CIS 8392
Topics in Big Data Analytics

#Data I/O & Exploration

Yu-Kai Lin

Agenda

The first steps for any data science are to import and explore the data. Here we will learn how to use R to

1. Import data

2. Export data

3. Explore data

[**Acknowledgements**] The materials in the following slides are based on the source(s) below:

- R for Data Science by Garrett Grolemund and Hadley Wickham

The tidyverse

- The **tidyverse** is a collection of R packages designed for data science.
    - https://www.tidyverse.org/
- Install the tidyverse package:

```
install.packages("tidyverse")
```

Whenever there is a package that you want to use, you need to ensure that the package is installed on your machine by running `install.packages("pkg_name")`.

- Load the tidyverse into the R environment

```
library(tidyverse)
```

Whenever there is a package that you want to use, you need to load the package into the R environment by running `library(pkg_name)`. You need to run this for each new scripit or when you restart RStudio.

Core tidyverse libraries

- **tibble**: a modern re-imagining of the data frame, keeping what time has proven to be effective, and throwing out what it has not.

- **readr**: a fast and friendly way to read rectangular data (like csv).

- **tidyr**: functions that help you get to tidy data.

- **dplyr**: a grammar of data manipulation

- **ggplot2**: a system for declaratively creating graphics, based on The Grammar of Graphics.

- **purrr**: a complete and consistent set of tools for working with functions and vectors.

- **forcats**: a suite of useful tools that solve common problems with factors.

- **stringr**: a cohesive set of functions designed to make working with strings as easy as possible.

Getting data into R

Importing data into R is fairly simple. Some common data sources:

- Text files (.txt)
- Comma-separated values files (.csv)
- Excel files (.xlsx or .xls)
- Tables in relational databases (SQL server, MySQL, Oracle, ...)
- JSON files (or MongoDB)

R can handle many other rich data formats/sources as well (such as ZIP, XML, HTML, images, videos, map shapes, BigQuery, and so on). I won't enumerate all of them here. We will see some of them later in our course.

Example data files

CSV file

Semicolon delimited text file

Tab delimited text file

JSON file

Excel file

- Note 1: If your browser opens the file directly (rather than downloading the file), you can still download/save it by entering `ctrl` and `s` together.
- Note 2: Some of your data may contain missing values. Here is an example:

CSV file with missing values

Working directory

Your working directory is the folder on your computer in which you are currently working.

Important: If you are in a RStudio Project, the project directory will be your working directory. You do not need to set working directory anymore.

```
# Show your current working directory
getwd()

# List the files and folders in the current working directory
list.files()

# Set your working directory; make sure the directory exist!!
setwd("C:/CIS8392/") #NOTE: / (forward slash) instead of \ (backward slash)
```

How to find a file's (or a folder's) path

- On Windows PC:
  - Option 1: http://windowsclan.com/how-to-find-file-path-in-windows-10/
  - Option 2: https://www.pcworld.com/article/251406/windows_tips_copy_a_file_path_show_o
- On Mac:
  - Option 1: http://osxdaily.com/2015/11/05/copy-file-path-name-text-mac-os-x-finder/
  - Option 2: http://osxdaily.com/2013/06/19/copy-file-folder-path-mac-os-x/

Your turn

1. Make sure that you are in the **CIS8392** project and you know where the project dirctory is

2. Put the example data files (from the "Example data files" slide) into the following directory: `<PROJECT_DIR>/data/`

3. Run `list.files(path = "data/")` in the R console. Verify that the HousePrices files are shown.

Reading a file

- Read file through an `absolute` path:

```
df <- read_csv(file="C:/CIS8392/data/HousePrices.csv")
df
```

```
## # A tibble: 546 x 12
##    price lotsize bedrooms bathrooms stories driveway recreation fullbase gasheat
##    <dbl>   <dbl>    <dbl>     <dbl>   <dbl> <chr>    <chr>      <chr>    <chr>
##  1 42000    5850        3         1       2 yes      no         yes      no
##  2 38500    4000        2         1       1 yes      no         no       no
##  3 49500    3060        3         1       1 yes      no         no       no
##  4 60500    6650        3         1       2 yes      yes        no       no
##  5 61000    6360        2         1       1 yes      no         no       no
##  6 66000    4160        3         1       1 yes      yes        yes      no
##  7 66000    3880        3         2       2 yes      no         yes      no
##  8 69000    4160        3         1       3 yes      no         no       no
##  9 83800    4800        3         1       1 yes      yes        yes      no
## 10 88500    5500        3         2       4 yes      yes        no       no
## # ... with 536 more rows, and 3 more variables: aircon <chr>, garage <dbl>,
## #   prefer <chr>
```

- Read file through a `relative` path (relative to the working/project directory):

```
df <- read_csv(file="data/HousePrices.csv")
df
```

```
## # A tibble: 546 x 12
##    price lotsize bedrooms bathrooms stories driveway recreation fullbase gasheat
##    <dbl>   <dbl>    <dbl>     <dbl>   <dbl> <chr>    <chr>      <chr>    <chr>
##  1 42000    5850        3         1       2 yes      no         yes      no
##  2 38500    4000        2         1       1 yes      no         no       no
##  3 49500    3060        3         1       1 yes      no         no       no
##  4 60500    6650        3         1       2 yes      yes        no       no
##  5 61000    6360        2         1       1 yes      no         no       no
##  6 66000    4160        3         1       1 yes      yes        yes      no
##  7 66000    3880        3         2       2 yes      no         yes      no
##  8 69000    4160        3         1       3 yes      no         no       no
##  9 83800    4800        3         1       1 yes      yes        yes      no
## 10 88500    5500        3         2       4 yes      yes        no       no
## # ... with 536 more rows, and 3 more variables: aircon <chr>, garage <dbl>,
## #   prefer <chr>
```

- **Important**: Always make sure that you are using the right working directory (run `getwd()` and `setwd()`) before you try to access a file.

Read other text files

The key is to specify the correct column separator character.

```
df_semicolon <- read_delim(file="data/HousePrices_semicolon.txt", delim=";")
```

If there is missing values in your data, you need to tell R how to recognize these missing values.

```
df_na <- read_csv(file="data/HousePrices_with_missing_values.csv",
                  na = c("","NA", "Not Available"))
```

Read data from the Internet

```r
url <- "https://stats.idre.ucla.edu/wp-content/uploads/2016/02/test-1.csv"
df <- read_csv(file=url)
df
```

```
## # A tibble: 5 x 5
##   make  model     mpg weight price
##   <chr> <chr>   <dbl>  <dbl> <dbl>
## 1 amc   concord    22   2930  4099
## 2 amc   oacer      17   3350  4749
## 3 amc   spirit     22   2640  3799
## 4 buick century    20   3250  4816
## 5 buick electra    15   4080  7827
```

Read Excel file

To read Excel data, we need to use a library: readxl

```r
# readxl is part of the tidyverse. So you have already installed readxl.
# But readxl is not a core tidyverse package.
# So library(tidyverse) will not load it.
# You need to load the library into R before you can use it.
library("readxl")
df <- read_excel("data/HousePrices.xlsx", col_names = TRUE)
df
```

Read data from database

- The `RODBC` package provides access to Microsoft Access and Microsoft SQL Server
- The `RMySQL` package provides an interface to MySQL.
- The `ROracle` package provides an interface for Oracle.

You just need to install these libraries and follow their reference manuals.

```r
install.packages("RODBC")
install.packages("RMySQL")
install.packages("ROracle")
```

Export data to a file

Exporting data is very simple and follows similar logic as importing data:

```r
# relative path - file will be written to the working directory
file_path <- "my_test.csv"
write_csv(df, path=file_path)

# absolute path
file_path <- "C:/CIS8392/data/my_test_semicolon.txt"
write_delim(df, file_path, delim=";")
```

JSON

JSON is a very popular data format, especially in web applications when the data is typically unstructured.

- Google Maps JSON example
- YouTube JSON example
- Twitter JSON example

To read and parse JSON files, we can use the `jsonlite` package:

```r
install.packages("jsonlite") #install the package on your machine
library("jsonlite") #load the library into R before you can use it.

df = read_csv("data/HousePrices.csv")
json_content = toJSON(df)
df2=fromJSON(json_content)

write(json_content, file="data/my_HousePrices.json")
write_json(df, path = "data/my_HousePrices.json")

result_list = read_json("data/my_HousePrices.json") # a list
result_df = read_json("data/my_HousePrices.json", simplifyDataFrame=T) # a df
```

Nested JSON structure

Some JSON files have nested structure. Take this json file for example. The values for `owner`, `license`, and `organization` are themselves a JSON object.

```r
result <- fromJSON("https://api.github.com/repos/tidyverse/ggplot2",
                   simplifyDataFrame = T)
class(result) #not a df because fromJSON cannot parse the nested structure
```

```
## [1] "list"
```

To parse such JSON data into a data.frame:

```r
result <- stream_in(file("https://api.github.com/repos/tidyverse/ggplot2"))
```

```
## 
 Found 1 records...
 Imported 1 records. Simplifying...
```

```r
class(result) #a data frame
```

```
## [1] "data.frame"
```

Nested data.frame

```r
str(result) #notice that the data type of the owner column is data.frame
```

```
## 'data.frame':    1 obs. of  77 variables:
##  $ id                 : int 19438
##  $ node_id            : chr "MDEwOlJlcG9zaXRvcnkxOTQzOA=="
##  $ name               : chr "ggplot2"
##  $ full_name          : chr "tidyverse/ggplot2"
##  $ private            : logi FALSE
##  $ owner              :'data.frame':    1 obs. of  18 variables:
##   ..$ login              : chr "tidyverse"
##   ..$ id                 : int 22032646
##   ..$ node_id            : chr "MDEyOk9yZ2FuaXphdGlvbjIyMDMyNjQ2"
##   ..$ avatar_url         : chr "https://avatars.githubusercontent.com/u/22032646?
##   ..$ gravatar_id        : chr ""
##   ..$ url                : chr "https://api.github.com/users/tidyverse"
##   ..$ html_url           : chr "https://github.com/tidyverse"
##   ..$ followers_url      : chr "https://api.github.com/users/tidyverse/followers"
##   ..$ following_url      : chr "https://api.github.com/users/tidyverse/following{
##   ..$ gists_url          : chr "https://api.github.com/users/tidyverse/gists{/gis
##   ..$ starred_url        : chr "https://api.github.com/users/tidyverse/starred{/o
##   ..$ subscriptions_url  : chr "https://api.github.com/users/tidyverse/subscripti
##   ..$ organizations_url  : chr "https://api.github.com/users/tidyverse/orgs"
##   ..$ repos_url          : chr "https://api.github.com/users/tidyverse/repos"
##   ..$ events_url         : chr "https://api.github.com/users/tidyverse/events{/pr
##   ..$ received_events_url: chr "https://api.github.com/users/tidyverse/received_e
```

Flatten nested data.frame

```
result_flat <- flatten(result)
str(result_flat) # notice the number of variables has increased from 77 to 115
```

```
## 'data.frame':    1 obs. of  115 variables:
##  $ id                               : int 19438
##  $ node_id                          : chr "MDEwOlJlcG9zaXRvcnkxOTQzOA=="
##  $ name                             : chr "ggplot2"
##  $ full_name                        : chr "tidyverse/ggplot2"
##  $ private                          : logi FALSE
##  $ html_url                         : chr "https://github.com/tidyverse/ggplot2"
##  $ description                      : chr "An implementation of the Grammar of Gra
##  $ fork                             : logi FALSE
##  $ url                              : chr "https://api.github.com/repos/tidyverse/
##  $ forks_url                        : chr "https://api.github.com/repos/tidyverse/
##  $ keys_url                         : chr "https://api.github.com/repos/tidyverse/
##  $ collaborators_url                : chr "https://api.github.com/repos/tidyverse/
##  $ teams_url                        : chr "https://api.github.com/repos/tidyverse/
##  $ hooks_url                        : chr "https://api.github.com/repos/tidyverse/
##  $ issue_events_url                 : chr "https://api.github.com/repos/tidyverse/
##  $ events_url                       : chr "https://api.github.com/repos/tidyverse/
##  $ assignees_url                    : chr "https://api.github.com/repos/tidyverse/
##  $ branches_url                     : chr "https://api.github.com/repos/tidyverse/
##  $ tags_url                         : chr "https://api.github.com/repos/tidyverse/
##  $ blobs_url                        : chr "https://api.github.com/repos/tidyverse/
##  $ git_tags_url                     : chr "https://api.github.com/repos/tidyverse/
```

Data Exploration

- Data summary
- Data visualization

Data summary: `str()`

```
df <- read_csv(file="data/HousePrices.csv")
str(df)
```

```
## spec_tbl_df [546 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ price     : num [1:546] 42000 38500 49500 60500 61000 66000 66000 69000 83800
##  $ lotsize   : num [1:546] 5850 4000 3060 6650 6360 4160 3880 4160 4800 5500 ...
##  $ bedrooms  : num [1:546] 3 2 3 3 2 3 3 3 3 3 ...
##  $ bathrooms : num [1:546] 1 1 1 1 1 1 2 1 1 2 ...
##  $ stories   : num [1:546] 2 1 1 2 1 1 2 3 1 4 ...
##  $ driveway  : chr [1:546] "yes" "yes" "yes" "yes" ...
##  $ recreation: chr [1:546] "no" "no" "no" "yes" ...
##  $ fullbase  : chr [1:546] "yes" "no" "no" "no" ...
##  $ gasheat   : chr [1:546] "no" "no" "no" "no" ...
##  $ aircon    : chr [1:546] "no" "no" "no" "no" ...
##  $ garage    : num [1:546] 1 0 0 0 0 0 2 0 0 1 ...
##  $ prefer    : chr [1:546] "no" "no" "no" "no" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   price = col_double(),
##   ..   lotsize = col_double(),
##   ..   bedrooms = col_double(),
##   ..   bathrooms = col_double(),
##   ..   stories = col_double(),
##   ..   driveway = col_character(),
##   ..   recreation = col_character(),
```

Data summary: `summary()`

```
summary(df)
```

```
##      price           lotsize         bedrooms       bathrooms
##  Min.   : 25000   Min.   : 1650   Min.   :1.000   Min.   :1.000
##  1st Qu.: 49125   1st Qu.: 3600   1st Qu.:2.000   1st Qu.:1.000
##  Median : 62000   Median : 4600   Median :3.000   Median :1.000
##  Mean   : 68122   Mean   : 5150   Mean   :2.965   Mean   :1.286
##  3rd Qu.: 82000   3rd Qu.: 6360   3rd Qu.:3.000   3rd Qu.:2.000
##  Max.   :190000   Max.   :16200   Max.   :6.000   Max.   :4.000
##     stories        driveway          recreation          fullbase
##  Min.   :1.000   Length:546         Length:546         Length:546
##  1st Qu.:1.000   Class :character   Class :character   Class :character
##  Median :2.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :1.808
##  3rd Qu.:2.000
##  Max.   :4.000
##    gasheat             aircon              garage          prefer
##  Length:546         Length:546         Min.   :0.0000   Length:546
##  Class :character   Class :character   1st Qu.:0.0000   Class :character
##  Mode  :character   Mode  :character   Median :0.0000   Mode  :character
##                                         Mean   :0.6923
##                                         3rd Qu.:1.0000
##                                         Max.   :3.0000
```

Data summary: `glimpse()`

```
glimpse(df)
```

```
## Rows: 546
## Columns: 12
## $ price      <dbl> 42000, 38500, 49500, 60500, 61000, 66000, 66000, 69000, 838~
## $ lotsize    <dbl> 5850, 4000, 3060, 6650, 6360, 4160, 3880, 4160, 4800, 5500,~
## $ bedrooms   <dbl> 3, 2, 3, 3, 2, 3, 3, 3, 3, 3, 3, 2, 3, 3, 2, 2, 3, 4, 1, 2,~
## $ bathrooms  <dbl> 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2,~
## $ stories    <dbl> 2, 1, 1, 2, 1, 1, 2, 3, 1, 4, 1, 1, 2, 1, 1, 1, 2, 3, 1, 1,~
## $ driveway   <chr> "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "ye~
## $ recreation <chr> "no", "no", "no", "yes", "no", "yes", "no", "no", "yes", "y~
## $ fullbase   <chr> "yes", "no", "no", "no", "no", "yes", "yes", "no", "yes", "~
## $ gasheat    <chr> "no", "no", "no", "no", "no", "no", "no", "no", "no", "no",~
## $ aircon     <chr> "no", "no", "no", "no", "no", "yes", "no", "no", "no", "yes~
## $ garage     <dbl> 1, 0, 0, 0, 0, 0, 2, 0, 0, 1, 3, 0, 0, 0, 0, 0, 1, 0, 0, 1,~
## $ prefer     <chr> "no", "no", "no", "no", "no", "no", "no", "no", "no", "no",~
```

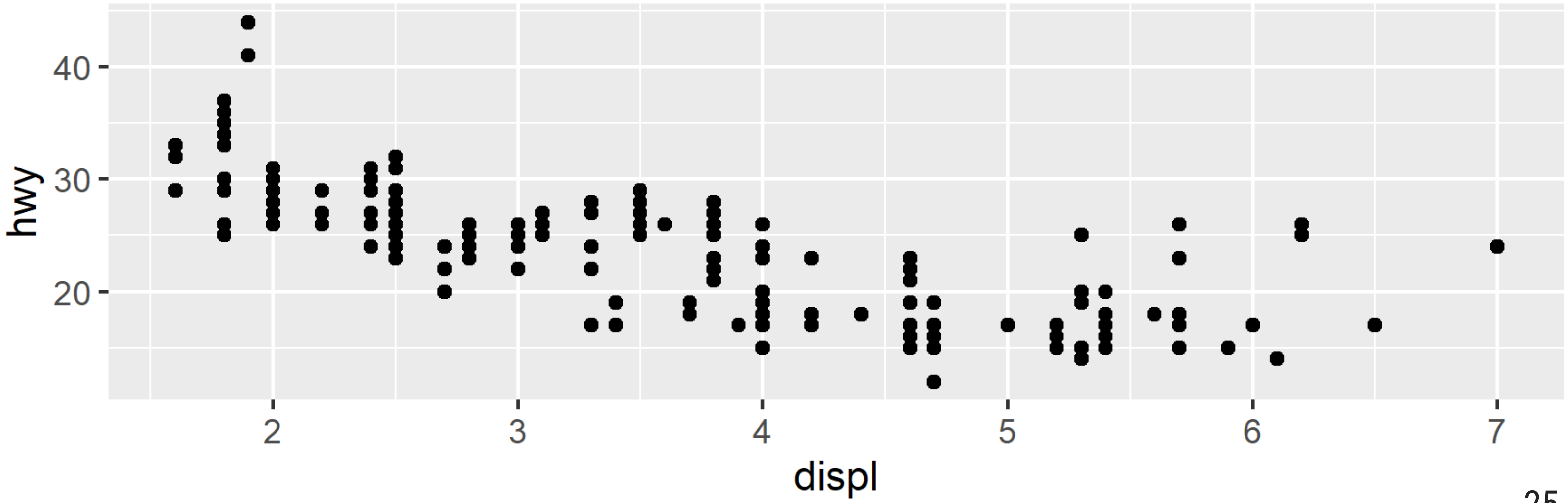
Data visualization with ggplot2

Template of ggplot2 usage:

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>)) # geom: geometrical object
```

For example:

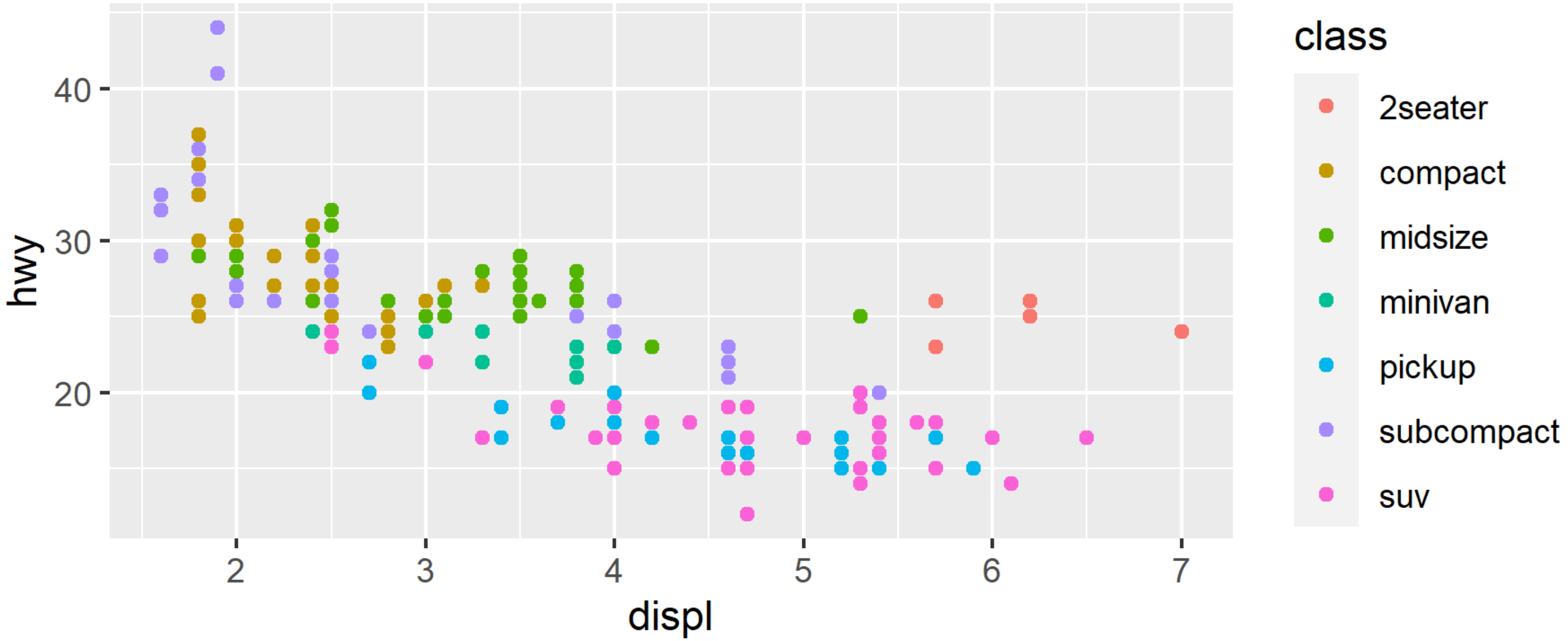
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



Customizing the aesthetic

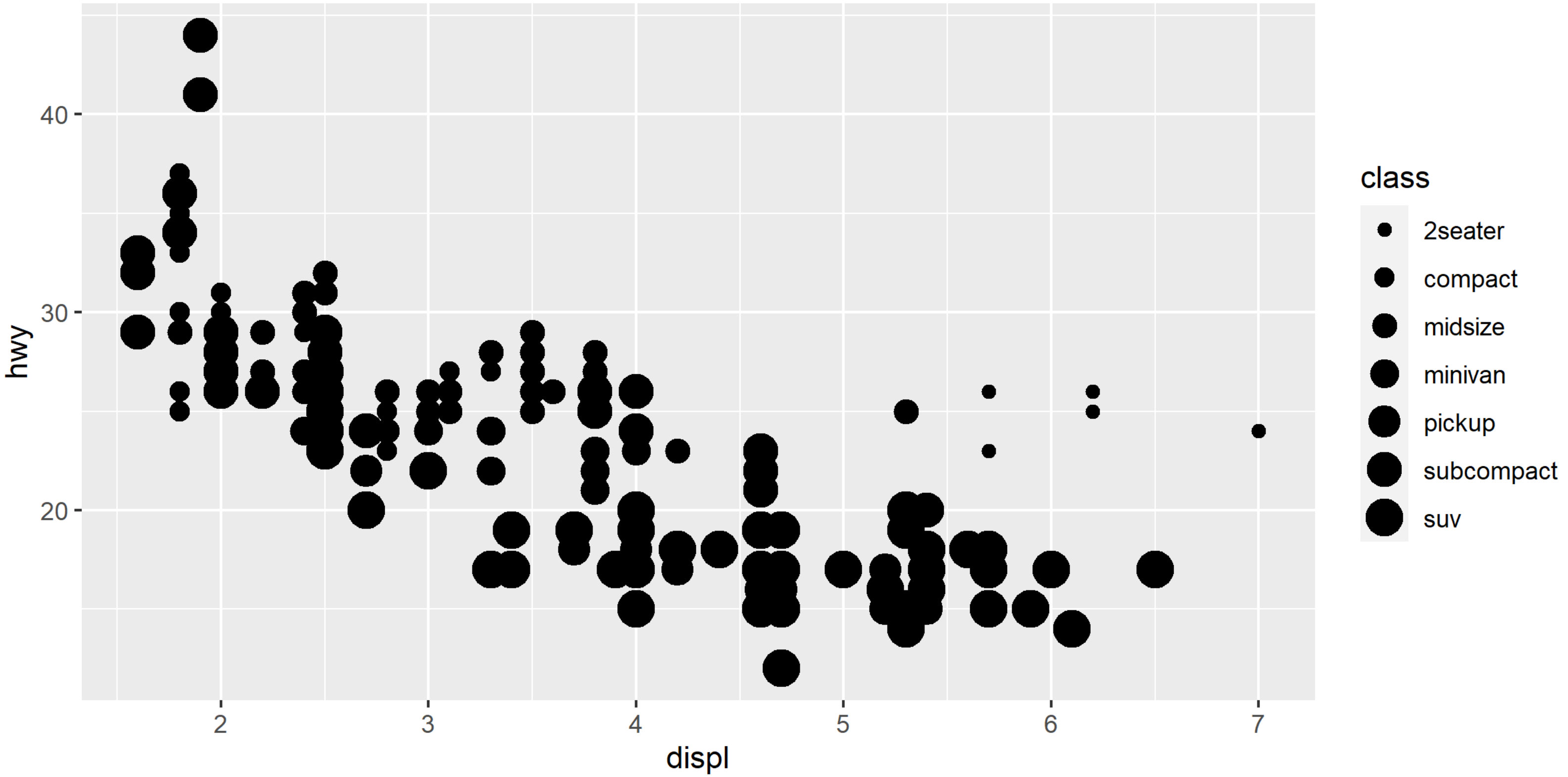
For each aesthetic, you use aes() to associate the name of the aesthetic with a variable to display.

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



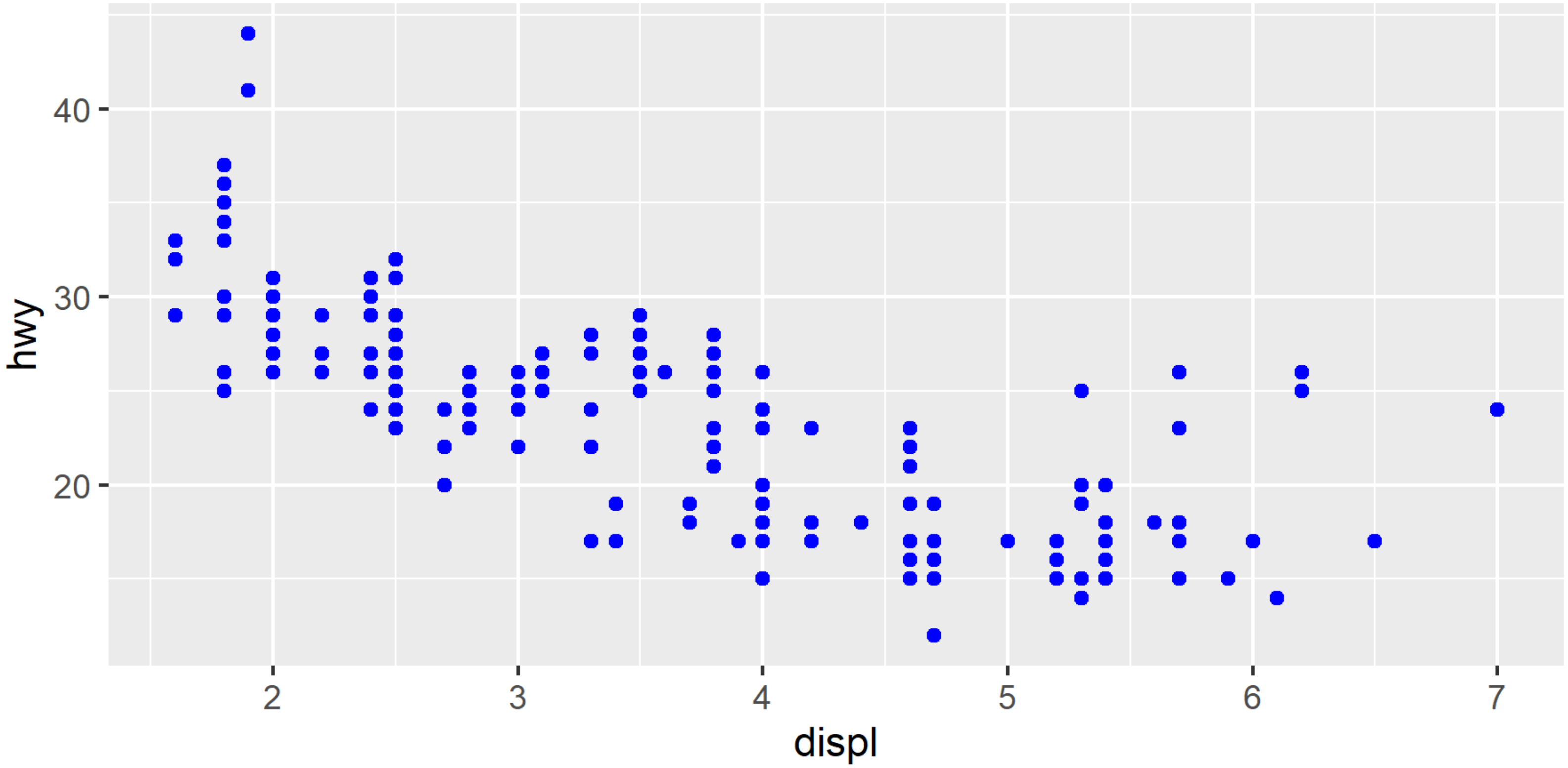
Not just color, but also size, alpha, shape, etc.

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



You can also set the aesthetic properties of your geom manually.

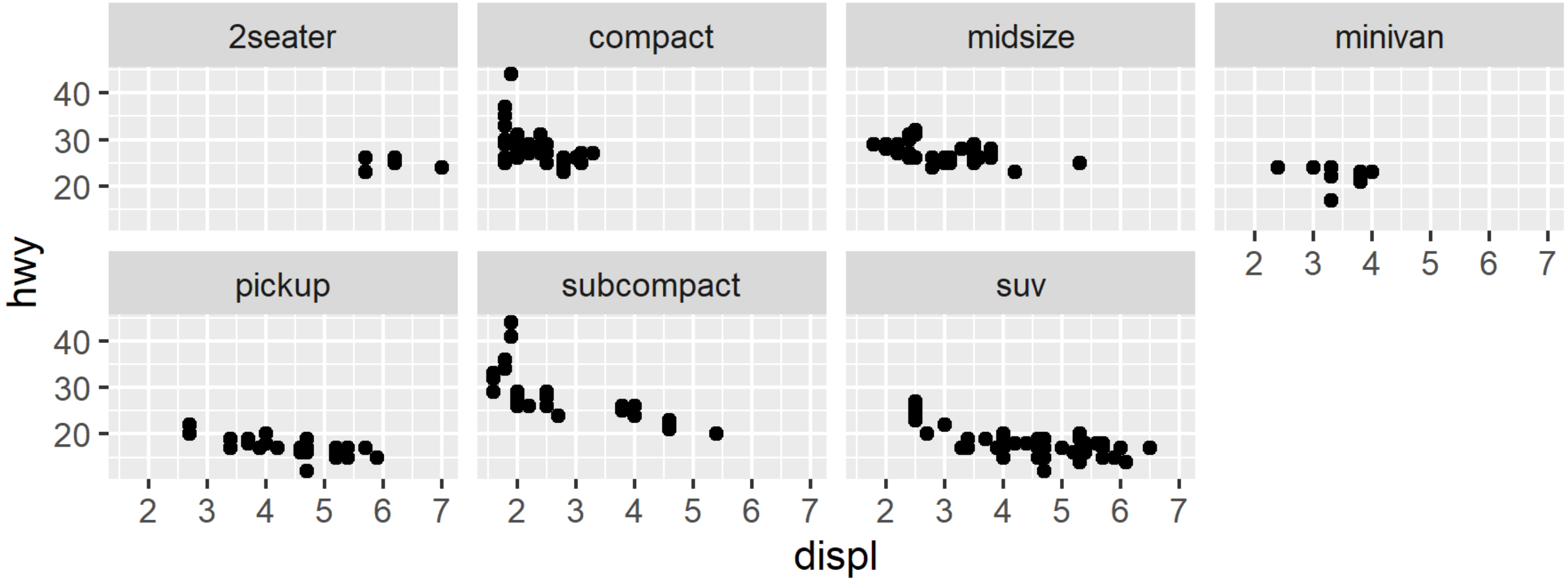
```
# notice that color is outside aes()
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



Facets

One way to add additional variables is with aesthetics (e.g., color by class). Another way, particularly useful for categorical variables, is to split your plot into facets, subplots that each display one subset of the data.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)  # y axis ~ x axis
```

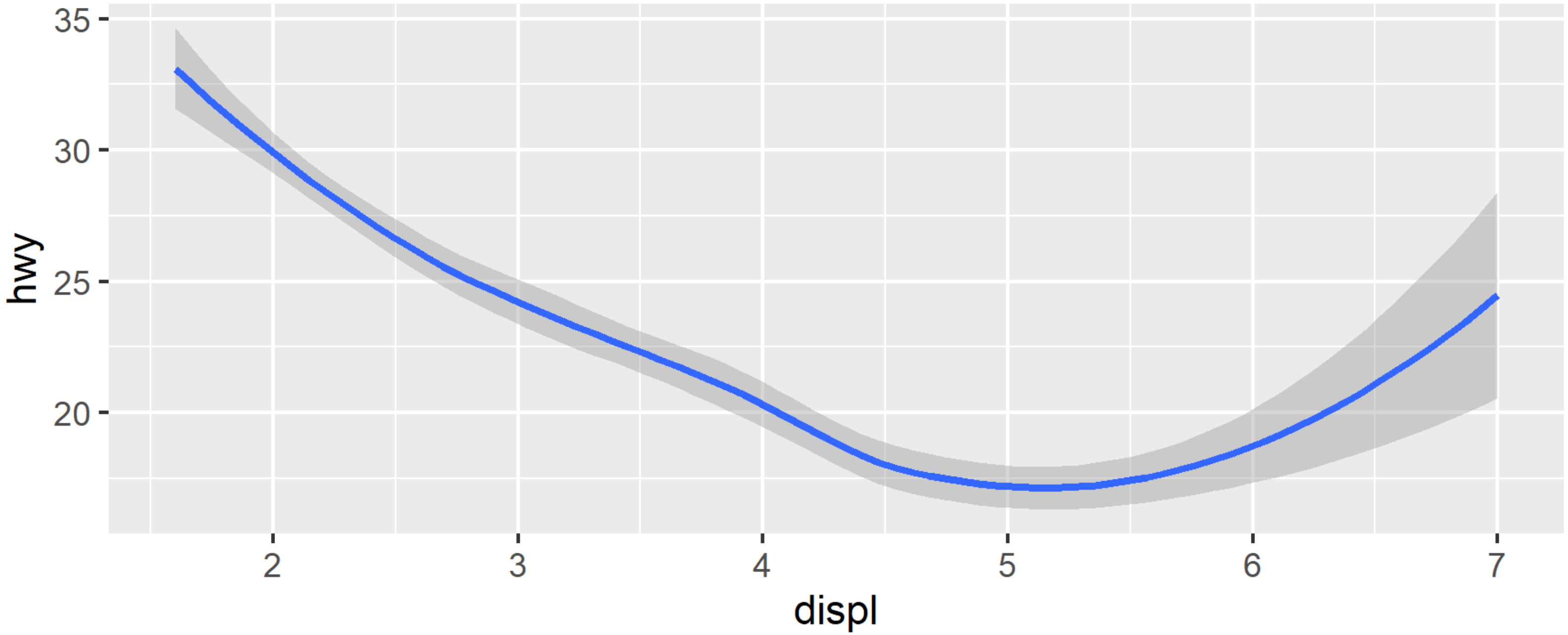


```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)  # y axis ~ x axis
```

geoms in ggplot2

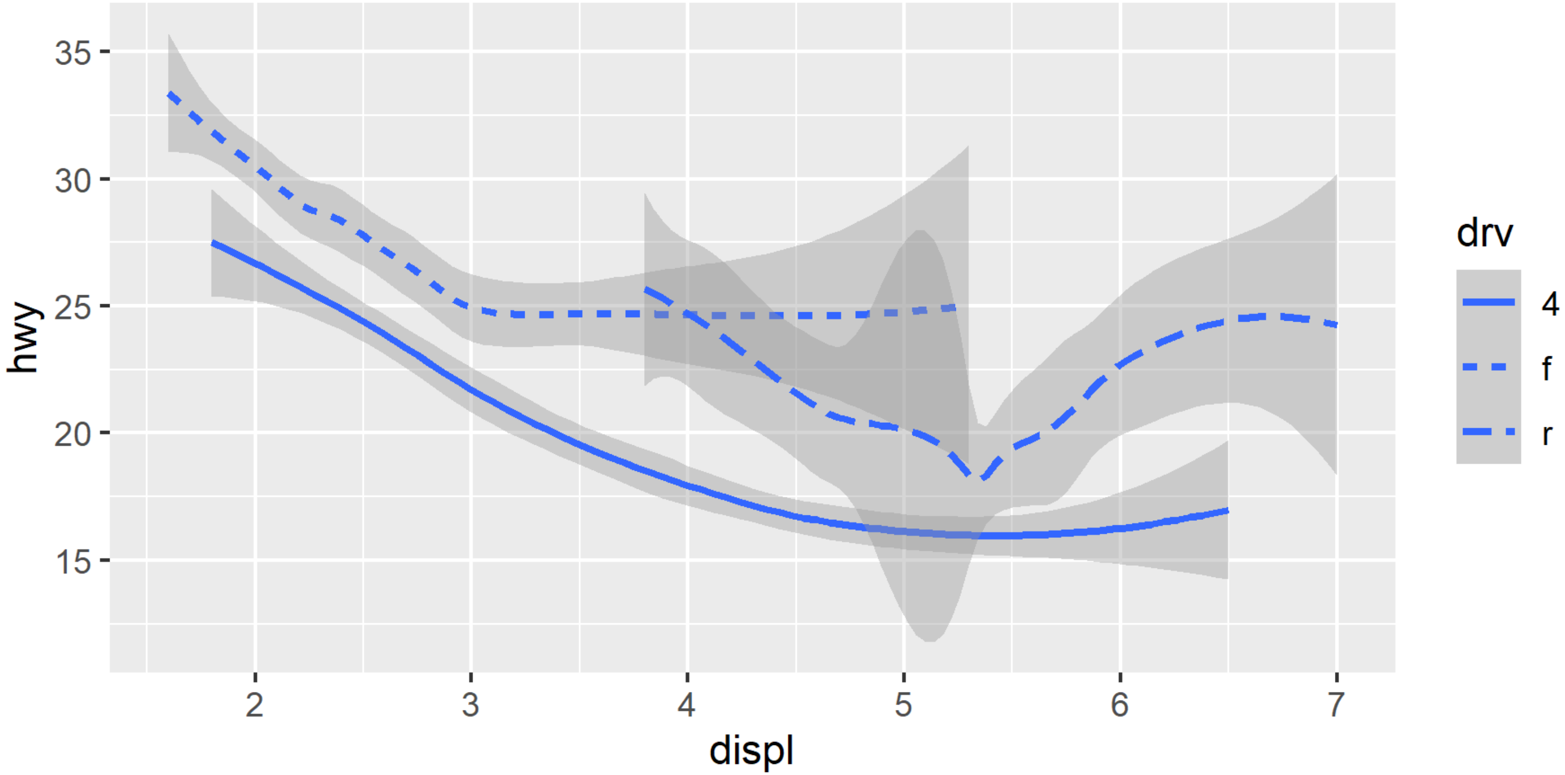
A **geom** is the geometrical object that a plot uses to represent data. People often describe plots by the type of geom that the plot uses. For example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and so on.

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



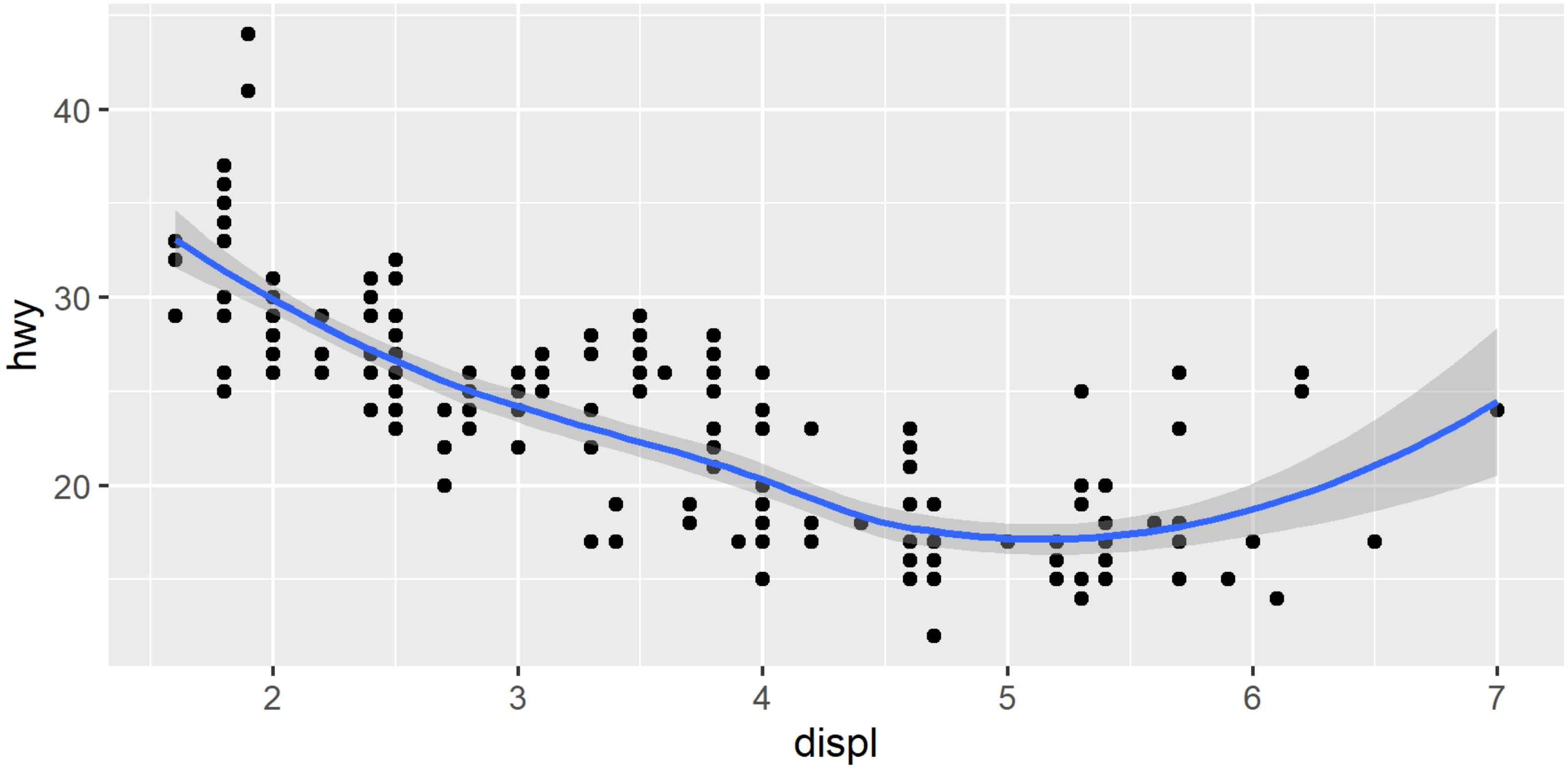
Every geom function in ggplot2 takes a `mapping` argument. However, not every aesthetic works with every geom. You could set the shape of a point, but you couldn't set the "shape" of a line. On the other hand, you could set the linetype of a line.

```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```



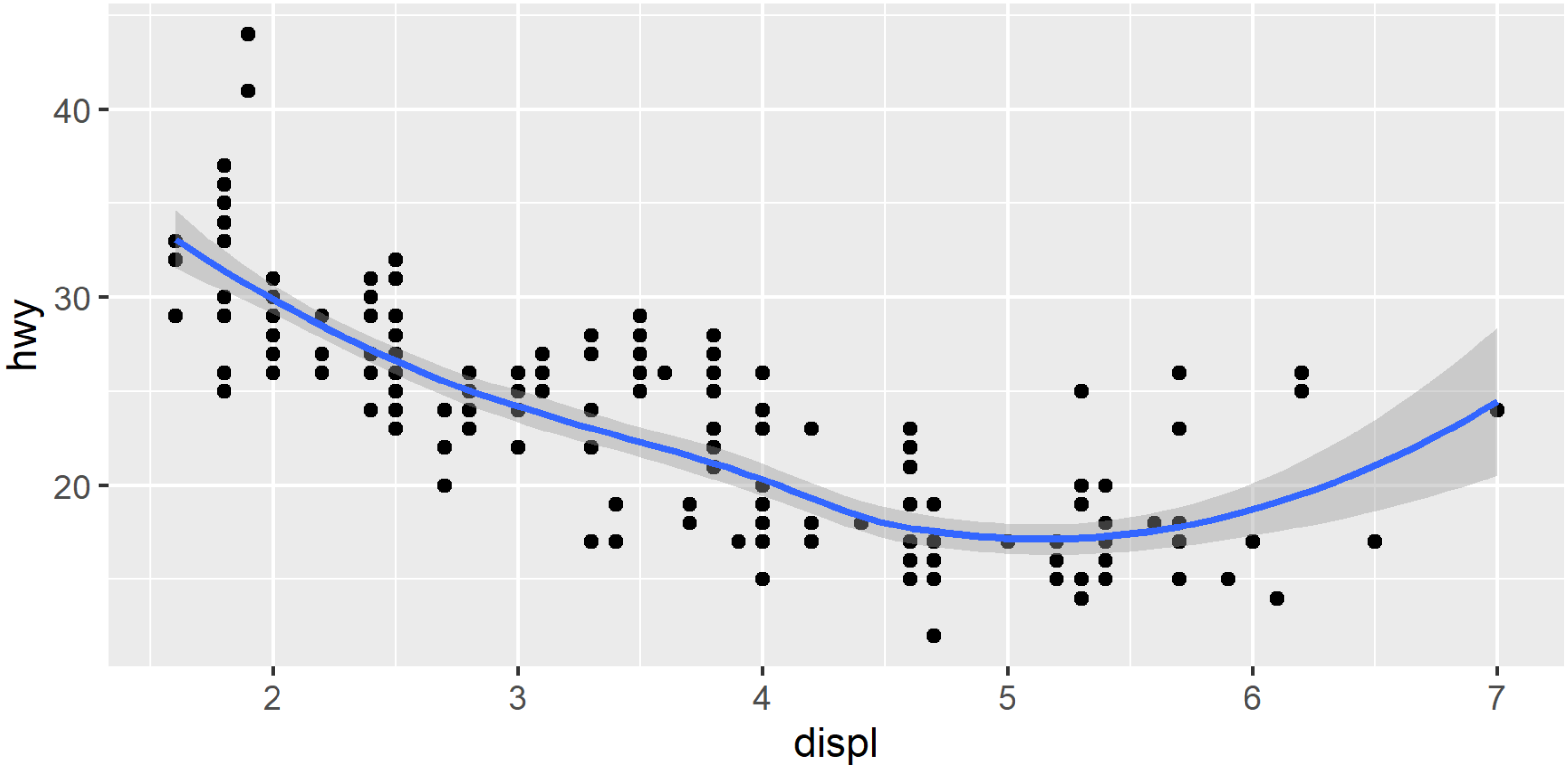
To display multiple geoms in the same plot, add multiple geom functions to `ggplot()`:

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



This, however, introduces some duplication in our code. You can avoid this type of repetition by passing a set of mappings to `ggplot()`.

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



ColorBrewer

`RColorBrewer` is an R package that allows users to create colorful graphs with pre-made color palettes that visualize data in a clear and distinguishable manner. There are 3 categories of palettes (click to see palette names):

- **Qualitative palettes** employ different hues to create visual differences between classes. These palettes are suggested for nominal or categorical data sets.

- **Sequential palettes** progress from light to dark. When used with interval data, light colors represent low data values and dark colors represent high data values.

- **Diverging palettes** are composed of darker colors of contrasting hues on the high and low extremes and lighter colors in the middle.

Template:

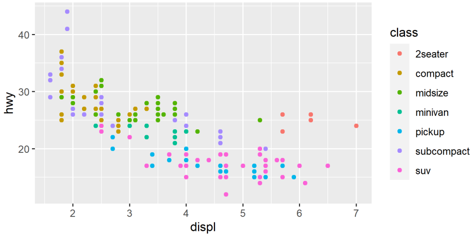
```r
install.packages("RColorBrewer")
library(RColorBrewer)
your_plot + scale_color_brewer(palette = "Palette_Name") #palette for color
your_plot + scale_fill_brewer(palette = "Palette_Name")   #palette for fill
```

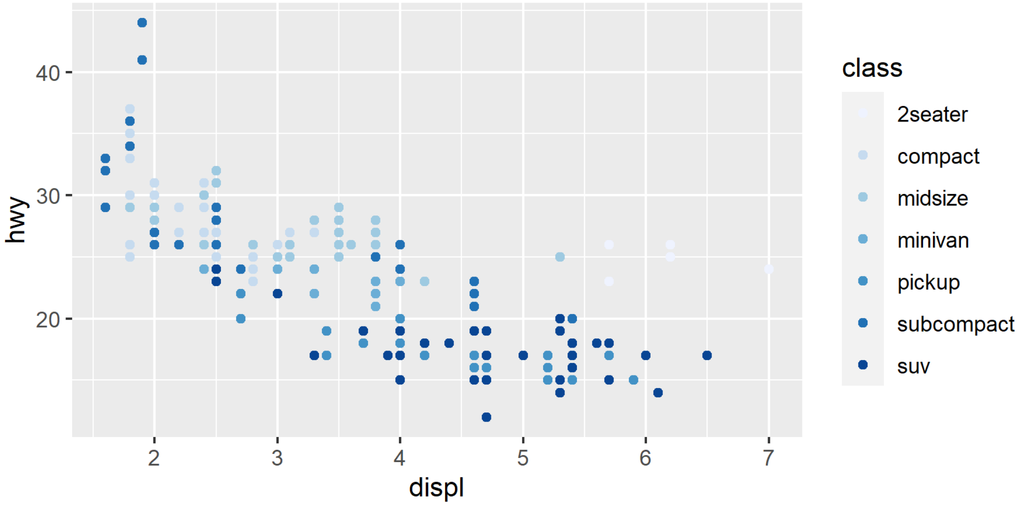
Example:

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

```
scatter #default color
```

```
scatter + scale_color_brewer(palette = "Blues")
```

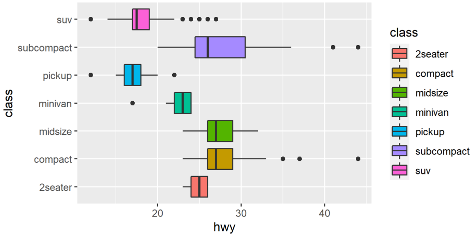




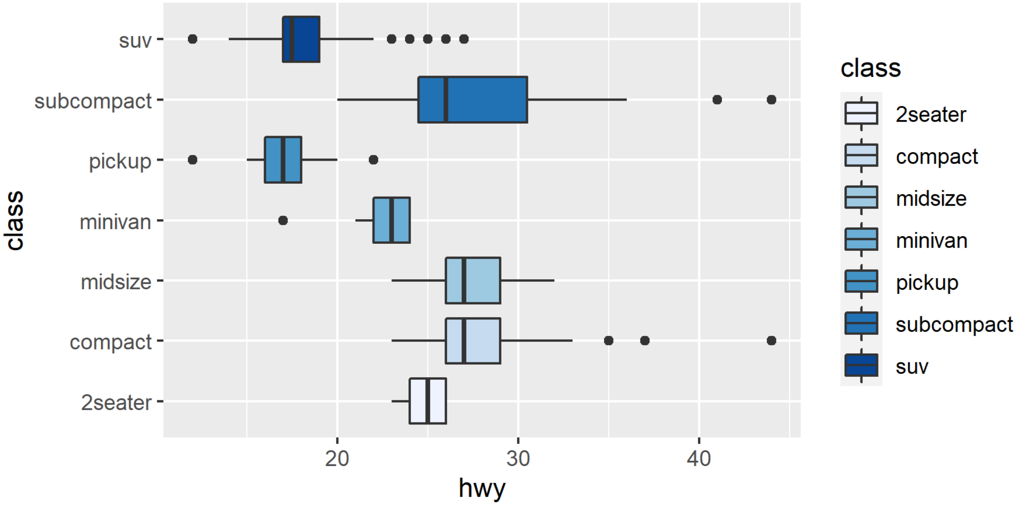
Example:

```
bar = ggplot(data = mpg, mapping = aes(x = class, y = hwy, fill=class)) +
  geom_boxplot() + coord_flip()
```

```
bar # default color
```



```
bar + scale_fill_brewer(palette = "Blues")
```



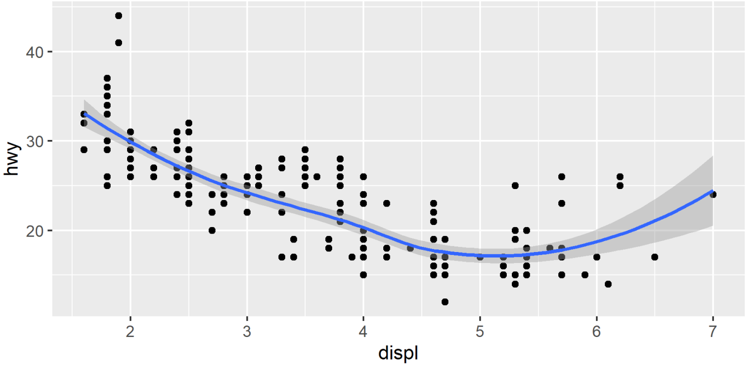
Themes

The ggthemes package gives you several beautiful themes for your ggplot.

```
install.packages("ggthemes")  # install the package on your machine first
library(ggthemes) # load it into R before you can use it
```

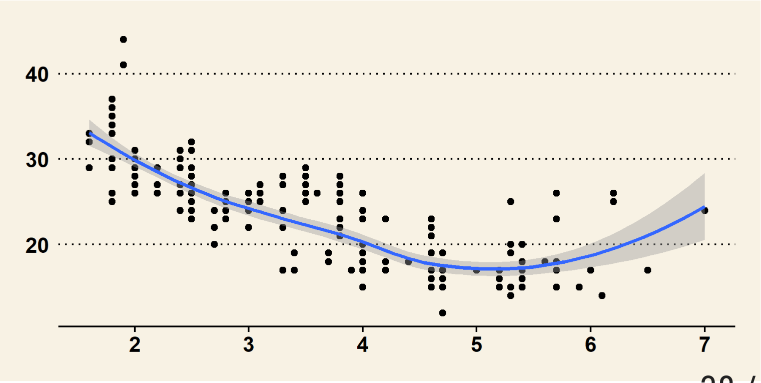
Default ggplot theme:

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



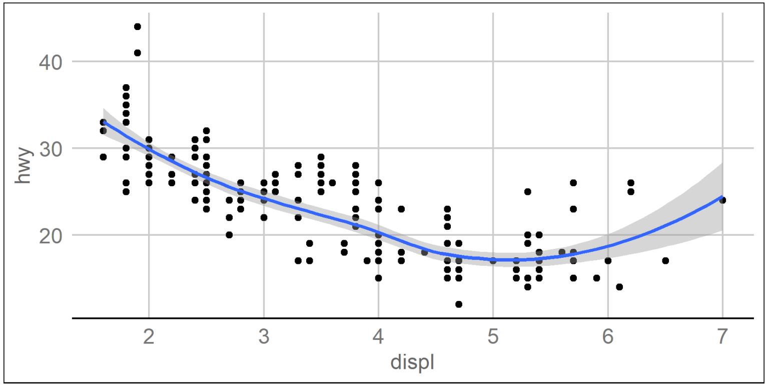
Wall Street Journal theme:

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



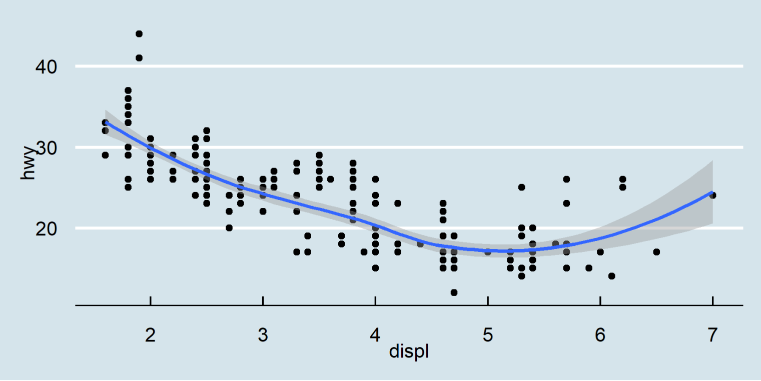
Theme based on Google Docs Chart:

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



Theme based on the Economist:

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



Export plots to files

```
p1 = ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
p2 = ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))

p1
p2

ggsave(filename = "my_plot_p2.png") #save the last plot displayed, which is p2
ggsave(filename = "my_plot_p1.png", plot = p1) #save a specific plot
```

Get inspired!

It is often useful to browse some example data visualizations and get inspired before creating your own charts.

Here are two good collections:

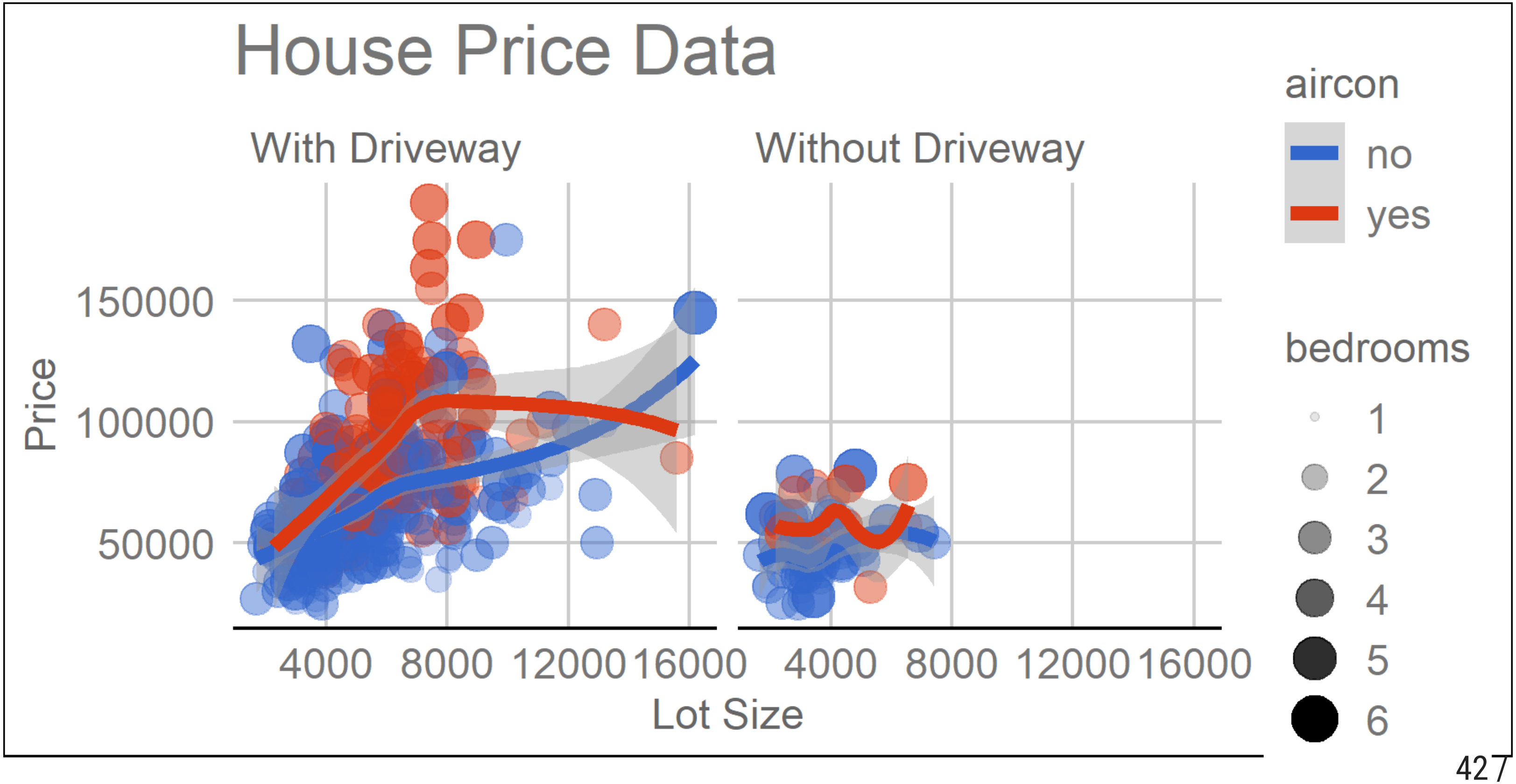
- Top 50 ggplot2 Visualizations
- The R Graph Gallery

You will notice that there are still a lot to learn about ggplot2, but I don't have time to cover all of them here. My goal is to show you the capabilities of ggplot2 along with some useful pointers so that you can start to visualize and explore data in R.

Your turn

Recreate the R code necessary to generate the following graph.

- Data: HousePrices.csv
- Theme & Color based on Google Docs Chart


House Price Data
With Driveway
Without Driveway
aircon
no
yes
bedrooms
Price
Lot Size