An Introduction to ggplot2

Laurel Stell

May 1, 2019

Stanford MEDICINE | Data Studio

Introduction

To run the examples yourself

1. http://web.stanford.edu/~lstell/
2. Links at bottom of page to R Markdown file and slides

May also need to install the following packages:

```
library(ggplot2)
library(gridExtra)
library(plyr)
library(reshape2)
```

Objectives

- How to make some common types of plots with **ggplot2**
- Demonstrate the paradigm for advanced customizations
- Tips on learning more

Prerequisites

Basic knowledge of R:

- Factors, data frames, etc
- Installing and loading packages
- Base graphics functions such as **plot**

Note: **ggplot2** is based on **grid** package. Do *not* mix with base graphics such as par(), split.screen(), axis(), legend().

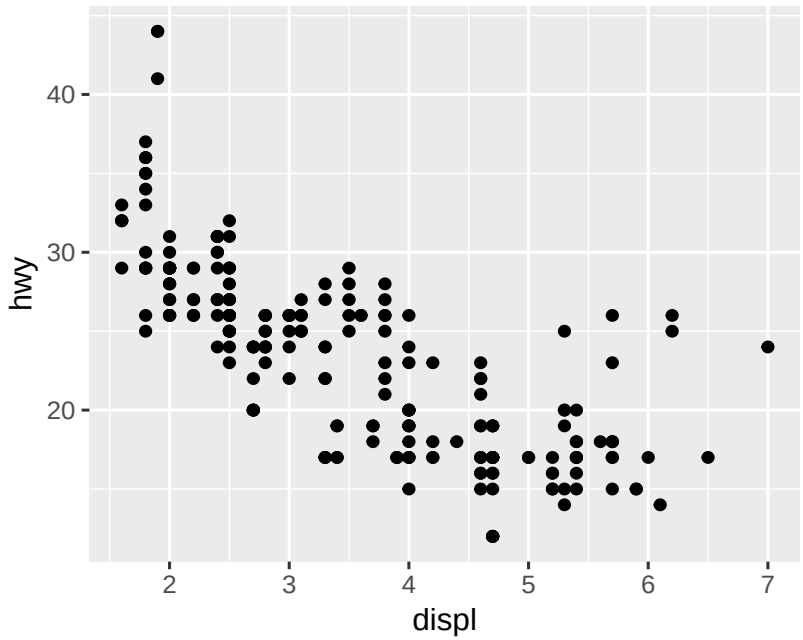
First Steps

The paradigm

- **ggplot2** is part of tidyverse by Hadley Wickham
- Syntax is based on *The Grammar of Graphics* by Leland Wilkinson
  - $140 new on Amazon
  - Available online for Stanford affiliates at https://ebookcentral.proquest.com/lib/stanford-ebooks/detail.action?docID=302755
  - But I've never looked at this book.
- Basically, build sentences:
  - Call **ggplot()** to specify data
  - Add "layers" with **geom_point()**, **geom_histogram()**, etc
  - Add customizations

Example: simple scatter plot

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



Basic plot specification

- First argument to **ggplot()** is a data frame
  - **mpg** is actually a tibble (**tbl_df**)
  - Tibbles are part of tidyverse
  - Tibbles inherit from data.frame
- Columns in data frame referred to simply by name in remainder of **ggplot2** sentence
- Aesthetics use data:
  - x and y axes
  - Color, shape, size, etc
  - Grouping
  - Etc

Jitter

Some points on top of each other:

```
nrow(mpg)
```

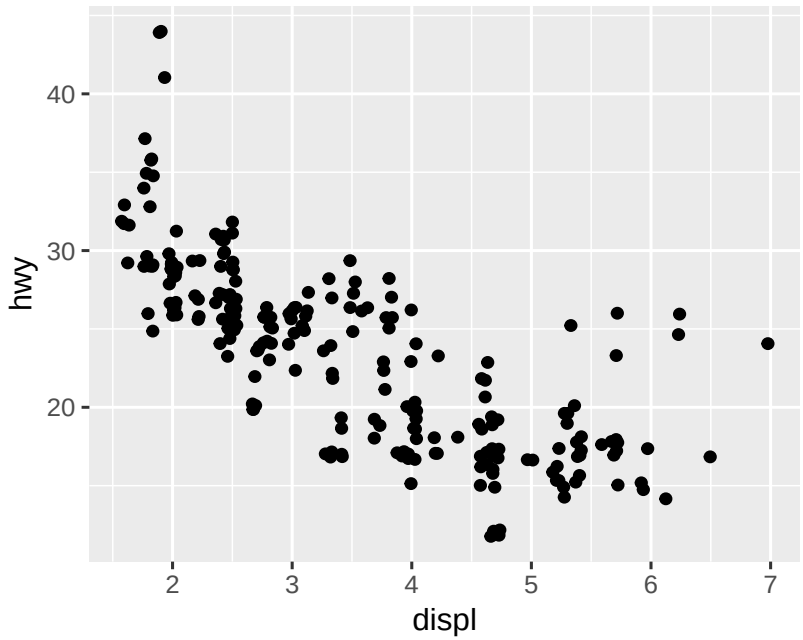
```
## [1] 234
```

```
nrow(unique(mpg[ , c("displ","hwy") ]))
```

```
## [1] 126
```

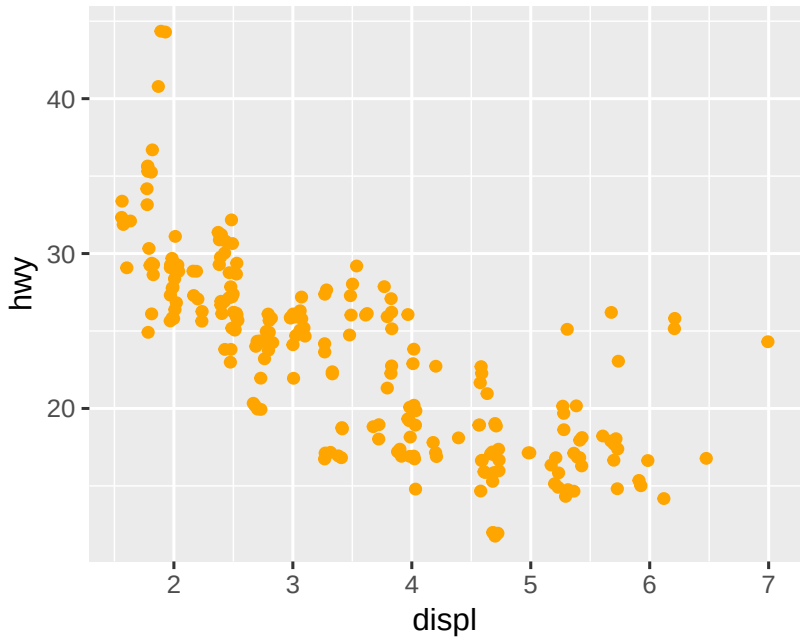
Jitter (continued)

```
ggplot(data = mpg) +
  geom_jitter(mapping = aes(x = displ, y = hwy))
```



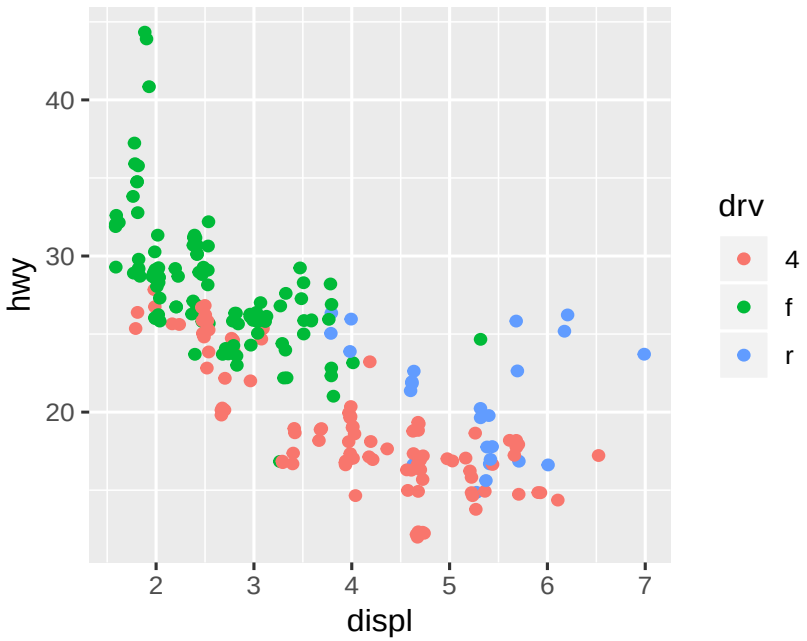
Custom color

```
ggplot(data = mpg) +
  geom_jitter(mapping = aes(x = displ, y = hwy),
              color="orange")  # Outside aes()
```



Color by type of drivetrain

```
ggplot(mpg) +
  geom_jitter(aes(x = displ, y = hwy,
                  color = drv)) # Inside aes()
```



Marker shape indicates year of manufacture

```
ggplot(mpg) +
  geom_jitter(aes(x = displ, y = hwy,
                  color = drv, shape = year))
```

```
Error: A continuous variable can not be mapped to shape
```

```
class(mpg$year)
```

```
## [1] "integer"
```

R thinks integers are continuous.

Turn year into a factor

```
table(mpg$year)
```

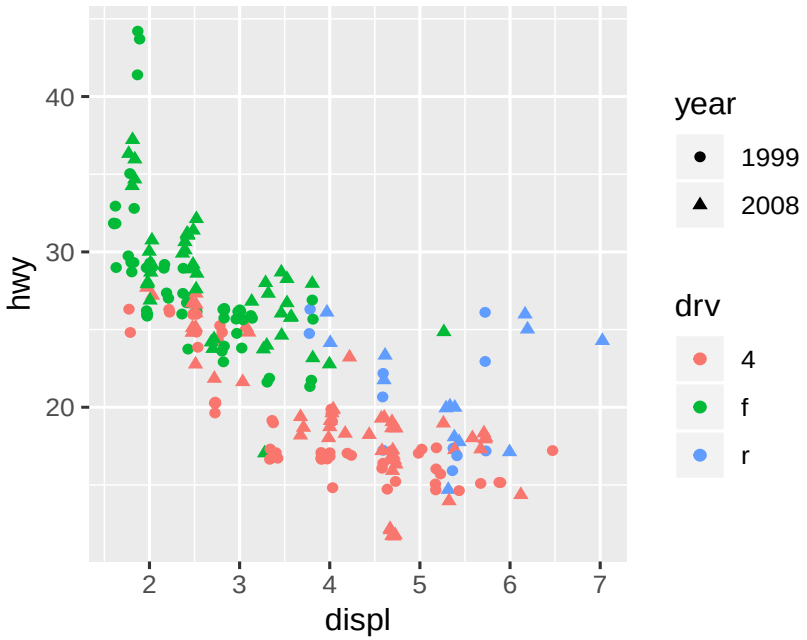
```
##
## 1999 2008
##  117  117
```

```
mpg.mod <- transform(mpg, year = as.factor(year))
class(mpg.mod$year)
```

```
## [1] "factor"
```

Marker shape indicates year of manufacture (CORRECT)

```
ggplot(mpg.mod) +
  geom_jitter(aes(x = displ, y = hwy,
                  color = drv, shape = year))
```



Multiple Layers

What if we want more than one layer?

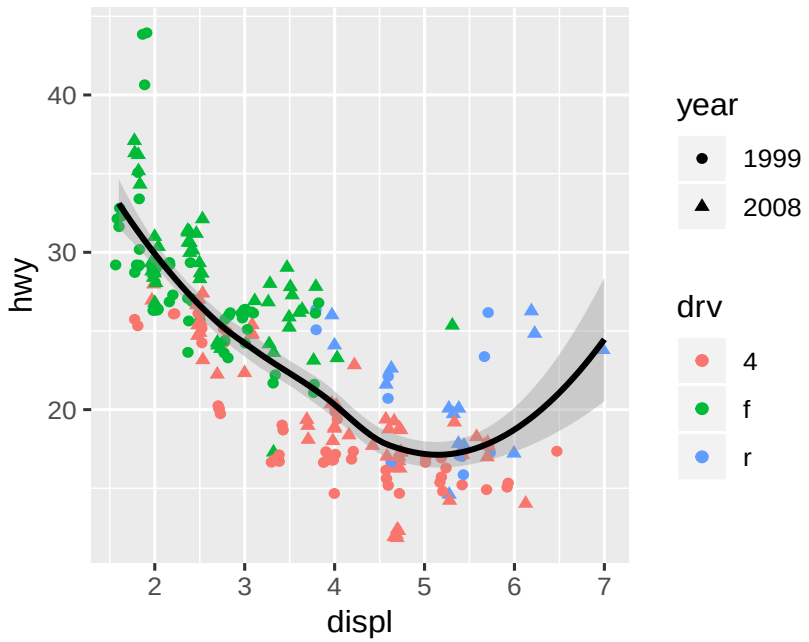
For example, add smoother to scatter plot:

```
ggplot(...) + geom_point(aes(...)) + geom_smooth(aes(...))
```

- Specifying same aesthetics in each geom call is hard to read and prone to error
- Instead:
  - Specify shared aesthetics in call to **ggplot()**
  - When creating a layer, add aesthetics specific to that layer
  - Can also override aesthetics with **inherit.aes** in geom call

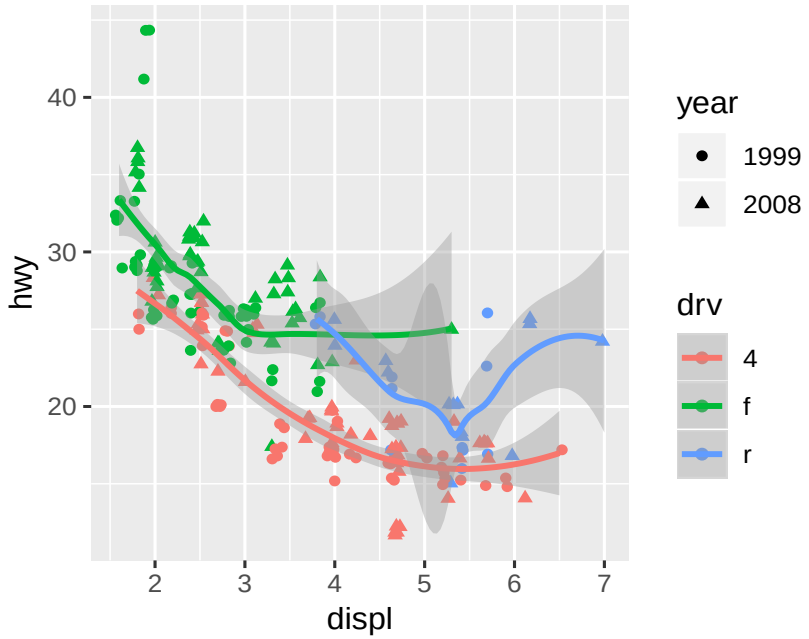
Single smoother for all data

```
ggplot(mpg.mod,
       aes(x = displ, y = hwy)) +
  geom_jitter(aes(group = drv, color = drv, shape = year)) +
  geom_smooth(method = "loess", color = "black")
```



All aesthetics shared

```
h.basic <- ggplot(mpg.mod,
                  aes(x = displ, y = hwy,
                      color = drv, group = drv,
                      shape = year)) +
           geom_jitter()
h.basic + geom_smooth(method="loess")
```



Indicate number of observations in each group

plyr package (by H. Wickham) makes it easy to compute group sizes:

```
(mpg.N <- ddply(mpg, .(drv), nrow))
```

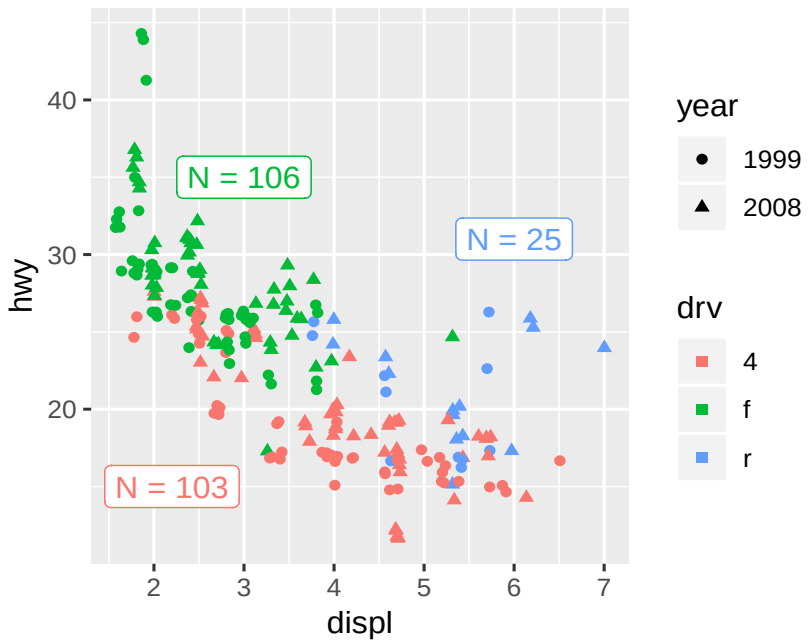
```
##   drv  V1
## 1   4 103
## 2   f 106
## 3   r  25
```

Also specify locations and prettify labels:

```
mpg.N <- cbind(mpg.N, x = c(2.2, 3, 6), y = c(15, 35, 31))
mpg.N <- transform(mpg.N, V1 = paste("N =", V1))
```

Using inherit.aes

```
h.basic +
  geom_label(data = mpg.N,
             aes(x = x, y = y, label = V1, color = drv),
             inherit.aes = F, show.legend = F) +
  guides(color = guide_legend(override.aes = list(shape = 15)))
```



Without show.legend argument, color legend shows "a" instead of shape.

Alternative approach

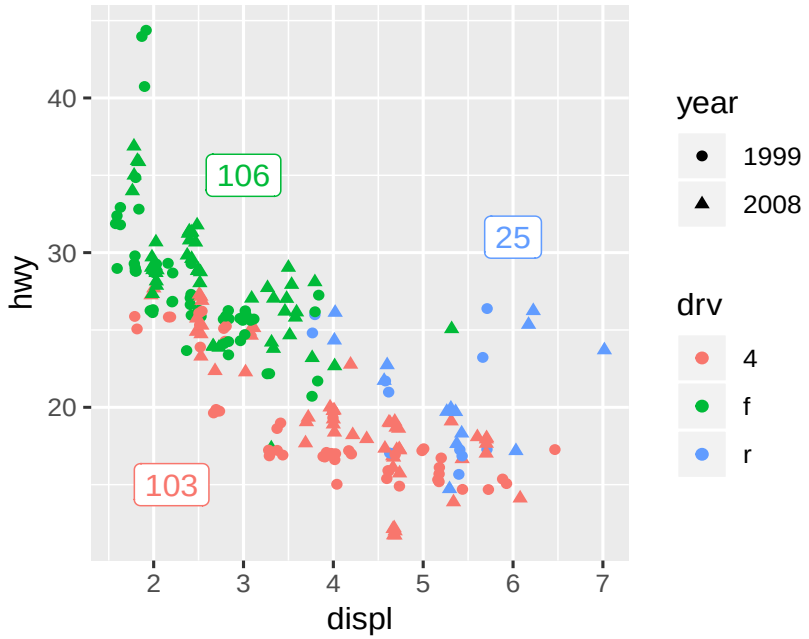
```
tmp <- cbind(ddply(mpg, .(drv), nrow),
             displ = c(2.2, 3, 6), hwy = c(15, 35, 31))
h.basic + geom_label(data = tmp, aes(label = V1))
```

```
Error in FUN(X[[i]], ...) : object 'year' not found
```

Shared aesthetic "shape = year" must be available even though geom_label doesn't understand shape aesthetic.

Alternative approach (CORRECT)

```
tmp <- cbind(ddply(mpg, .(drv), nrow),
             displ = c(2.2, 3, 6), hwy = c(15, 35, 31))
ggplot(mpg.mod,
       aes(x = displ, y = hwy, color = drv, group = drv)) +
  geom_jitter(aes(shape = year)) +
  geom_label(data = tmp, aes(label = V1), show.legend = F)
```



Some other layer types

- geom_histogram, geom_density
- geom_bar, geom_boxplot
- geom_line
- geom_abline
- geom_errorbar

To discover more:

- Cheat sheet
- Search R help for "geom_"
- Google

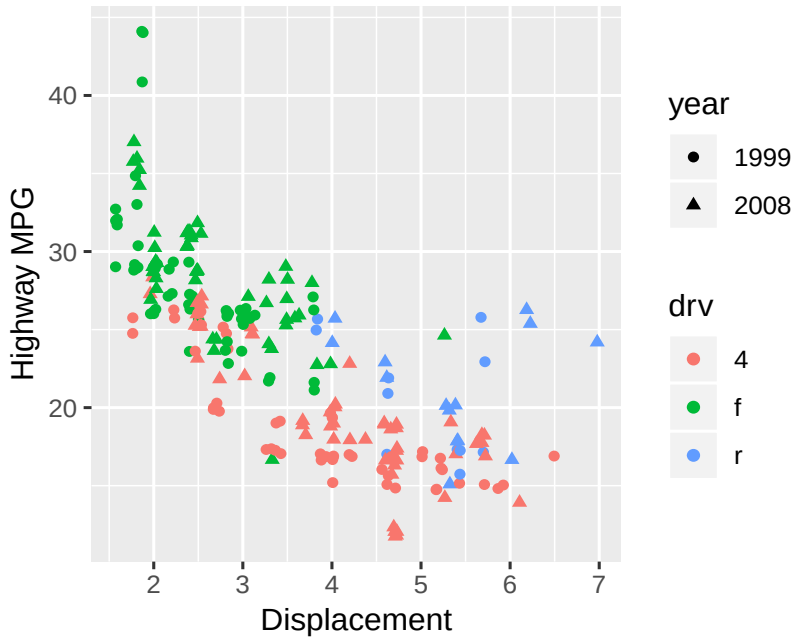
Documentation for a geom function

- Explains arguments
- Explains aesthetics; boldface $\Rightarrow$ required
- Examples
- Combines similar functions on same page (eg **geom_label** and **geom_text**)
- Available via:
    - Your favorite R help interface
    - https://ggplot2.tidyverse.org/
        - Appears in Google searches
        - Shows results of examples

Additional Customization

Change axis titles

```
(h.basic <- h.basic +
  labs(x = "Displacement", y = "Highway MPG"))
```



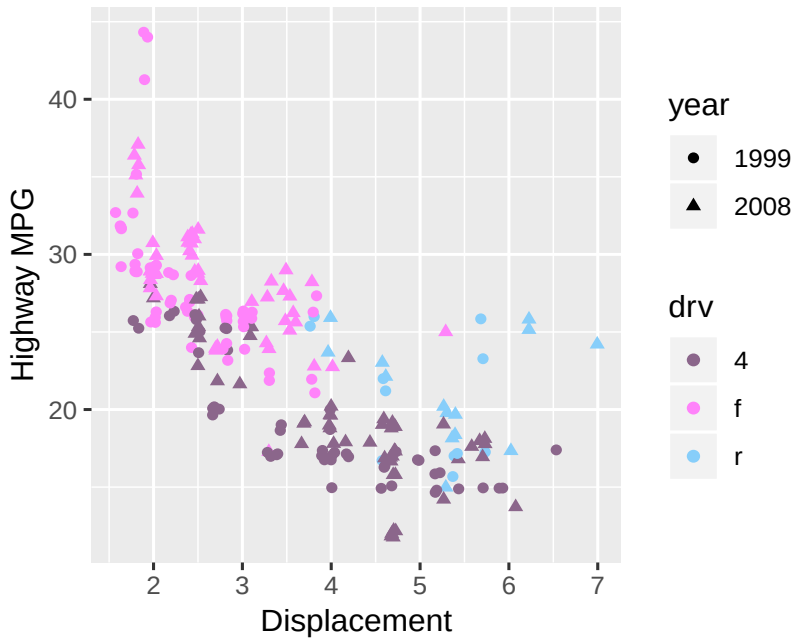
Change group colors

```
clr.drive <- c(f = "orchid1",
               r = "lightskyblue",
               "4" = "plum4")
```

- If you do not explicitly specify factor levels in vector, colors will be mapped by order.
- Can also use color hex codes (eg "#8c1515")

Change group colors (continued)

```
h.basic + scale_color_manual(values = clr.drive)
```



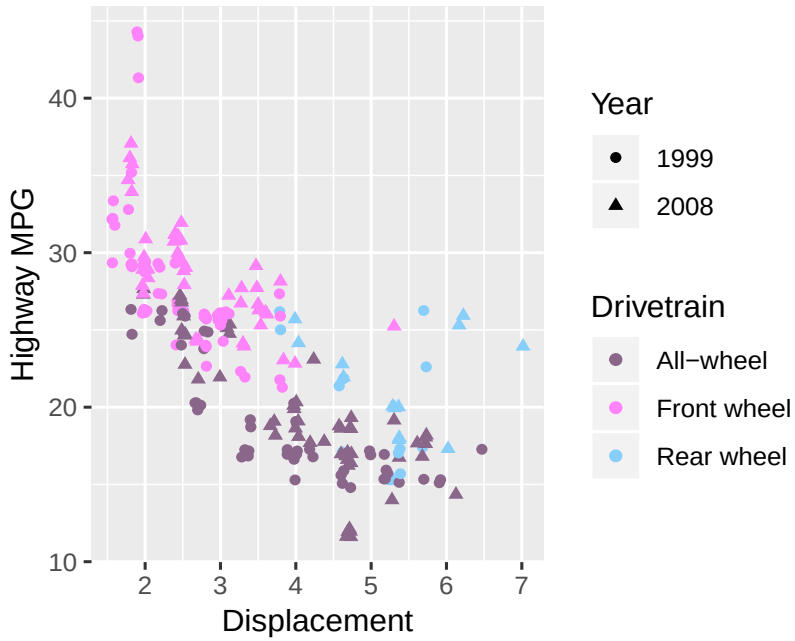
Change legend labels

```
lbl.drive <- c(f = "Front wheel",
               r = "Rear wheel",
               "4" = "All-wheel")
```

Now it's VERY IMPORTANT to get the mapping correct!

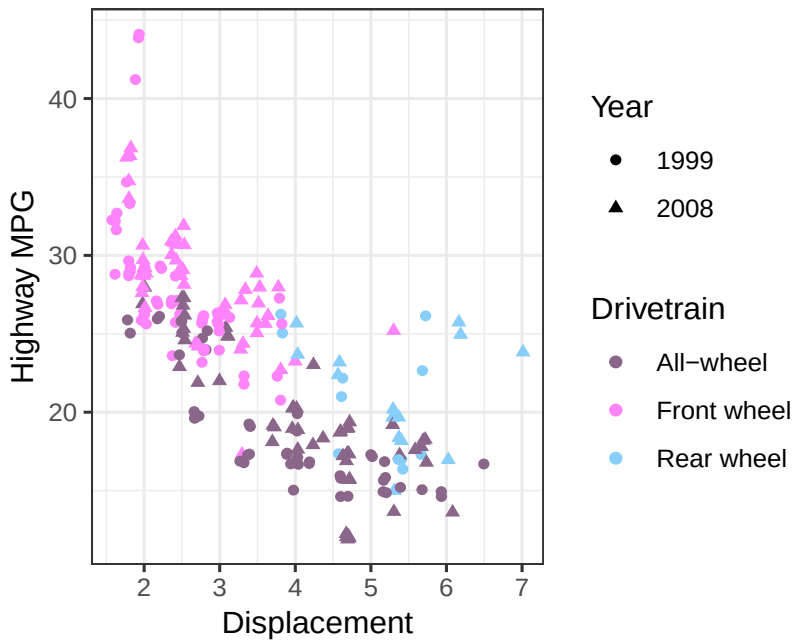
Change legend labels (continued)

```
(h.basic <- h.basic +
  labs(shape = "Year") +
  scale_color_manual(name = "Drivetrain",
                     values = clr.drive,
                     labels = lbl.drive))
```



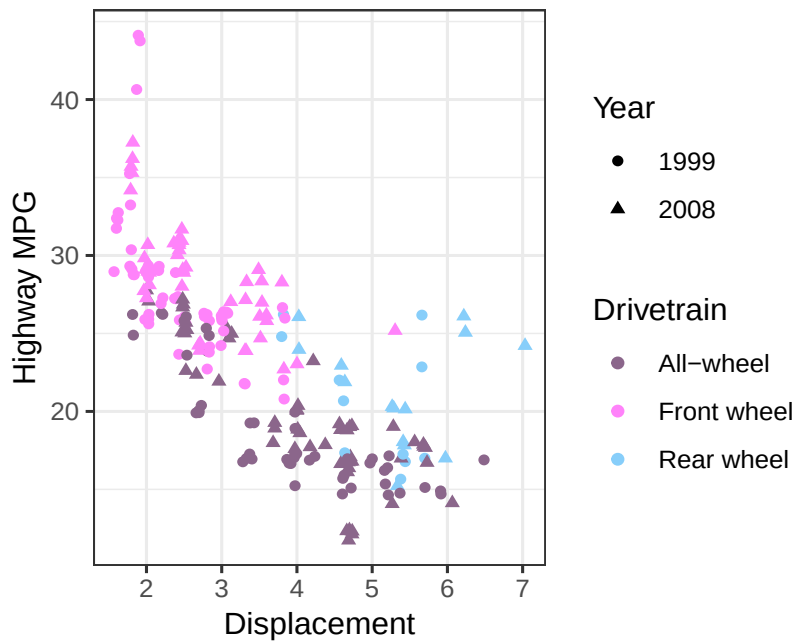
Change background

```
h.basic + theme_bw()
```



Change grid

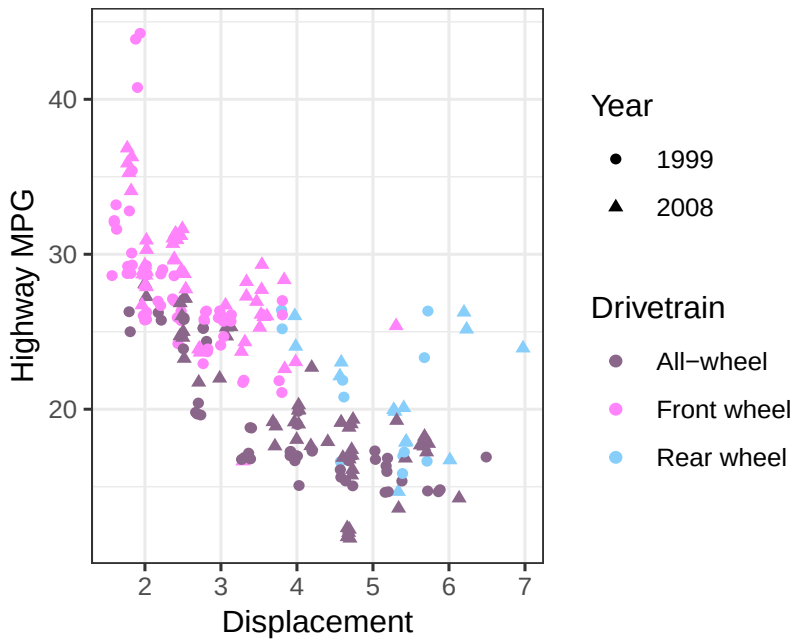
```
h.basic + theme_bw() +
  theme(panel.grid.minor.x = element_blank())
```



theme() must be called after theme_bw()

Use theme_bw() for future plots in R session

```
theme.default <- theme_set(theme_bw())
h.basic + theme(panel.grid.minor.x = element_blank())
```

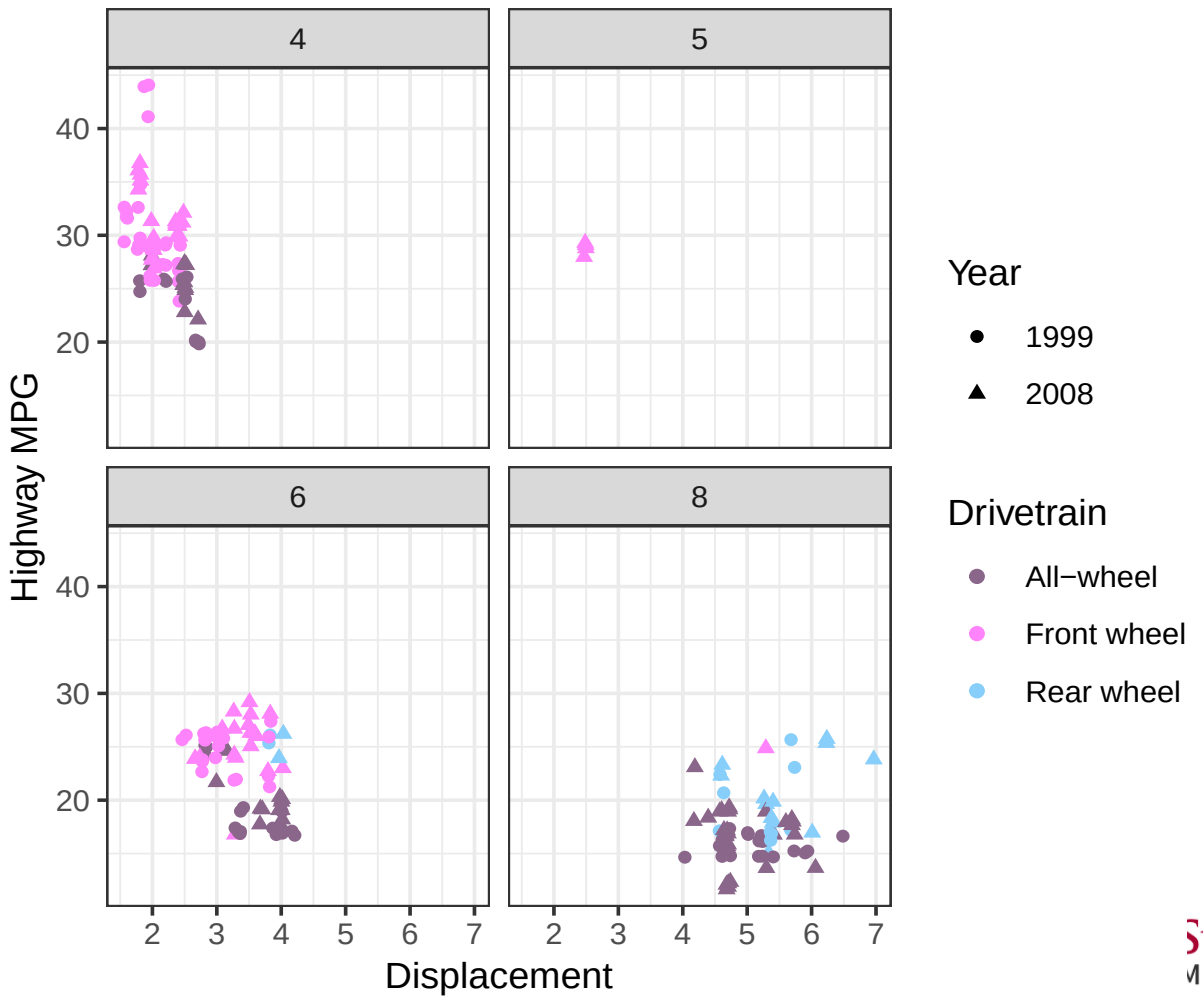


Multiple Plots in a Figure

Facets

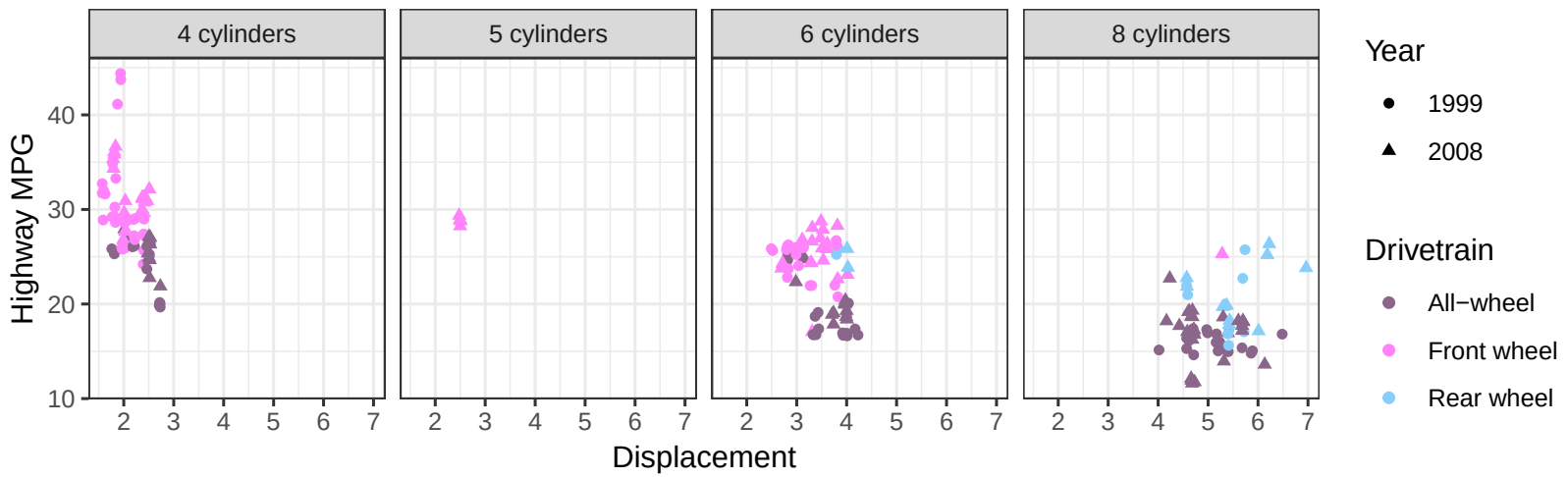
Split plot by number of cylinders

```
h.basic + facet_wrap(vars(cyl))
```



Changing facet panel strip labels

```
lbl.cyl <- paste(unique(mpg.mod$cyl), "cylinders")
names(lbl.cyl) <- unique(mpg.mod$cyl)  # REQUIRED
h.basic +
  facet_wrap(vars(cyl), nrow = 1,
             labeller = as_labeller(lbl.cyl))
```



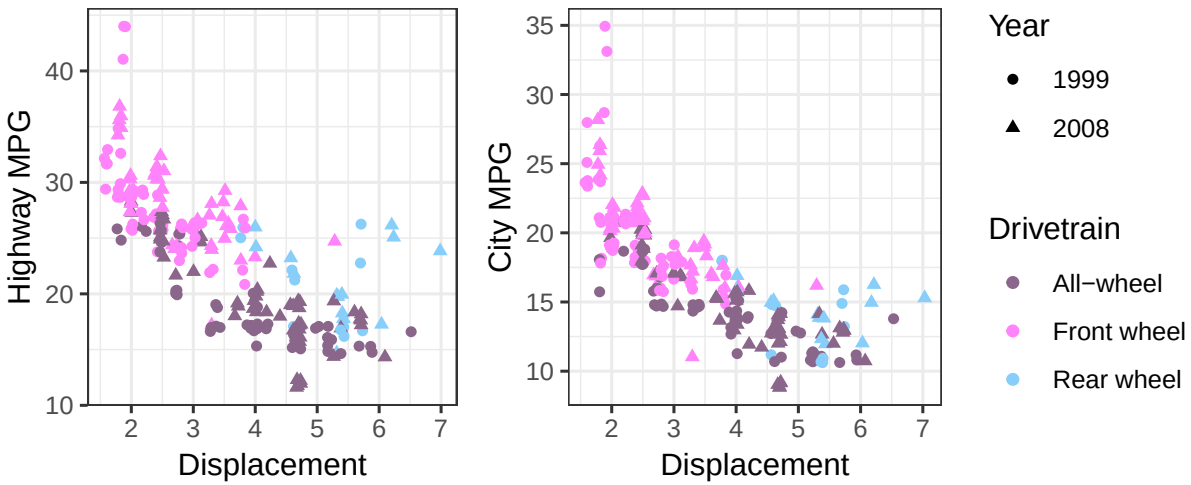
Changing facet panel strip labels (ALTERNATIVE)

Labeller can also be a function:

```
lbl.cyl <- function(string) paste(string, "cylinders")
# Same facet_wrap() call as preceding slide
```

Using grid layout

```
# HOMEWORK: Define h.city with mpg$cty on y-axis.
grid.arrange(h.basic + theme(legend.position="none"),
             h.city,
             nrow=1, widths = c(1, 1.55))
```



Using list of ggplot objects with grid.arrange

```
h <- list()
for (y in yvars) {
  htmp <- ggplot(...) + ...
  h <- append(h, list(htmp))
}
do.call(grid.arrange, append(h, list(...)))
    # grid.arrange optional args in ...
```

Revisiting facets

- Using **grid.arrange** required fiddling to get widths correct.
- Couldn't use facets because **ggplot2** uses a single column for y.
- **melt()** in **reshape2** package (by H. Wickham) makes it easy to combine M columns in data frame into single column
  - Result has M times as many rows
  - The M columns are replaced by 2:
    - Original column name
    - Value in that column
  - Do not combine factors with continuous values

Example: Using melt

```
tmp <- mpg.mod[ , c("displ", "drv", "year", "hwy", "cty") ]
tmp <- melt(tmp, measure.vars = c("hwy", "cty"))
names(tmp)
```

```
## [1] "displ"    "drv"      "year"     "variable" "value"
```

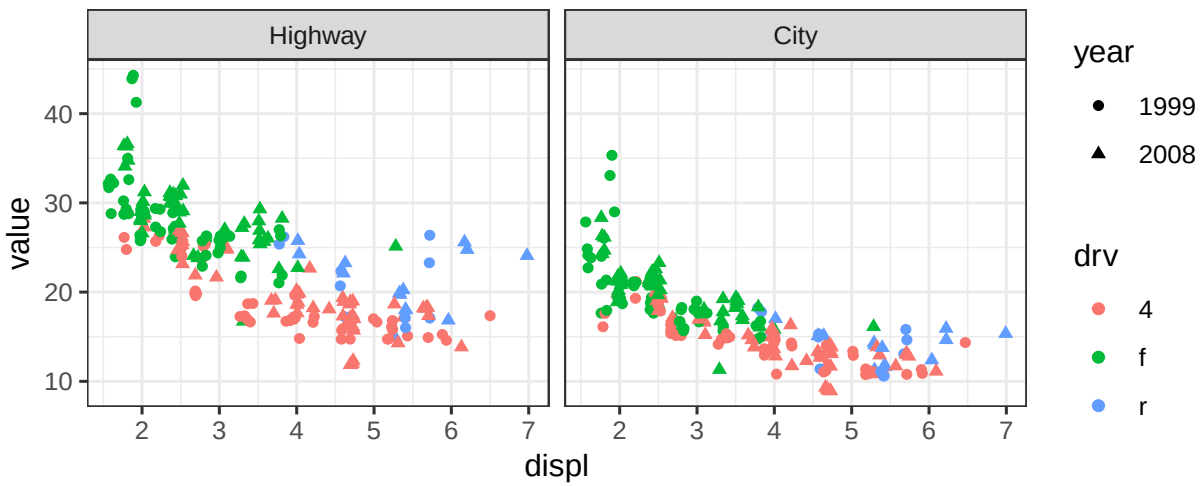
```
summary(tmp$variable)
```

```
## hwy cty
## 234 234
```

tmp$value contains values previously in columns hwy and cty

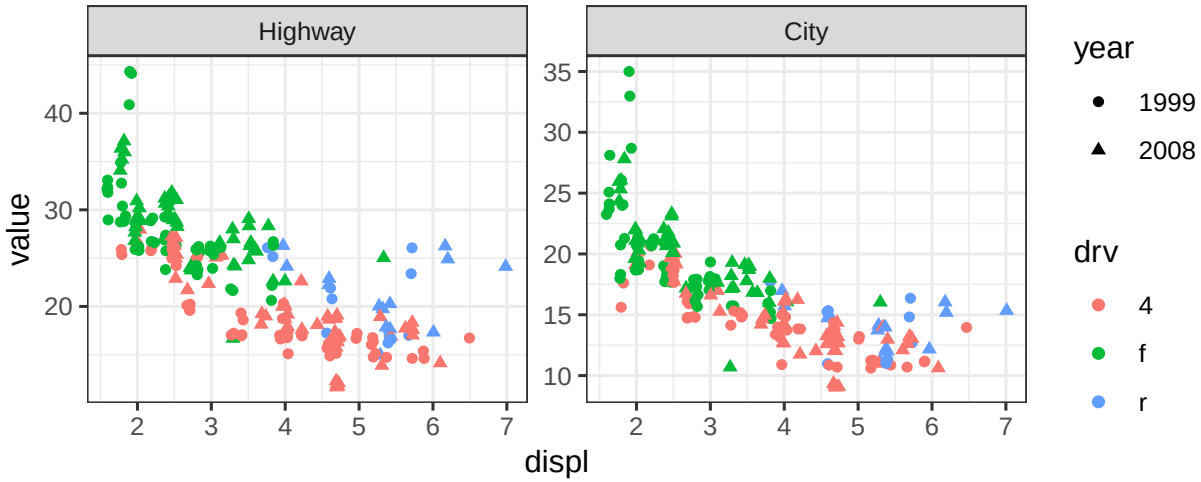
Example: Using facets after melt

```
lbls <- c(hwy="Highway", cty="City")
ggplot(tmp,
       aes(x = displ, y = value,
           color = drv, group = drv, shape = year)) +
  geom_jitter() +
  facet_wrap(vars(variable), nrow = 1,
             labeller = as_labeller(lbls))
```



Allowing different y-axis limits

```
ggplot(tmp,
       aes(x = displ, y = value,
           color = drv, group = drv, shape = year)) +
  geom_jitter() +
  facet_wrap(vars(variable), nrow = 1, scales = "free_y",
             labeller = as_labeller(lbls))
```



Carrying On

Some other ways to get started

- https://ggplot2.tidyverse.org/ has links to:
  - Cheat sheet
  - Wickham's recommended places to start
  - Tips for getting help
  - https://ggplot2.tidyverse.org/reference/index.html
- Google "ggplot2 tutorial"; some top hits:
  - http://r-statistics.co/ggplot2-Tutorial-With-R.html
  - https://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html

Becoming an expert

- Function documentation
- Google (eg "ggplot2 strip color"); best information usually at tidyverse.org or Stack Overflow
- Further explore the tidyverse, particularly:
  - **dplyr** instead of **plyr**
  - **tidyr** instead of **reshape2**
- Often multiple ways to achieve an effect