Introduction to `ggplot2`

Statistical Consulting Group

Seminar for Statistics, ETH Zürich

Table of Contents

Getting Started with ggplot

In this section we will . . .

. . . get started with `ggplot2`

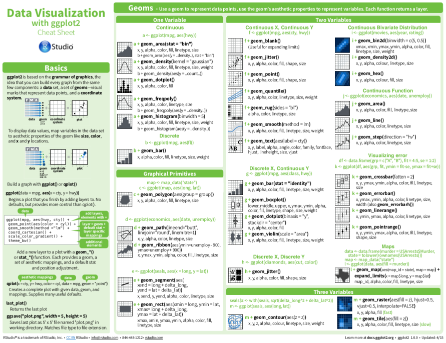
. . . create plots of one variable

. . . create plots of two variables

. . . learn how to save a plot

Must Have

Useful cheatsheet: https://www.rstudio.com/resources/cheatsheets/ (pick Data Visualisation with `ggplot2`)



Data Visualization
with ggplot2
Cheat Sheet

RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system.**

To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.

Build a graph with **ggplot()** or **qplot()**

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**
Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

```
ggplot(mpg, aes(hwy, cty)) +
 geom_point(aes(color = cyl)) +
 geom_smooth(method ="lm") +
 coord_cartesian() +
 scale_color_gradient() +
 theme_bw()
```

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot()
Returns the last plot

ggsave("plot.png", width = 5, height = 5)
Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous
a <- ggplot(mpg, aes(hwy))

a + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + **geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..count..))

a + **geom_dotplot()**
x, y, alpha, color, fill

a + **geom_freqpoly()**
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + **geom_histogram(**binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete
b <- ggplot(mpg, aes(fl))

b + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

map <- map_data("state")
c <- ggplot(map, aes(long, lat))

c + **geom_polygon(**aes(group = group)**)**
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + **geom_path(**lineend="butt", linejoin="round", linemitre=1**)**
x, y, alpha, color, linetype, size

d + **geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)**)**
x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

e + **geom_segment(**aes(xend = long + delta_long, yend = lat + delta_lat)**)**
x, xend, y, yend, alpha, color, linetype, size

e + **geom_rect(**aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y
f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank()**
(Useful for expanding limits)

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug(**sides = "bl"**)**
alpha, color, linetype, size

f + **geom_smooth(**method = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text(**aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y
g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar(stat = "identity")**
x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + **geom_dotplot(**binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

g + **geom_violin(**scale = "area"**)**
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y
h <- ggplot(diamonds, aes(cut, color))

h + **geom_jitter()**
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution
i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d(**binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill, size

Continuous Function
j <- ggplot(economics, aes(date, unemploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

j + **geom_line()**
x, y, alpha, color, linetype, size

j + **geom_step(**direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizing error
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

k + **geom_crossbar(**fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps
data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

l + **geom_map(**aes(map_id = state), map = map**)** + **expand_limits(**x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

Three Variables

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

m + **geom_contour(**aes(z = z)**)**
x, y, z, alpha, colour, linetype, size, weight

m + **geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill (fast)

m + **geom_tile(**aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size (slow)

Learn more at docs.ggplot2.org • ggplot2 1.0.0 • Updated: 4/15

data
geom
coordinate system
plot
add layers, elements with +
layer = geom + default stat + layer specific mappings
additional elements
aesthetic mappings

Source: link above. This image is under Creative Commons licence.

Book Recommendations: R Graphics & `ggplot`

R Graphics Cookbook

Winston Chang, O'Reilly Media, 2012

and its online companion:

http://www.cookbook-r.com/Graphs/

ggplot2: Elegant Graphics for Data Analysis (Use R!)

Hadley Wickham, Springer, 2009

See also: https://ggplot2.tidyverse.org/

Why ggplot2?

Some advantages:

- nice labels
- nice colors
- small margins
- beautiful faceting or multipanel plots
- very powerful and flexible: we will have a glimpse at the grammar of graphics
- can easily change or update plots

Why ggplot2?

Some disadvantages:

- `ggplot2` can only deal with `data.frames`
- default plots of model outputs are usually not possible
- `ggplot2` is not optimized for speed performance
- 3D plots are not possible

Functions in Package ggplot2

There are **two important functions**:

- `qplot`: similar to `base` plotting functions (“for beginners”)
- `ggplot`: the feature-rich “workhorse” (**our focus**)

Grammar of Graphics

The ”gg” in `ggplot2` stands for grammar of graphics which is based on Wilkinson’s (2005) grammar of graphics.

The grammar is useful because . . .

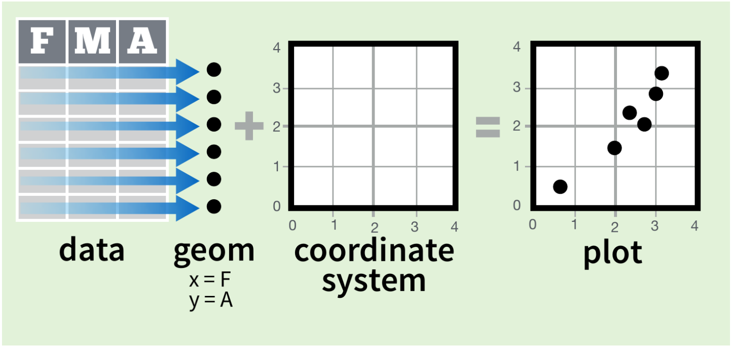
- it is a generic way of creating a plot
- it does not rely on a specific or customized graphic for a particular problem
- it allows for iterative updates of a plot
- it uses the concepts of layers

Grammar of Graphics

Idea: all plots can be built from the same components

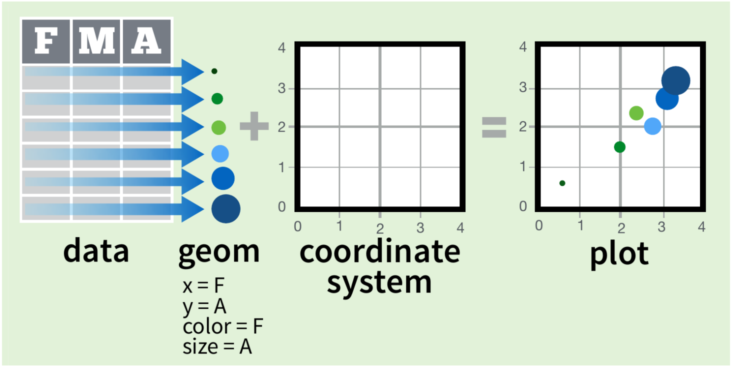
- **data** set
- **coordinate system**
- **aesthetic mapping** that describes how information in data is being mapped to visual properties (aesthetics) of geometric objects, so called **geoms**.

Grammar of Graphics



Source: https://www.rstudio.com/resources/cheatsheets/

Grammar of Graphics



Source: https://www.rstudio.com/resources/cheatsheets/

Overview: Plots of One Variable

One **continuous** variable:

- histogram: `geom_histogram()`
- densities: `geom_density()`
- frequency plot: `geom_freqpoly()`

One **discrete** (categorical) variable:

- barplot: `geom_bar()`
- pie plot: different coordinate system of barplot . . .

Illustration with mpg Data Set

Let us use the `mpg` data set from `ggplot2`.

It contains 234 observations about the fuel efficiency of 38 popular cars in 1999 and 2008.

Let's have a look at the democode.

Overview: Plots of Two Variables

Two continuous variables

- scatter plot: `geom_point()`
- scatter plot using jitter: `geom_jitter()`
- smoother: `geom_smooth()`

Discrete x and continuous y

- boxplot: `geom_boxplot()`
- bar plot: `geom_bar(stat = "identity")`

Continuous function like time series

- line plot: `geom_line()`

Illustration with mpg Data Set

Let's look at the democode.

How to Save a Plot?

First we create an R object with the corresponding plot:

```
> v <- ggplot(data = mpg, aes(x = class, y = cty)) + geom_boxplot() +
+   geom_jitter(alpha = 0.3)
```

Plots can then be saved by `ggsave()`:

```
> ggsave(filename = "cool-boxplot-II.png", plot = v)
```

`ggsave` automatically recognizes the output format (pdf, png, jpg, eps, svg)!

How to Save a Plot?

Control the width & height and change the path:

```
> ggsave(filename = "cool-boxplot-III.jpg", plot = v, width = 5, height = 4,
+        path = "/path/of/figures/")
```

Alternatively, don't forget to `print` the plot:

```
> pdf("cool-boxplot-IV.pdf")
> print(v)
> dev.off()
```

Aesthetics

In this section we will have a look at the aesthetics . . .

. . . size

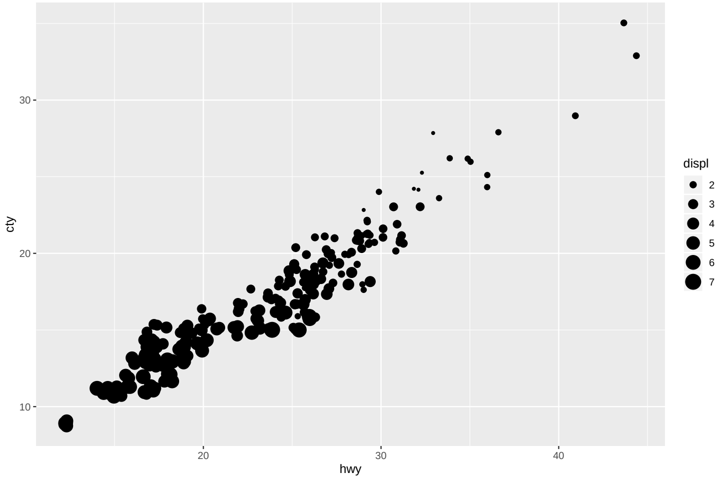
. . . shape

. . . color

. . . and combine them

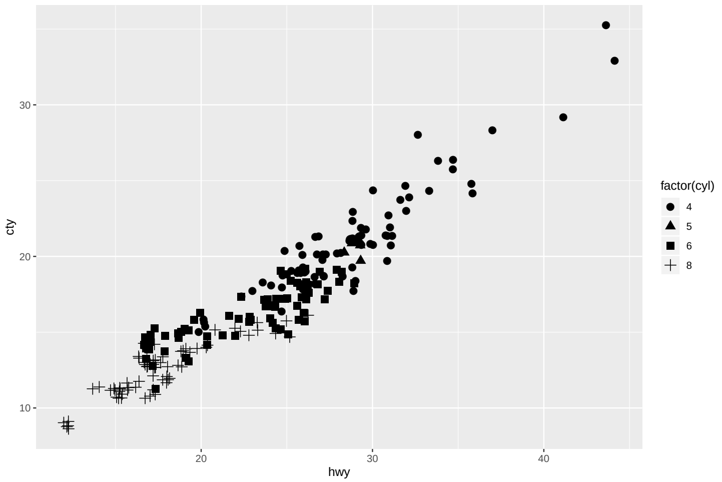
Aesthetics: size

```
> ggplot(data = mpg, aes(x = hwy, y = cty, size = displ)) +
+   geom_jitter()
> # displ: engine displacement, in litres
```



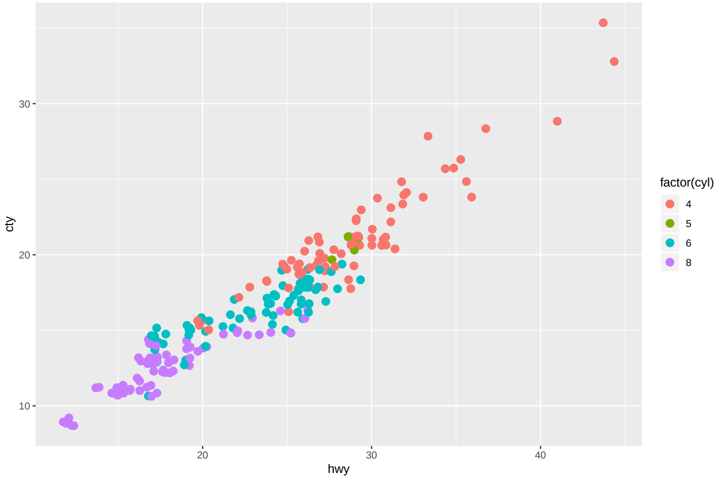
Aesthetics: shape

```
> ggplot(data = mpg, aes(x = hwy, y = cty, shape = factor(cyl))) +
+   geom_jitter(size = 3)
> # we can set a fixed size for all the points
```



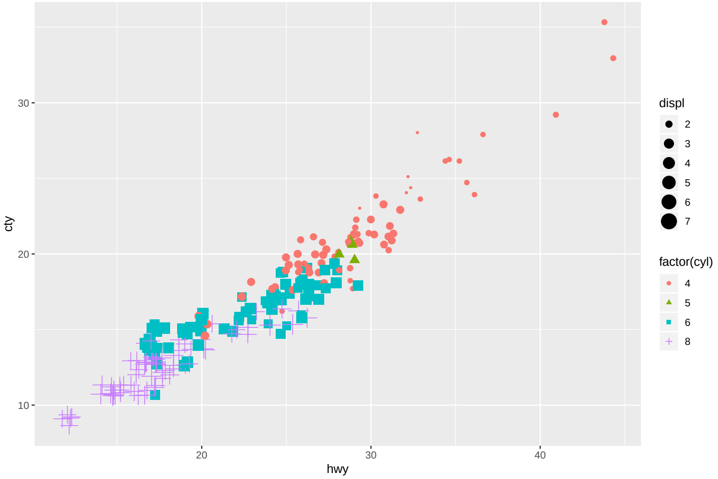
Aesthetics: color

```
> ggplot(data = mpg, aes(x = hwy, y = cty, color = factor(cyl))) +
+   geom_jitter(size = 3)
```



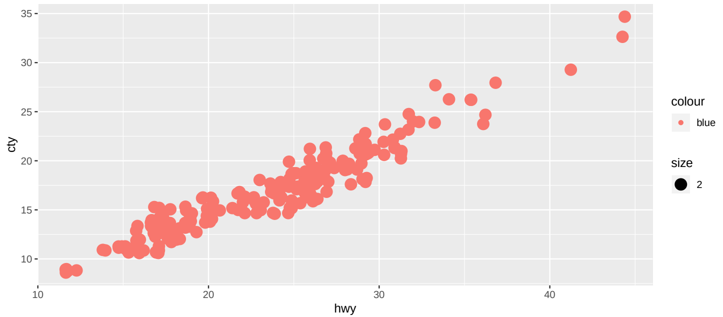
Aesthetics: Combination

```
> ggplot(data = mpg, aes(x = hwy, y = cty, color = factor(cyl),
+                        shape = factor(cyl), size = displ)) +
+   geom_jitter()
> # there is only one combined legend for shape and color
```



Aesthetics: Setting vs. Mapping

```
> ggplot(data = mpg, aes(x = hwy, y = cty, color = "blue", size = 2)) +
+   geom_jitter()
```



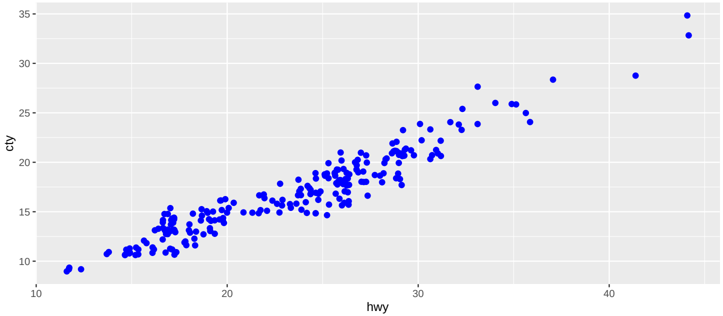
The `color` argument in `aes` will create a new variable with a single entry `"blue"` that is **mapped** to `color` (getting the *first* default color). Similarly for `size`.

In addition, a legend is being created.

Aesthetics: Setting vs. Mapping

If we want to set color to the explicit value "blue", we can do this in the corresponding layer (*outside* of aes). Similarly for size.

```
> ggplot(data = mpg, aes(x = hwy, y = cty)) +
+   geom_jitter(color = "blue", size = 2)
```



Layers

In this section we will look at. . .

. . . where to place which arguments.

. . . the order of layers.

Layers: Where to Place Which Arguments?

- All arguments specified in function `ggplot()` are passed to all subsequent layers.
- This holds true unless a layer contains another specification.
- Arguments specified in a single layer only affect the corresponding layer.

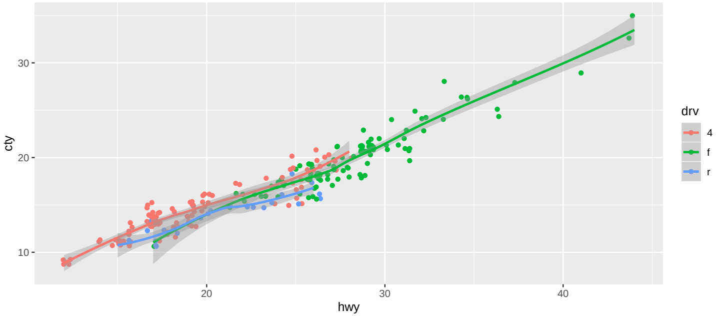
Layers: Where to Place Which Arguments?

Basic plot to start with

```
> ggplot(data = mpg, aes(x = hwy, y = cty)) + geom_jitter() + geom_smooth()
```

Color both points and smoothers per group of `drv`
⇒ **three** smoothers are fitted:

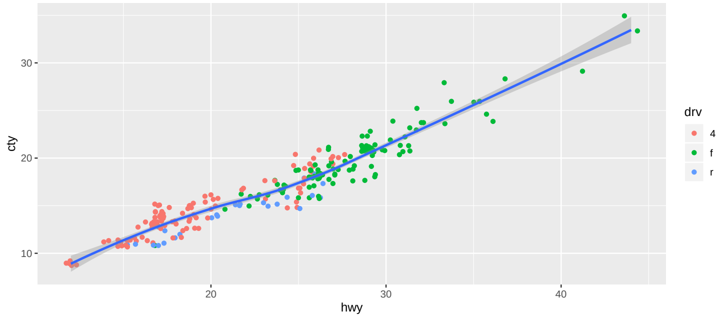
```
> ggplot(data = mpg, aes(x = hwy, y = cty, color = drv)) + geom_jitter() +
+   geom_smooth()
```



Layers: Where to Place Which Arguments?

Color the points per group of `drv`
⇒ **one** smoother is fitted:

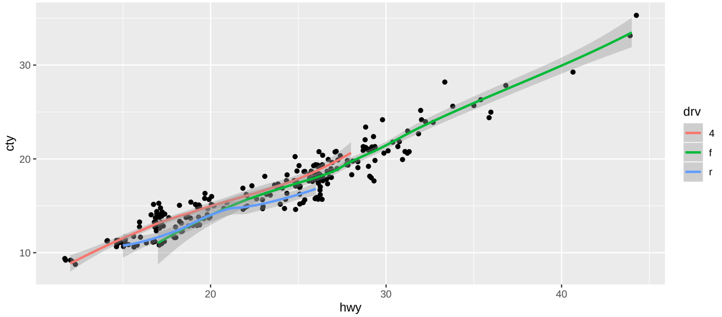
```
> ggplot(data = mpg, aes(x = hwy, y = cty)) + geom_jitter(aes(color = drv)) +
+   geom_smooth()
```



Layers: Where to Place Which Arguments?

Color the smoothers per group `drv`
⇒ **three** smoothers are fitted:

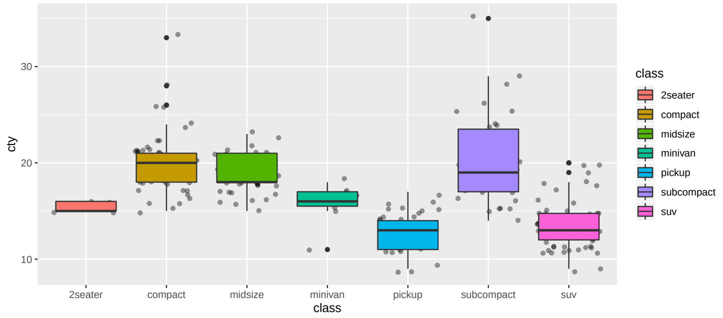
```
> ggplot(data = mpg, aes(x = hwy, y = cty)) + geom_jitter() +
+   geom_smooth(aes(color = drv))
```



Layers: Order of Layers

Plot the points first and then add the layer with boxplots:

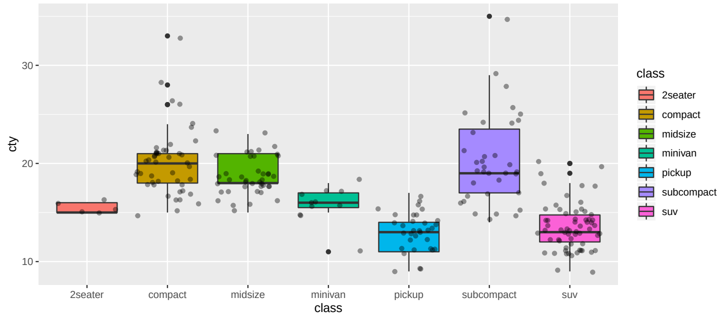
```
> ggplot(data = mpg, aes(x = class, y = cty)) +
+   geom_jitter(alpha = 0.4) +
+   geom_boxplot(aes(fill = class))
```



Layers: Order of Layers

Plot the boxplot first and afterwards add the layer of points:

```
> ggplot(data = mpg, aes(x = class, y = cty)) +
+   geom_boxplot(aes(fill = class)) +
+   geom_jitter(alpha = 0.4)
```



Faceting

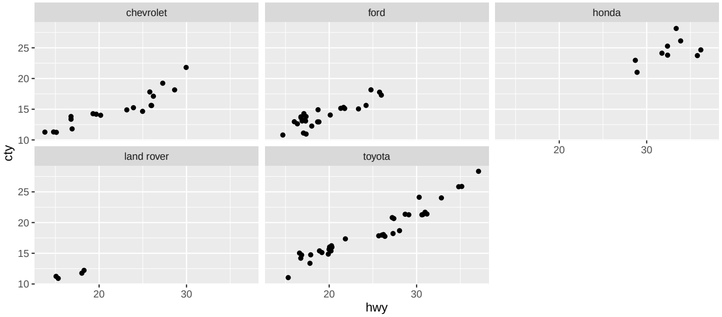
In this section we will . . .

. . . consider faceting or multi-panel conditioning plots

Faceting: facet_wrap

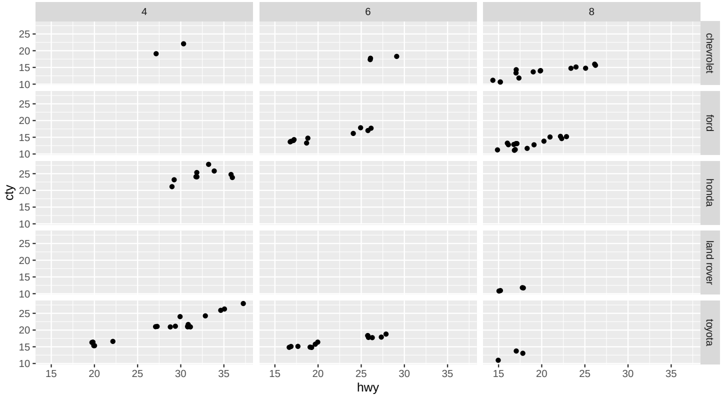
Let's look a multi-panel plots

```
> # subset of the mpg data set
> mpg.small <- subset(mpg, manufacturer %in%
+                       c("ford", "land rover", "toyota",
+                         "chevrolet", "honda"))#, "volkswagen"))
> ggplot(data = mpg.small, aes(x = hwy, y = cty)) +
+   geom_jitter() + facet_wrap(~ manufacturer)
```



Faceting: `facet_grid`

```
> ggplot(data = mpg.small, aes(x = hwy, y = cty)) +
+   geom_jitter() + facet_grid(manufacturer ~ cyl)
```



Changing Colors or the Theme

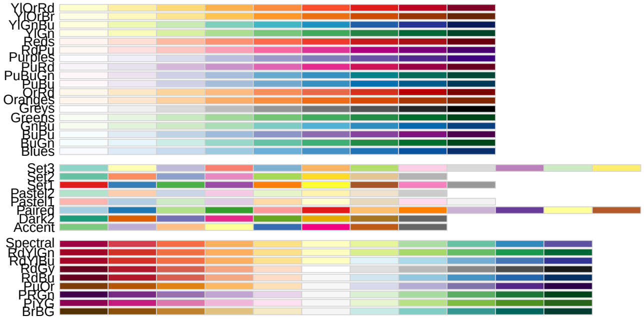
In this section we will look at . . .

. . . how to select colors

. . . themes

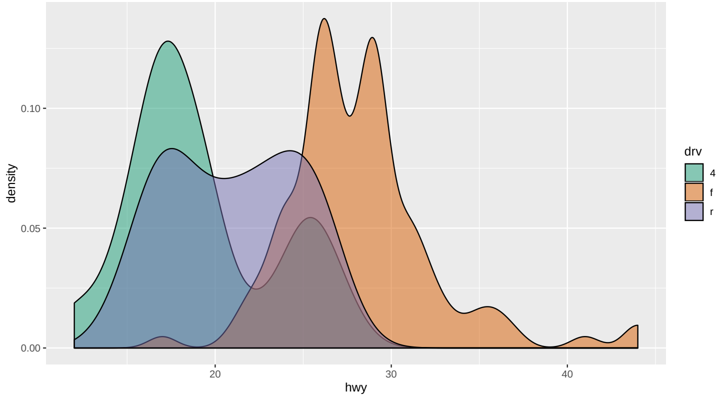
How to Select Colors?

```
> require(RColorBrewer)
> display.brewer.all()
```



How to Select Colors?

```
> # use brewer
> ggplot(data = mpg, aes(x = hwy, fill = drv)) + geom_density(alpha = 0.5) +
+   scale_fill_brewer(palette = "Dark2")
```



How to Select Colors?

Websites helping you to select colors
http://colorbrewer2.org/
http://tools.medialab.sciences-po.fr/iwanthue/

Define your own colors

```
> ggplot(data = mpg, aes(x = hwy, fill = drv)) +
+   geom_density(alpha = 0.5) +
+   scale_fill_manual(values = c("red", "green", "black"))
```

Themes

Change the theme of a plot using `theme_...()`.

Let's have a look at the democode.

```
> ggplot(data = mpg, aes(x = class, y = cty)) +
+   geom_boxplot(aes(fill = class)) +
+   geom_jitter(alpha = 0.4) +
+   theme_bw()
```

See the R package `ggthemes` for additional themes.