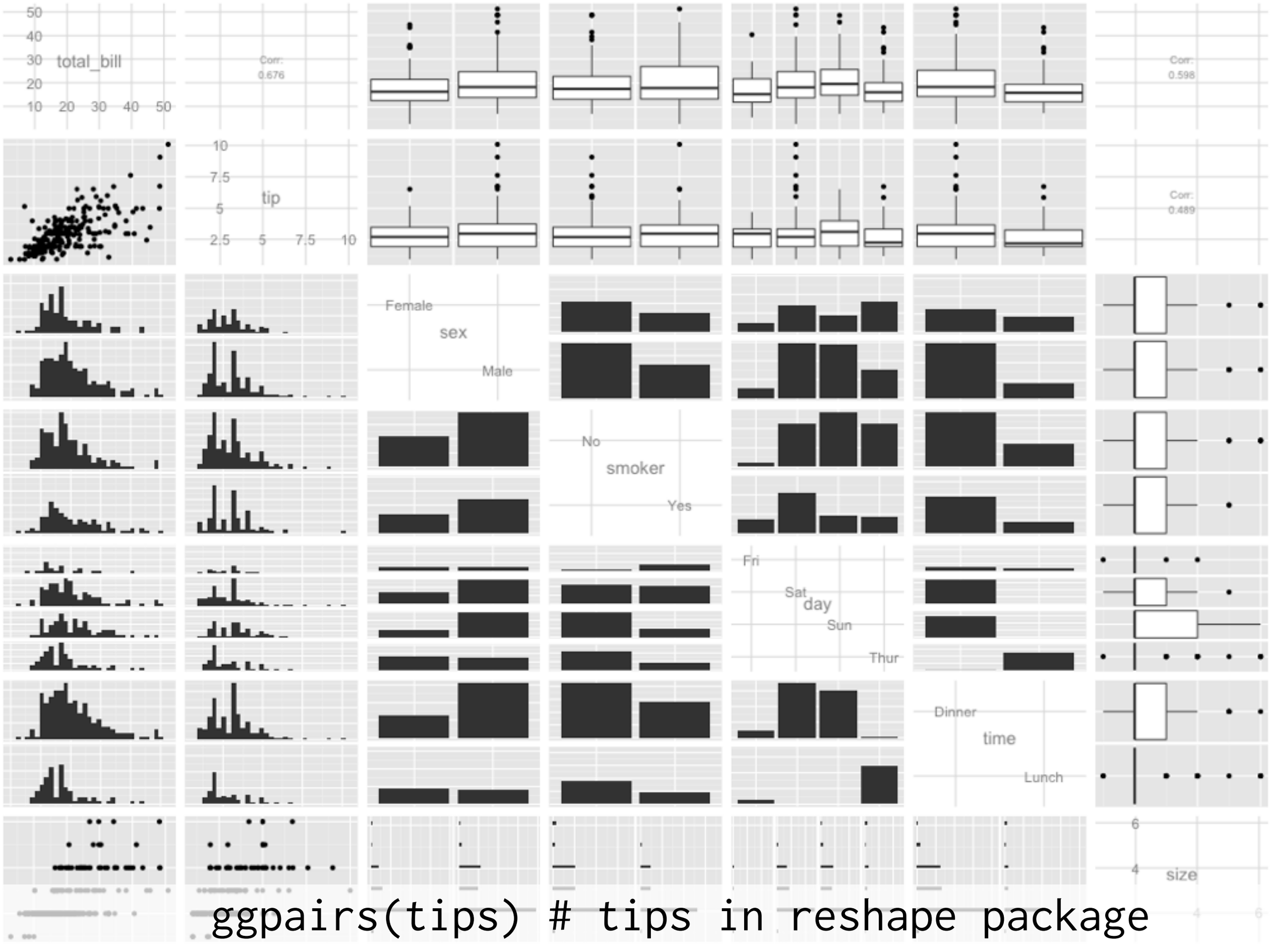
```
library(extrafont)
loadfonts()

p + scale_fill_grey() +
  ggtitle("Made with extrafont") +
  theme(text = element_text(size=16,
    family = "Orbitron",
    colour = "red"))

ggsave("fonttest-orbitron.pdf")
embed_fonts("fonttest-orbitron.pdf")
```

specialized plot types

GGally

```
install.packages("GGally")
```

https://github.com/ggobi/ggally



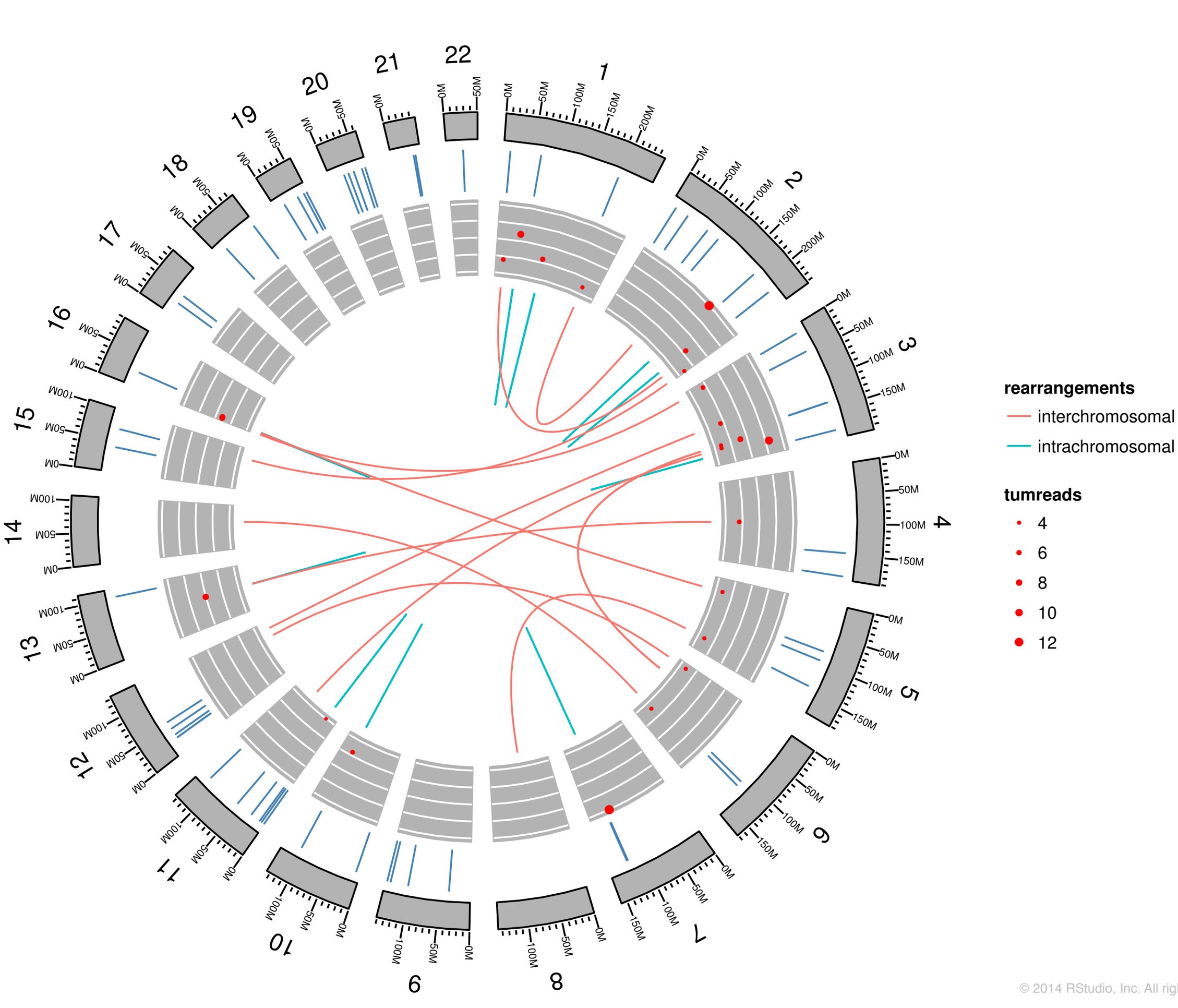
ggpairs(tips) # tips in reshape package

ggplot for genomics data

ggbio

```
source("http://bioconductor.org/biocLite.R")
biocLite("ggbio")
```

http://tengfei.github.com/ggbio/



graphical tools for Markov Chain Monte Carlo

ggmcmc

```
install.packages("ggmcmc")
```

http://xavier-fim.net/packages/ggmcmc/

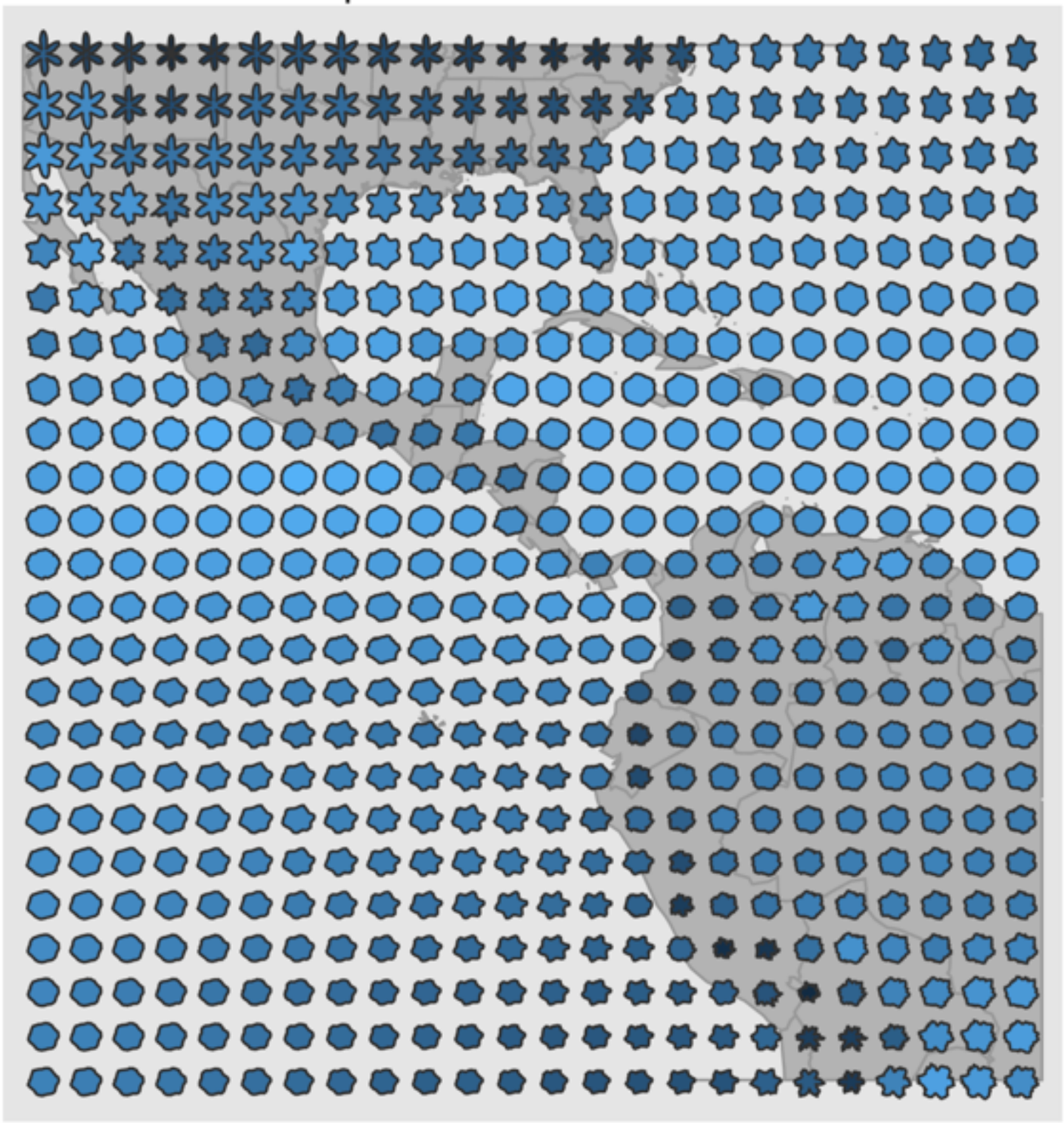
subplots and glyphs

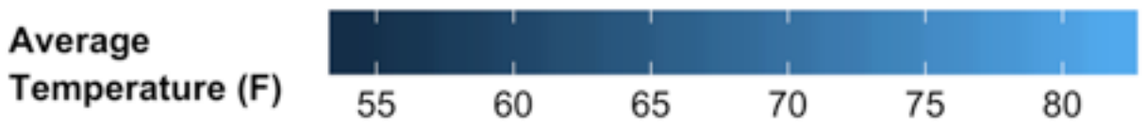
ggsubplot

```
install.packages("ggsubplot")
```

http://blog.revolutionanalytics.com/2012/09/visualize-complex-data-with-subplots.html

Surface temperature fluctuations 1995 - 2001





geom_subplot2d