Topics in R

UCI Data Science Initiative

Dustin Pluta

2018-05-04

Overview

AM

- Intro to the Tidyverse
- Importing and wrangling data with `readr` and `tidyr`.
- Exploration and visualization with `ggplot2` and `dplyr`.

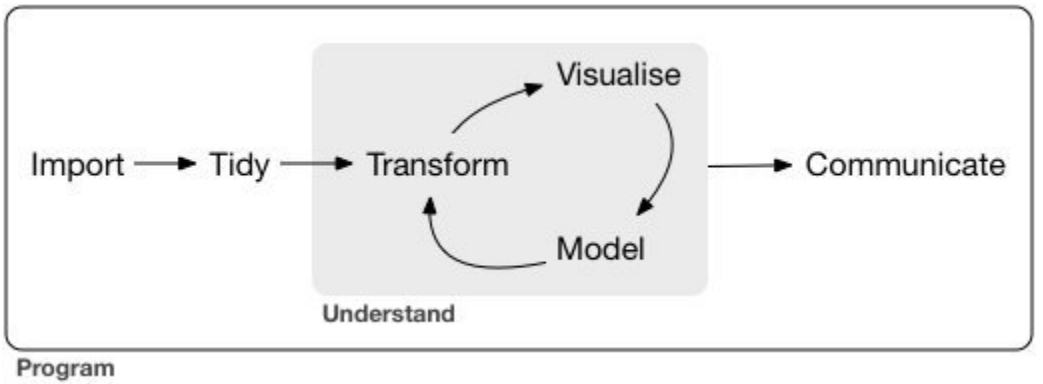
Lunch

PM

- Intro to `RMarkdown`.
- Building interactive reports with `Shiny`.

Intro to the Tidyverse

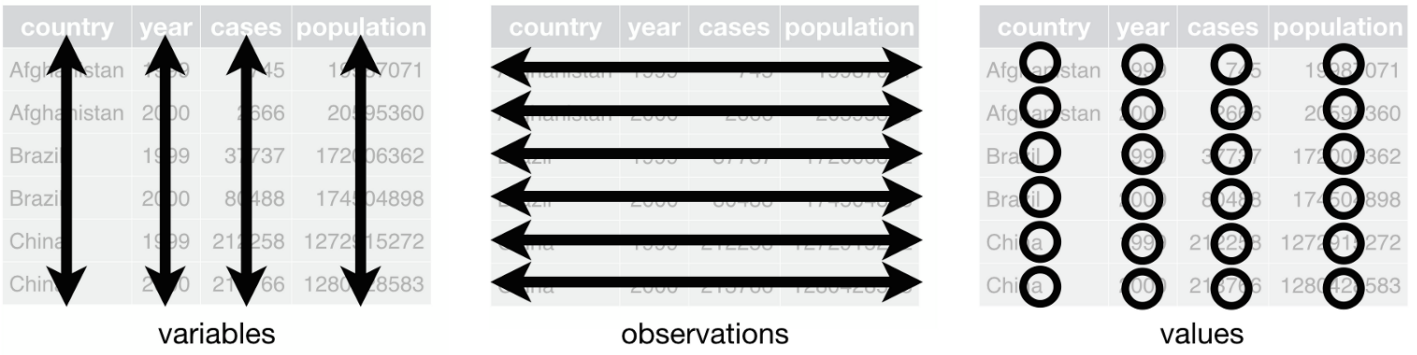
Tidy Analysis Pipeline



- *tidyverse* philosophy: collection of small, simple functions that each do one thing well
- Written by Hadley Wickham, Chief Scientist for R Studio, who also developed:
    - `ggplot2`
    - `reshape2`
    - `tidyr`
    - many others

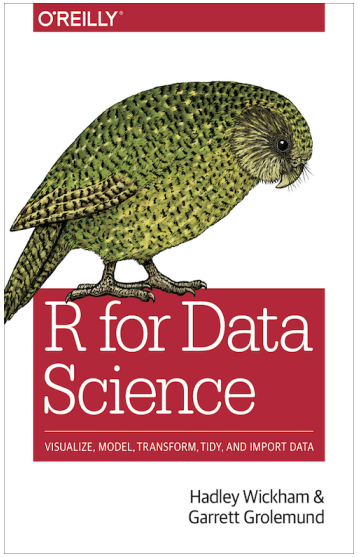
Intro to the Tidyverse

Tidy Data



Intro to the Tidyverse

Book: R for Data Science



http://r4ds.had.co.nz/

Intro to the Tidyverse

Packages

readr: import and export data

tidyr: wrangle and clean data

dplyr: slice, subset, transform, and summarize data

ggplot2: visualization

RMarkdown: preparing and presenting results

Shiny: making interactive analyses

Intro to Tidyverse: Getting Started

Install Packages

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

Load Tidyverse

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──

## ✔ ggplot2 2.2.1.9000     ✔ purrr   0.2.4
## ✔ tibble  1.4.2          ✔ dplyr   0.7.4
## ✔ tidyr   0.7.2          ✔ stringr 1.2.0
## ✔ readr   1.1.1          ✔ forcats 0.2.0

## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

Load IMDB Data

```
imdb <- read_csv("movie_metadata.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   color = col_character(),
##   director_name = col_character(),
##   actor_2_name = col_character(),
##   genres = col_character(),
##   actor_1_name = col_character(),
##   movie_title = col_character(),
##   actor_3_name = col_character(),
##   plot_keywords = col_character(),
##   movie_imdb_link = col_character(),
##   language = col_character(),
##   country = col_character(),
##   content_rating = col_character(),
##   imdb_score = col_double(),
##   aspect_ratio = col_double()
## )

## See spec(...) for full column specifications.
```

IMDB Data

```
head(imdb)
```

```
## # A tibble: 6 x 28
##   color director_name    num_critic_for_rev… duration director_facebook_l…
##   <chr> <chr>                          <int>    <int>                 <int>
## 1 Color James Cameron                  723      178                     0
## 2 Color Gore Verbinski                 302      169                   563
## 3 Color Sam Mendes                     602      148                     0
## 4 Color Christopher Nol…               813      164                 22000
## 5 <NA>  Doug Walker                     NA       NA                   131
## 6 Color Andrew Stanton                 462      132                   475
## # ... with 23 more variables: actor_3_facebook_likes <int>,
## #   actor_2_name <chr>, actor_1_facebook_likes <int>, gross <int>,
## #   genres <chr>, actor_1_name <chr>, movie_title <chr>,
## #   num_voted_users <int>, cast_total_facebook_likes <int>,
## #   actor_3_name <chr>, facenumber_in_poster <int>, plot_keywords <chr>,
## #   movie_imdb_link <chr>, num_user_for_reviews <int>, language <chr>,
## #   country <chr>, content_rating <chr>, budget <int>, title_year <int>,
## #   actor_2_facebook_likes <int>, imdb_score <dbl>, aspect_ratio <dbl>,
## #   movie_facebook_likes <int>
```

IMDB Data

```
colnames(imdb)
```

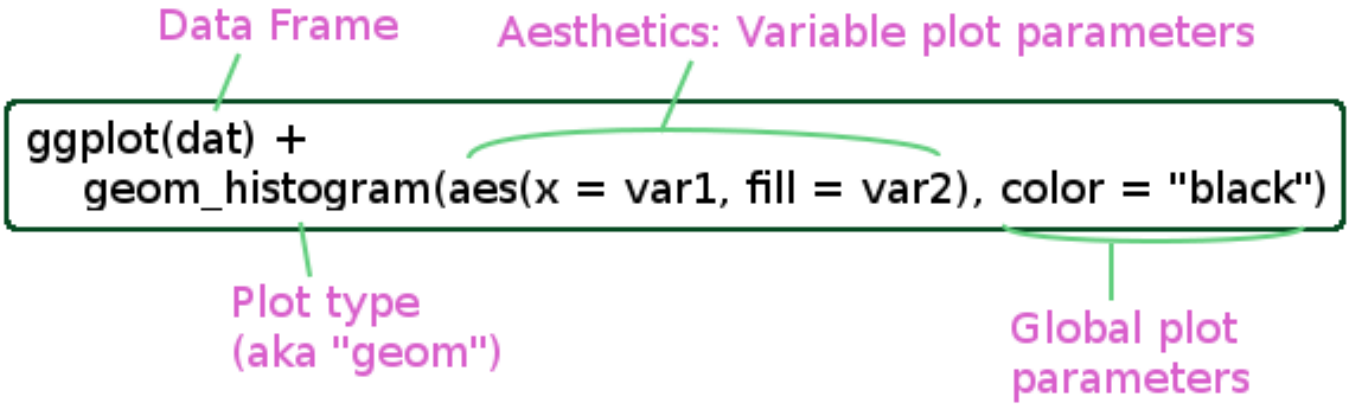
```
##  [1] "color"                   "director_name"
##  [3] "num_critic_for_reviews"  "duration"
##  [5] "director_facebook_likes" "actor_3_facebook_likes"
##  [7] "actor_2_name"            "actor_1_facebook_likes"
##  [9] "gross"                   "genres"
## [11] "actor_1_name"            "movie_title"
## [13] "num_voted_users"         "cast_total_facebook_likes"
## [15] "actor_3_name"            "facenumber_in_poster"
## [17] "plot_keywords"           "movie_imdb_link"
## [19] "num_user_for_reviews"    "language"
## [21] "country"                 "content_rating"
## [23] "budget"                  "title_year"
## [25] "actor_2_facebook_likes"  "imdb_score"
## [27] "aspect_ratio"            "movie_facebook_likes"
```

Visualization with `ggplot2`

- `ggplot2` is a plotting package that is a nice and more modern alternative to `R` base plots
- Based on the idea of a *grammar of graphics*...
  - Think of a **plot like a sentence...**
  - **noun**: the plot data
  - **verbs**: the plot types
  - **adverbs**: the plot characteristics

Visualization with `ggplot2`

Visualization with `ggplot2`



Visualization with `ggplot2`

List of `geoms`

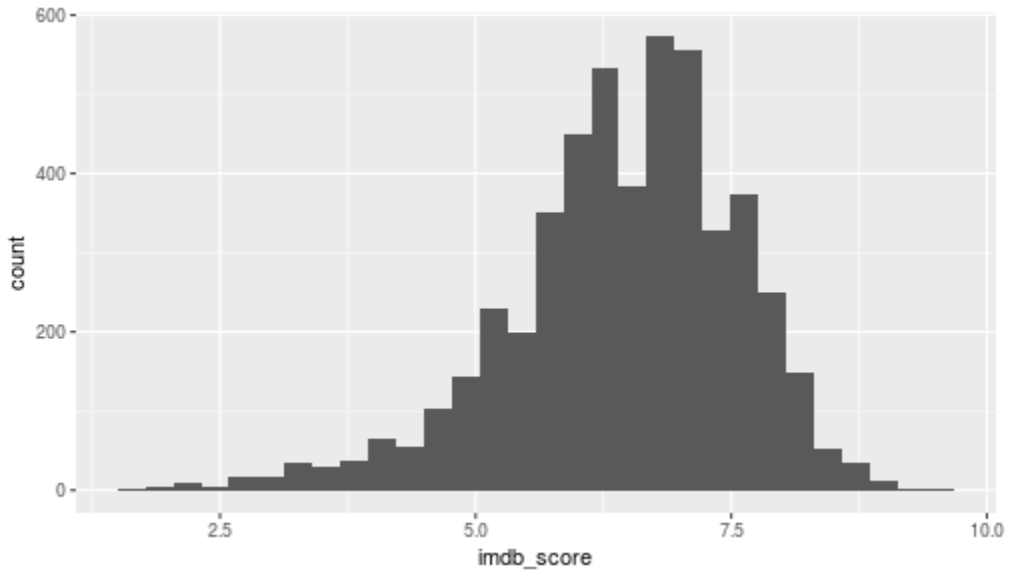
- `geom_histogram`
- `geom_density`
- `geom_point`
- `geom_line`
- `geom_boxplot`
- ... many others

Visualization with `ggplot2`

```
library(ggplot2)
```

```
ggplot(imdb) +
  geom_histogram(aes(x = imdb_score))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Visualization with `ggplot2`

Some Aesthetics

x: horizontal position

y: vertical position

alpha: transparency

color: border color

fill: interior color

group: grouping variable

linetype

size

Visualization with `ggplot2`

Different geoms have different aesthetics

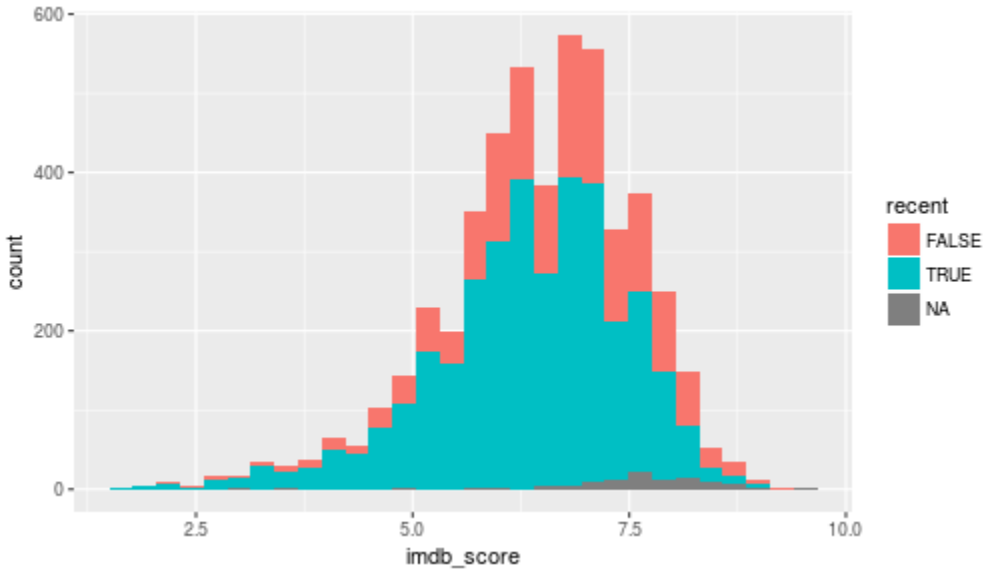
Refer to the documentation to see which aesthetics are supported for a geom

```
?geom_histogram
```

Visualization with `ggplot2`

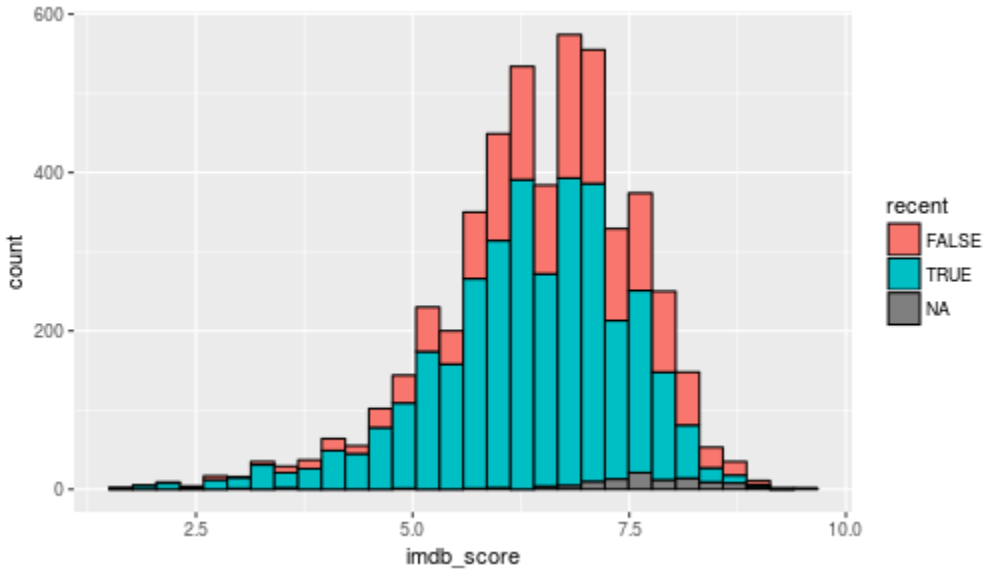
```
imdb$recent <- imdb$title_year > 2000
ggplot(imdb) +
  geom_histogram(aes(x = imdb_score, fill = recent))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



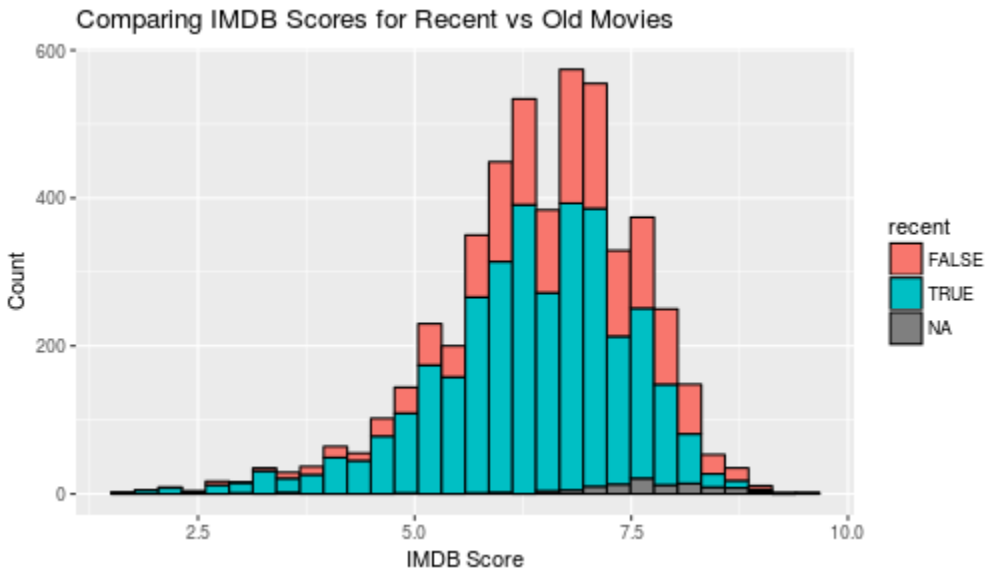
Visualization with `ggplot2`

```
imdb$recent <- imdb$title_year > 2000
ggplot(imdb) +
  geom_histogram(aes(x = imdb_score, fill = recent),
                 color = "black", bins = 30)
```



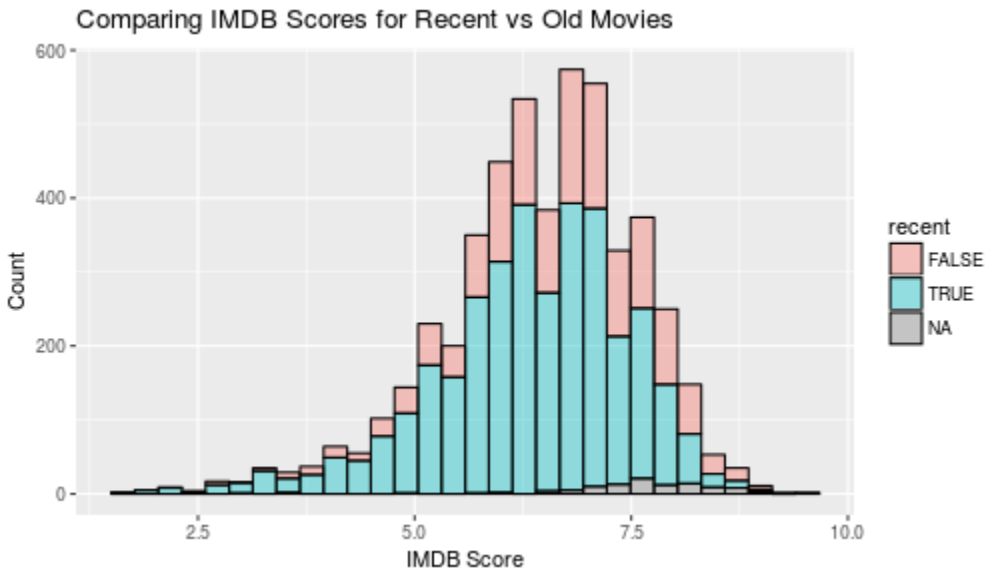
Visualization with `ggplot2`

```
ggplot(imdb) +
  geom_histogram(aes(x = imdb_score, fill = recent),
                 color = "black", bins = 30) +
  xlab("IMDB Score") +
  ylab("Count") +
  ggtitle("Comparing IMDB Scores for Recent vs Old Movies")
```



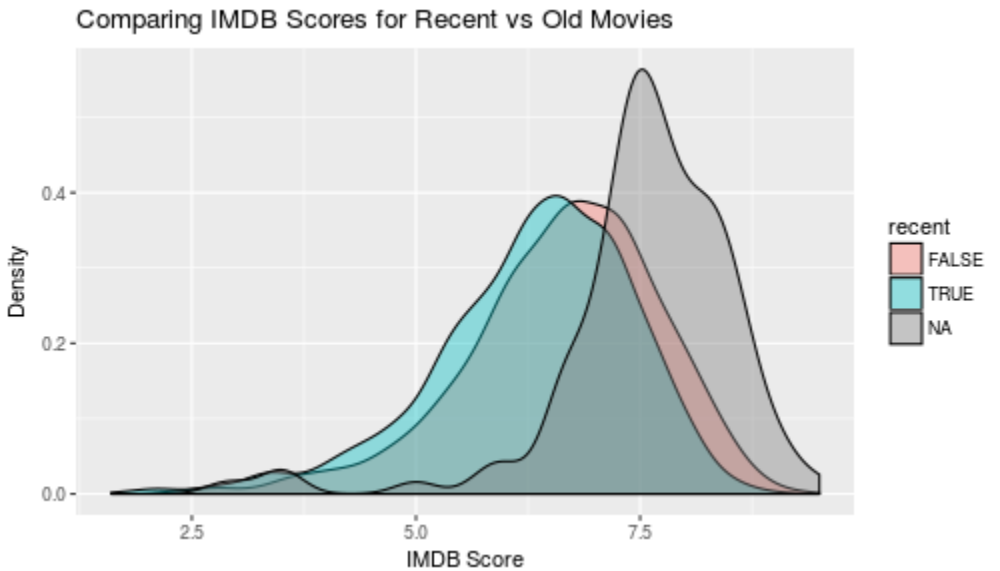
Visualization with `ggplot2`

```
ggplot(imdb) +
  geom_histogram(aes(x = imdb_score, fill = recent),
                 color = "black", bins = 30, alpha = 0.4) +
  xlab("IMDB Score") +
  ylab("Count") +
  ggtitle("Comparing IMDB Scores for Recent vs Old Movies")
```



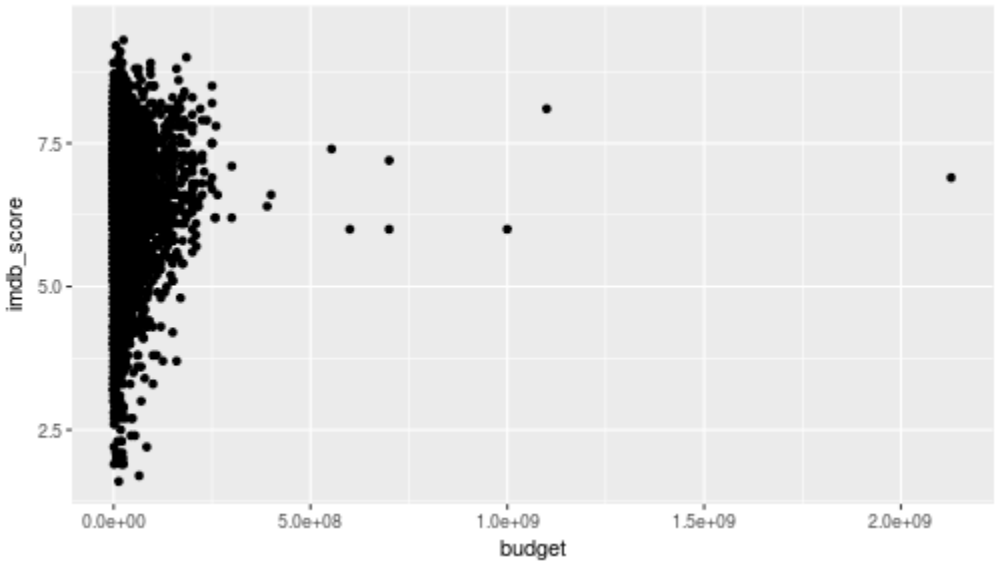
Visualization with `ggplot2`

```
ggplot(imdb) +
  geom_density(aes(x = imdb_score, fill = recent),
               color = "black", alpha = 0.4) +
  xlab("IMDB Score") +
  ylab("Density") +
  ggtitle("Comparing IMDB Scores for Recent vs Old Movies")
```



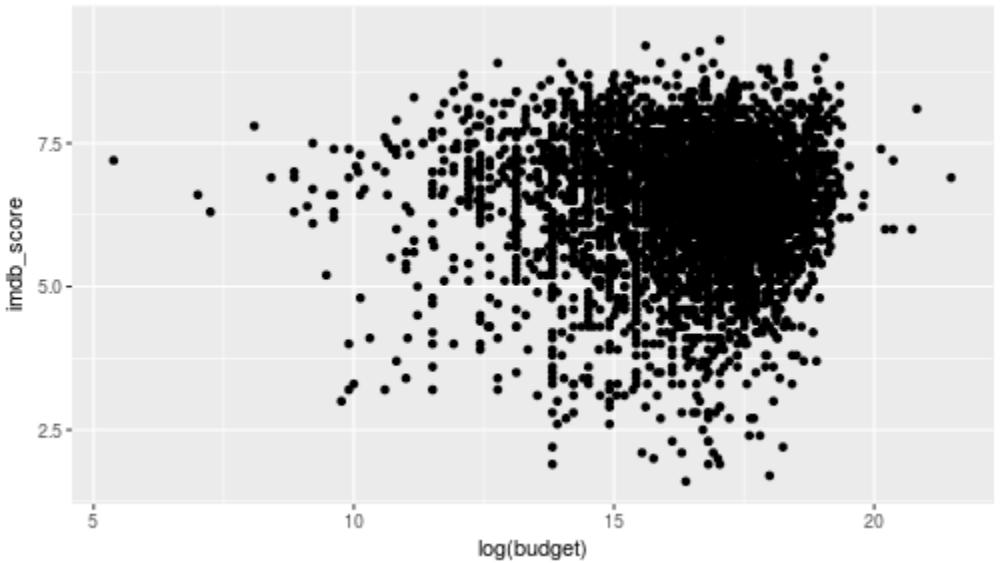
Visualization with `ggplot2`

```
ggplot(imdb) +
  geom_point(aes(x = budget, y = imdb_score))
```



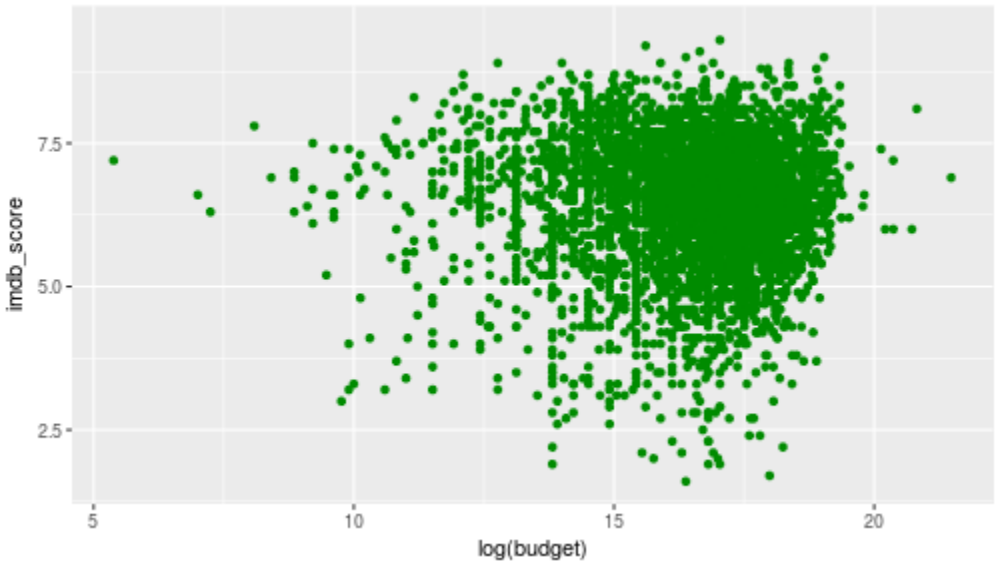
Visualization with `ggplot2`

```
ggplot(imdb) +
  geom_point(aes(x = log(budget), y = imdb_score))
```



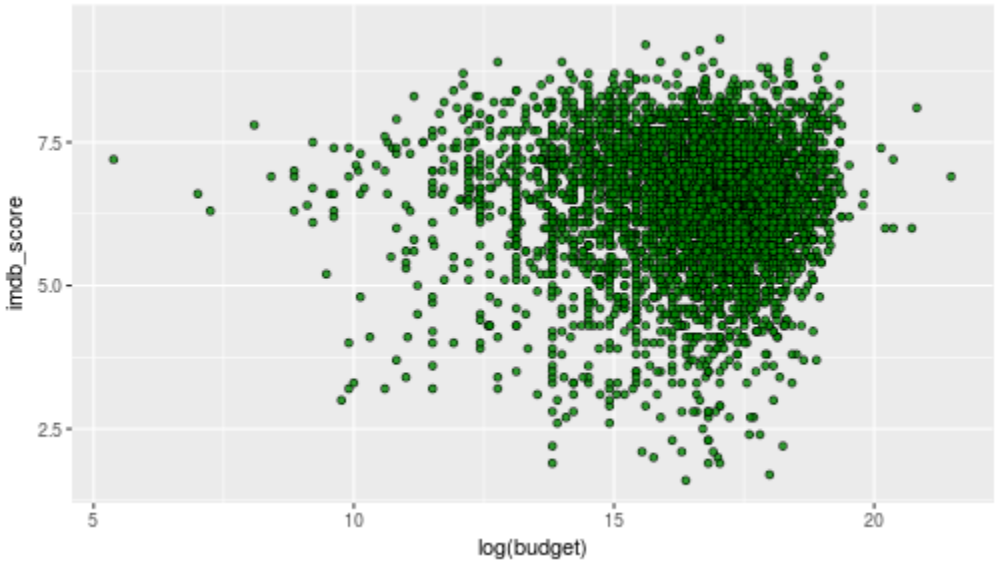
Visualization with `ggplot2`

```
ggplot(imdb) +
  geom_point(aes(x = log(budget), y = imdb_score), color = "green4")
```



Visualization with `ggplot2`

```
ggplot(imdb) +
  geom_point(aes(x = log(budget), y = imdb_score),
             pch = 21, fill = "green4",
             color = "black", alpha = 0.8)
```



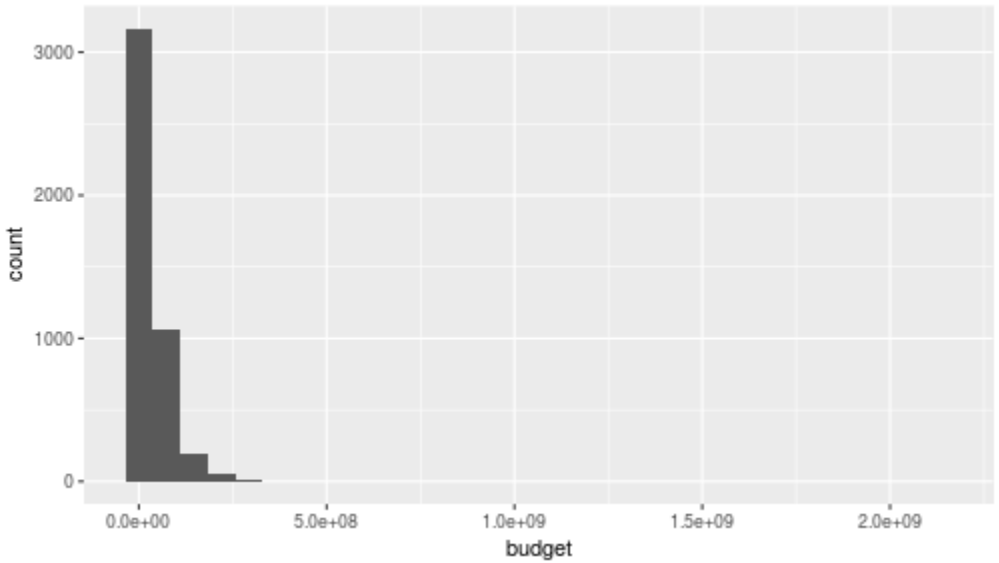
Visualization Exercises

1. Plot a histogram of `budget` and compare it to a histogram of `log(budget)`.
2. Add some color, change the title and axis labels for the `log(budget)` histogram.
3. Make a new variable `recent` to indicate if a movie is more recent than 2000 using `imdb$recent <- imdb$title_year > 2000`, then plot a histogram of `log(budget)` grouped by `recent`.
4. Create a scatterplot of `imdb_score` by `log(budget)` and colored by `recent`.
5. Create a boxplot of `imdb_score` grouped by `recent`, using `geom_boxplot`.

Visualization Exercise # 1

```
ggplot(imdb) +
  geom_histogram(aes(x = budget))
```

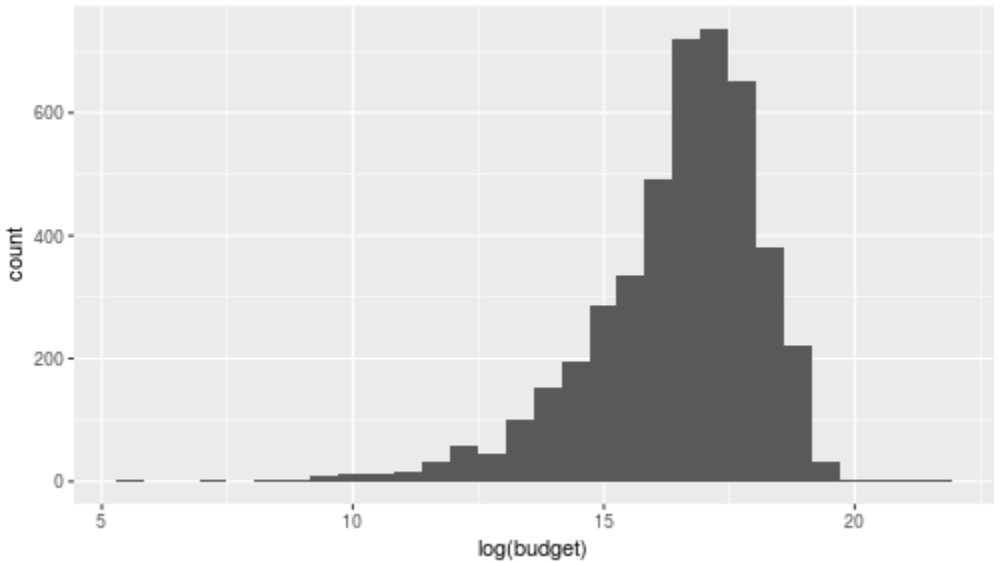
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Visualization Exercise # 1

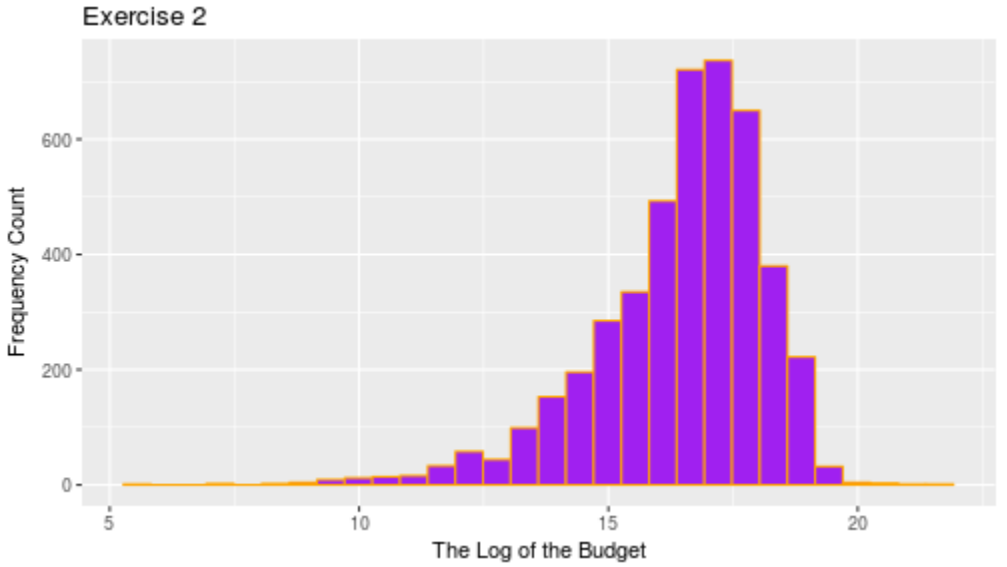
```
ggplot(imdb) +
  geom_histogram(aes(x = log(budget)))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Visualization Exercise # 2

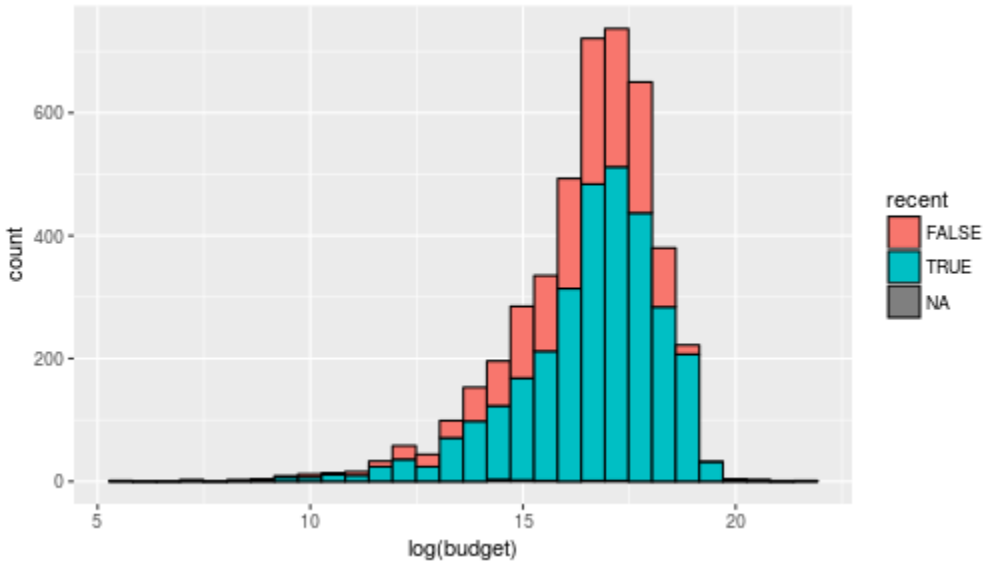
```
ggplot(imdb) +
  geom_histogram(aes(x = log(budget)),
                 color = "orange", fill = "purple", bins = 30) +
  ggtitle("Exercise 2") +
  xlab("The Log of the Budget") +
  ylab("Frequency Count")
```



Visualization Exercise # 3

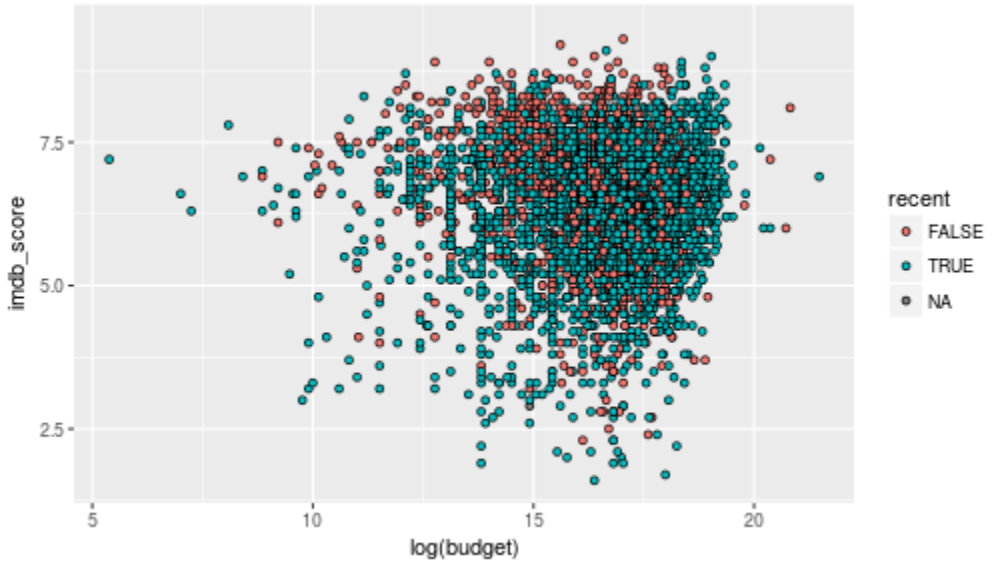
```
imdb$recent <- imdb$title_year > 2000
ggplot(imdb) +
  geom_histogram(aes(x = log(budget), fill = recent),
                 color = "black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



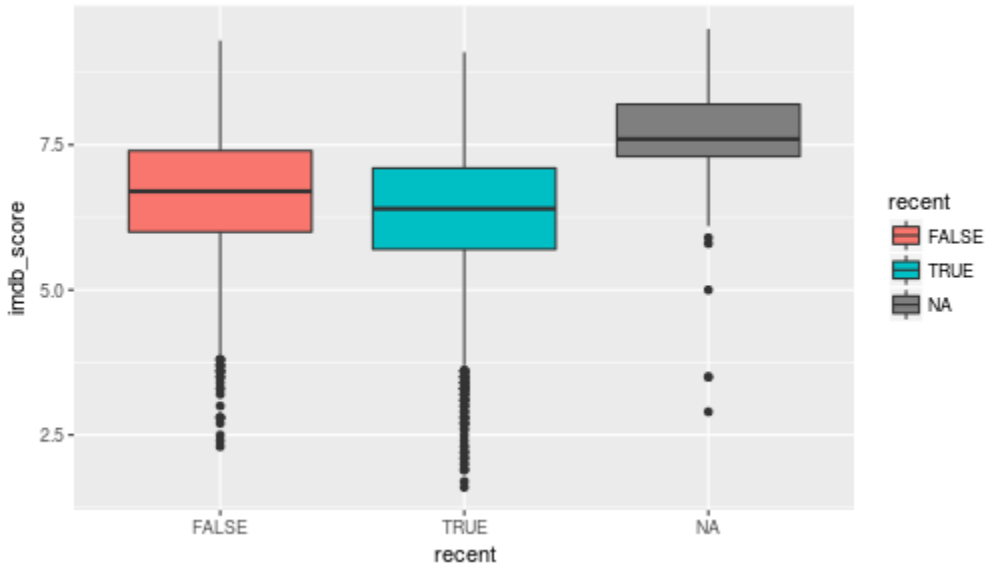
Visualization Exercise # 4

```
ggplot(imdb) +
  geom_point(aes(x = log(budget), y = imdb_score, fill = recent),
             pch = 21, color = "black")
```

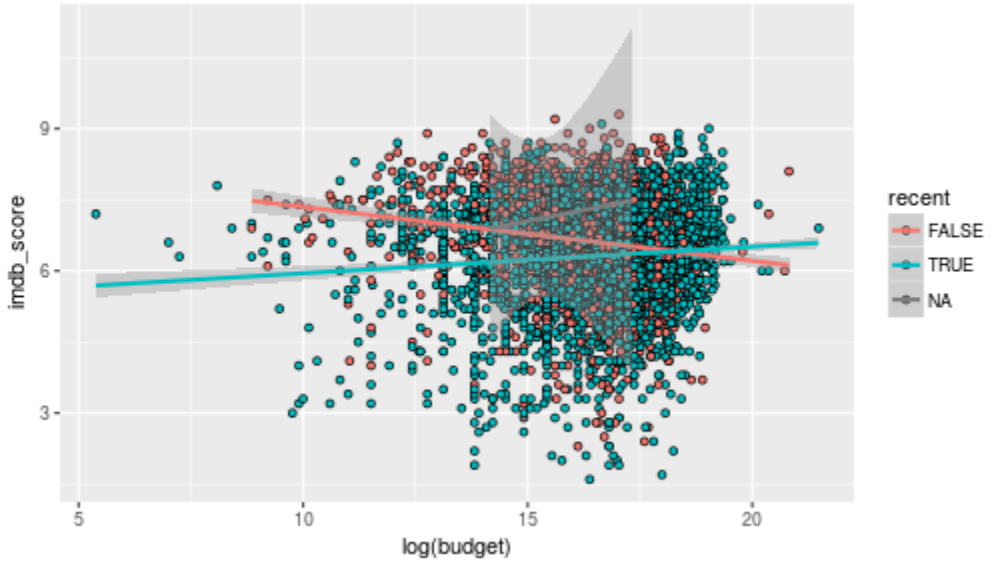


Visualization Exercises # 5

```
ggplot(imdb) +
  geom_boxplot(aes(y = imdb_score, x = recent, fill = recent))
```

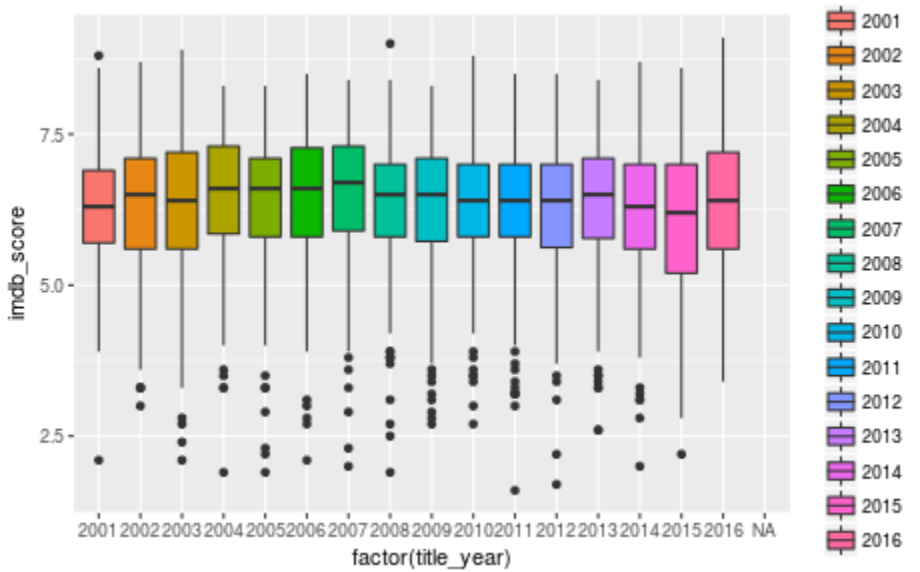


```
ggplot(imdb) +
  geom_point(aes(x = log(budget), y = imdb_score, fill = recent),
             pch = 21, color = "black") +
  geom_smooth(aes(x = log(budget), y = imdb_score, color = recent),
              method = "lm")
```



Visualization with `ggplot2`

```
ggplot(imdb[imdb$recent, ]) +
  geom_boxplot(aes(y = imdb_score, fill = factor(title_year),
                   x = factor(title_year)))
```



Data Wrangling with `dplyr`

Data Wrangling with `dplyr`

- `dplyr` is a package designed for easy and efficient data manipulation

Key Functions

- `filter`: select subset of rows (observations)
- `select`: select subset of columns (variables)
- `mutate`: transform variables in a data set
- `arrange`: reorder rows
- `summarise`: collapses a data frame into a single row
- `group_by`

Data Wrangling with `dplyr`

- Let's try some `dplyr` functions with the `iris` data set:

```
# Print species means of Sepal Width
data(iris)
iris <-  filter(iris, Species!="setosa")
iris <- select(iris, c(Sepal.Width, Species))
iris <- group_by(iris, Species)
species_means <- summarise(iris, mean(Sepal.Width))
print(species_means)
```

```
## # A tibble: 2 x 2
##   Species    `mean(Sepal.Width)`
##   <fct>                    <dbl>
## 1 versicolor                2.77
## 2 virginica                 2.97
```

Data Wrangling with `dplyr`

Introducing the Pipe: `%>%`

- `dplyr` (and much of the *tidyverse*) is designed around the use of the pipe operator `%>%`
- The pipe operator `%>%` allows you to chain operations on a data set together without having to create specific intermediate objects
- When using `%>%`, the first argument to a function is taken as the output of the previous step in the chain

Data Wrangling with `dplyr`

- For example, the following is equivalent to the previous code:

```
# Prints species means, does not save anything
# Original data.frame iris is unaffected
data(iris)
iris %>% filter(Species!="setosa") %>%
    select(c(Sepal.Width, Species)) %>%
    group_by(Species) %>%
    summarise(mean(Sepal.Width))
```

```
## # A tibble: 2 x 2
##   Species    `mean(Sepal.Width)`
##   <fct>                    <dbl>
## 1 versicolor                2.77
## 2 virginica                 2.97
```

Data Wrangling with `dplyr`

```
# To save the results instead
species_means <- iris %>%
    filter(Species!="setosa") %>%
    select(c(Sepal.Width, Species)) %>%
    group_by(Species) %>%
    summarise(mean(Sepal.Width))
```

```
species_means
```

```
## # A tibble: 2 x 2
##   Species    `mean(Sepal.Width)`
##   <fct>                    <dbl>
## 1 versicolor                2.77
## 2 virginica                 2.97
```

`dplyr` Exercises

1. Use `dplyr` to calculate the mean Sepal Width of the virginica species.

2. `summarise` can summarise multiple variables simultaneously, applying a (possibly different) function to each variable.
   Adapt the code below to find the minimum, median, maximum, and standard deviation of the Sepal.Width for the virginica species.

3. `group_by()` makes `summarise` even more useful by allowing you to summarise values across groups of a category simultaneously.
   Using `group_by`, adapt your code from the previous problem to produce the summary values for each species.

Modify this code for problems 2 and 3:

```
data(iris)
iris %>% summarise(mean_sepal_width = mean(Sepal.Width),
                   min_sepal_width = min(Sepal.Width))
```

dplyr Exercises

Solution

- (1) Use dplyr to calculate the mean Sepal Width of the virginica species.

```
data(iris)

iris %>%
    filter(Species == "virginica") %>%
    summarise(mean_sepal_width = mean(Sepal.Width))
```

```
##   mean_sepal_width
## 1            2.974
```

dplyr Exercises

Solution

- (2) `summarise` can summarise multiple variables simultaneously, applying a (possibly different) function to each variable.
  Adapt the code below to find the minimum, median, maximum, and standard deviation of the Sepal.Width for the virginica species.

```
data(iris)
iris %>%
    filter(Species == "virginica") %>%
    summarise(min_sepal_width = min(Sepal.Width),
              med = median(Sepal.Width), maximum = max(Sepal.Width),
              stdev = sd(Sepal.Width))
```

```
##   min_sepal_width med maximum     stdev
## 1             2.2   3     3.8 0.3224966
```

dplyr Exercises

Solution

- (3) `group_by()` makes `summarise` even more useful by allowing you to summarise values across groups of a category simultaneously.
  Using `group_by`, adapt your code from the previous problem to produce the summary values for each species.

```
data(iris)
iris %>%
    group_by(Species) %>%
    summarise(min_sepal_width = min(Sepal.Width),
              med = median(Sepal.Width), maximum = max(Sepal.Width),
              stdev = sd(Sepal.Width))
```

```
## # A tibble: 3 x 5
##   Species    min_sepal_width   med maximum stdev
##   <fct>                <dbl> <dbl>   <dbl> <dbl>
## 1 setosa                2.30  3.40    4.40 0.379
## 2 versicolor            2.00  2.80    3.40 0.314
## 3 virginica             2.20  3.00    3.80 0.322
```

Back to the Movies

Exploring the IMDB Data

How many movies for each actor in the dataset?

```
imdb %>%
  group_by(actor_1_name) %>%
  summarize(n())
```

Exploring the IMDB Data

How many movies for each actor in the dataset?

```
imdb %>%
  group_by(actor_1_name) %>%
  summarize(n())
```

```
## # A tibble: 2,098 x 2
##    actor_1_name      `n()`
##    <chr>             <int>
##  1 50 Cent               1
##  2 Aaliyah               1
##  3 Aasif Mandvi          1
##  4 Abbie Cornish         3
##  5 Abhishek Bachchan     2
##  6 Abigail Evans         1
##  7 Abigail Spencer       1
##  8 Adam Arkin            2
##  9 Adam Baldwin          4
## 10 Adam Garcia           3
## # ... with 2,088 more rows
```

Exploring the IMDB Data

How many movies for each actor in the dataset?

Arranged by decreasing number of movies

```
imdb %>%
  group_by(actor_1_name) %>%
  summarize(n_movies = n()) %>%
  arrange(desc(n_movies))
```

```
## # A tibble: 2,098 x 2
##    actor_1_name      n_movies
##    <chr>                <int>
##  1 Robert De Niro          49
##  2 Johnny Depp             40
##  3 Nicolas Cage            32
##  4 J.K. Simmons            31
##  5 Bruce Willis            30
##  6 Denzel Washington       30
##  7 Matt Damon              30
##  8 Liam Neeson             29
##  9 Harrison Ford           27
## 10 Robin Williams          27
```

Exploring the IMDB Data

How many movies for each actor in the dataset?

Arranged by decreasing mean IMDB score

```
imdb %>%
  group_by(actor_1_name) %>%
  summarize(mean_imdb_score = mean(imdb_score)) %>%
  arrange(desc(mean_imdb_score))
```

```
## # A tibble: 2,098 x 2
##    actor_1_name       mean_imdb_score
##    <chr>                        <dbl>
##  1 Krystyna Janda                9.10
##  2 Jack Warden                   8.90
##  3 Rob McElhenney                8.80
##  4 Abigail Evans                 8.70
##  5 Elina Abai Kyzy               8.70
##  6 Jackie Gleason                8.70
##  7 Kimberley Crossman            8.70
##  8 Maria Pia Calzone             8.70
##  9 Takashi Shimura               8.70
## 10 Bunta Sugawara                8.60
```

Exploring the IMDB Data

Considering actors with more than 5 movies, list top 10 actors with highest mean IMDB scores in decreasing order.

```
imdb %>%
  group_by(actor_1_name) %>%
  summarize(mean_imdb_score = mean(imdb_score), n_movies = n()) %>%
  filter(n_movies > 5) %>%
  top_n(10, mean_imdb_score) %>%
  arrange(desc(mean_imdb_score))
```

```
## # A tibble: 10 x 3
##    actor_1_name           mean_imdb_score n_movies
##    <chr>                            <dbl>    <int>
##  1 Leonardo DiCaprio                 7.50       21
##  2 Tom Hanks                         7.42       24
##  3 Clint Eastwood                    7.34       16
##  4 Tom Hardy                         7.31       11
##  5 Alan Rickman                      7.29        8
##  6 Benedict Cumberbatch              7.29        7
##  7 Philip Seymour Hoffman            7.24       20
##  8 Toby Jones                        7.22        6
##  9 Minnie Driver                     7.21        7
```

Exploring the IMDB Data

How many movie entries does Harrison Ford have?

```
actor <- "Harrison Ford"
filter(imdb, actor_1_name == actor) %>% nrow
```

```
## [1] 27
```

Exploring the IMDB Data

Harrison Ford IMDB Scores

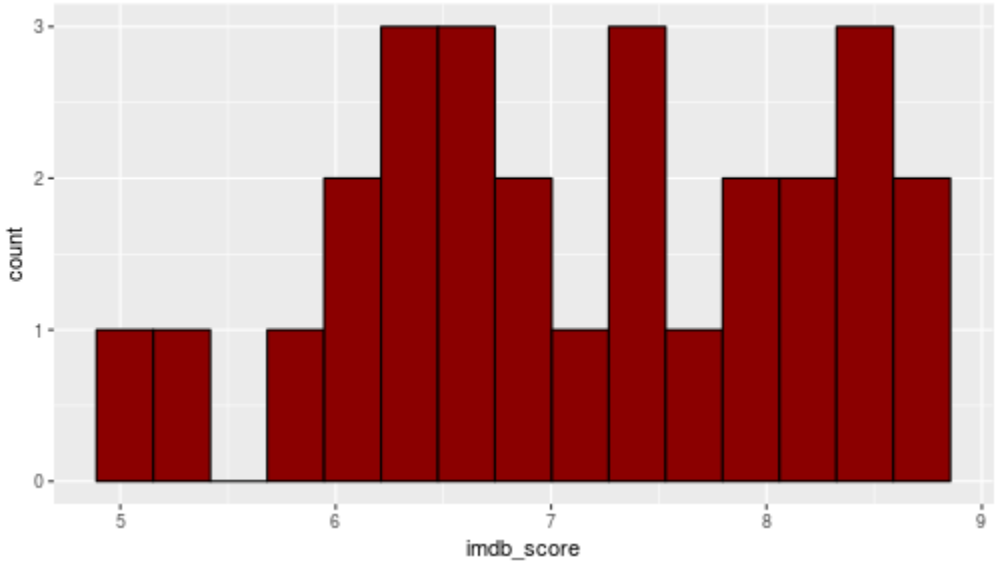
```
imdb %>%
  filter(actor_1_name == actor) %>%
  select(imdb_score)
```

```
## # A tibble: 27 x 1
##    imdb_score
##         <dbl>
##  1       6.20
##  2       6.70
##  3       6.60
##  4       6.10
##  5       6.40
##  6       5.70
##  7       5.30
##  8       5.10
##  9       6.90
## 10       6.30
## # ... with 17 more rows
```

Exploring the IMDB Data

Histogram of Harrison Ford IMDB Scores

```
ggplot(imdb %>% filter(actor_1_name == actor)) +
  geom_histogram(aes(x = imdb_score),
                 fill = "darkred", color = "black", bins = 15)
```



Exploring the IMDB Data

Harrison By Genre

```
imdb %>%
  filter(actor_1_name %in% actor) %>%
  group_by(genres) %>%
  summarize(mean_score = mean(imdb_score), n_movies = n())
```

```
## # A tibble: 22 x 3
##    genres                                          mean_score n_movies
##    <chr>                                                <dbl>    <int>
##  1 Action|Adventure                                      8.05        2
##  2 Action|Adventure|Comedy|Romance                       5.70        1
##  3 Action|Adventure|Crime|Drama|Mystery|Thriller         7.80        1
##  4 Action|Adventure|Drama|Thriller                       6.40        1
##  5 Action|Adventure|Fantasy                              7.25        2
##  6 Action|Adventure|Fantasy|Sci-Fi                       8.63        3
##  7 Action|Comedy|Crime|Thriller                          5.30        1
##  8 Action|Crime|Drama|Thriller                           6.50        2
##  9 Action|Drama|War                                      6.30        1
## 10 Action|Sci-Fi                                         6.70        1
## # ... with 12 more rows
```

Exploring the IMDB Data

Harrison's Action Movies

```
imdb %>%
  filter(actor_1_name %in% actor) %>%
  transmute(action = str_detect(genres, "Action"))
```

```
## # A tibble: 27 x 1
##    action
##    <lgl>
##  1 T
##  2 T
##  3 F
##  4 T
##  5 T
##  6 T
##  7 T
##  8 F
##  9 T
## 10 F
## # ... with 17 more rows
```

Exploring the IMDB Data

Harrison's Action Movies

```
imdb %>%
  filter(actor_1_name %in% actor) %>%
  mutate(action = str_detect(genres, "Action")) %>%
  group_by(action) %>%
  summarize(mean(imdb_score))
```

```
## # A tibble: 2 x 2
##   action `mean(imdb_score)`
##   <lgl>                <dbl>
## 1 F                     7.15
## 2 T                     7.16
```

Exercises: Exploring the IMDB Data

Liam vs Harrison

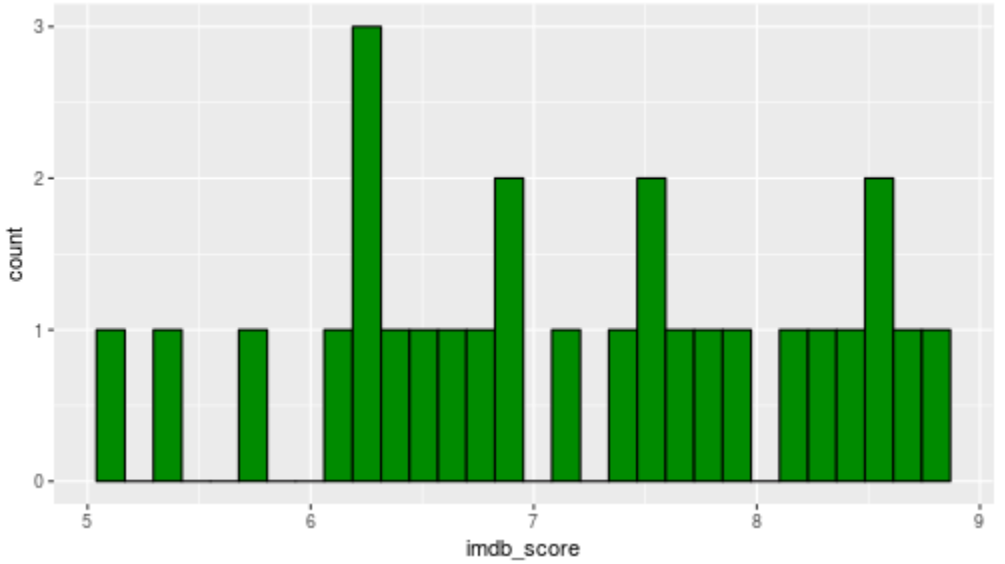
- (1) Plot a histogram of Liam Neeson's IMDB scores
- (2) Create side-by-side boxplots of Harrison's and Liam's IMDB scores. Overlay the specific points on top of the boxplots with `geom_point`.
- (3) Find the median gross earned for Harrison movies and Liam movies.
- (4) Create a scatterplot of IMDB scores by `log(gross)` for just Harrison and Liam, and color the points according to the actor.
- (5) Do Harrison's action movies or Liam's drama movies have larger median gross?

Liam vs Harrison

(1) Plot a histogram of Harrison Ford's IMDB scores

```
ggplot(imdb %>% filter(actor_1_name == "Harrison Ford")) +
  geom_histogram(aes(x = imdb_score), fill = "green4", color = "black
```

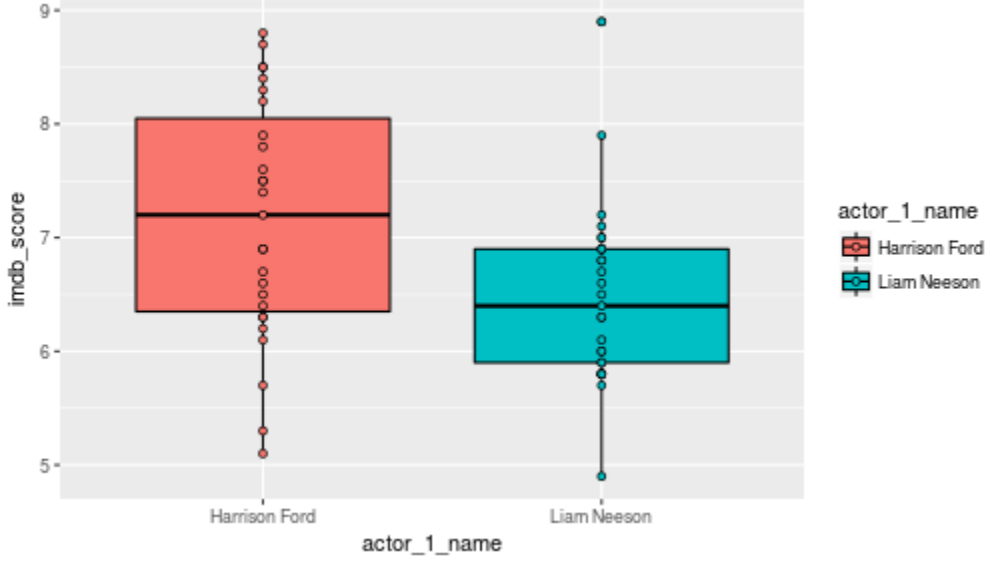
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Liam vs Harrison

(2) Create side-by-side boxplots of Liam's and Harrison's IMDB scores. Overlay the specific points on top of the boxplots with `geom_point`.

```
ggplot(imdb %>% filter(actor_1_name %in% c("Liam Neeson", "Harrison F
  geom_boxplot(aes(y = imdb_score, x = actor_1_name, fill = actor_1_n
                color = "black") +
  geom_point(aes(x = actor_1_name, y = imdb_score, fill = actor_1_nam
             color = "black", pch = 21)
```



Liam vs Harrison

(3) Find the median gross earned for both actors.

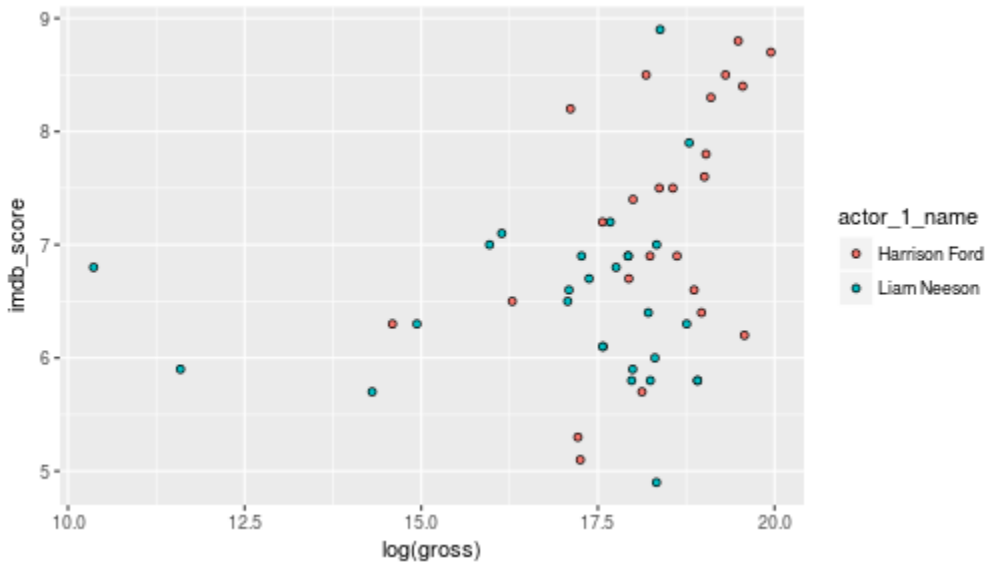
```
imdb %>% filter(actor_1_name %in% c("Liam Neeson", "Harrison Ford"))
  group_by(actor_1_name) %>%
  summarize(median_gross = median(gross, na.rm = TRUE))
```

```
## # A tibble: 2 x 2
##   actor_1_name  median_gross
##   <chr>                <int>
## 1 Harrison Ford     95001343
## 2 Liam Neeson       61094903
```

Liam vs Harrison

(4) Create a scatterplot of IMDB scores by `log(gross)` for just Harrison and Liam, and color the points according to the actor.

```
imdb_HL <- imdb %>% filter(actor_1_name %in% c("Liam Neeson", "Harris
ggplot(imdb_HL) +
  geom_point(aes(x = log(gross), y = imdb_score, fill = actor_1_name)
             color = "black", pch = 21)
```



Liam vs Harrison

(5) Do Harrison's action movies or Liam's drama movies have larger median gross?

```
imdb %>%
  filter(actor_1_name %in% "Harrison Ford") %>%
  mutate(action = str_detect(genres, "Action")) %>%
  group_by(actor_1_name, action) %>%
  summarize(mean(imdb_score)) %>%
  filter(action == TRUE)
```

```
## # A tibble: 1 x 3
## # Groups:   actor_1_name [1]
##   actor_1_name  action `mean(imdb_score)`
##   <chr>         <lgl>                <dbl>
## 1 Harrison Ford T                     7.16
```

```
imdb %>%
  filter(actor_1_name == "Liam Neeson") %>%
  mutate(drama = str_detect(genres, "Drama")) %>%
  group_by(actor_1_name, drama) %>%
  summarize(mean(imdb_score)) %>%
  filter(drama == TRUE)
```

Intro to RMarkdown

Needed Packages

- `shiny`
- `flexdashboard`
- `plotly`
- `stargazer`

Optional Packages

- `biclust`
- `xaringan`

Intro to R Markdown

- **R Markdown** is an implementation of the *Markdown* markup language
- Markdown is a versatile tool that makes it easy to make readable scientific documents in a variety of formats
- R markdown is actively developed and supported by the RStudio team, which means:
    - RStudio has many tools and features to make R Markdown flexible and easy to use
    - New R Markdown features and packages are frequently released

Intro to R Markdown

- **R For Data Science** on the intent of R Markdown:

R Markdown files are designed to be used in three ways:

1. *For communicating to decision makers, who want to focus on the conclusions, not the code behind the analysis.*

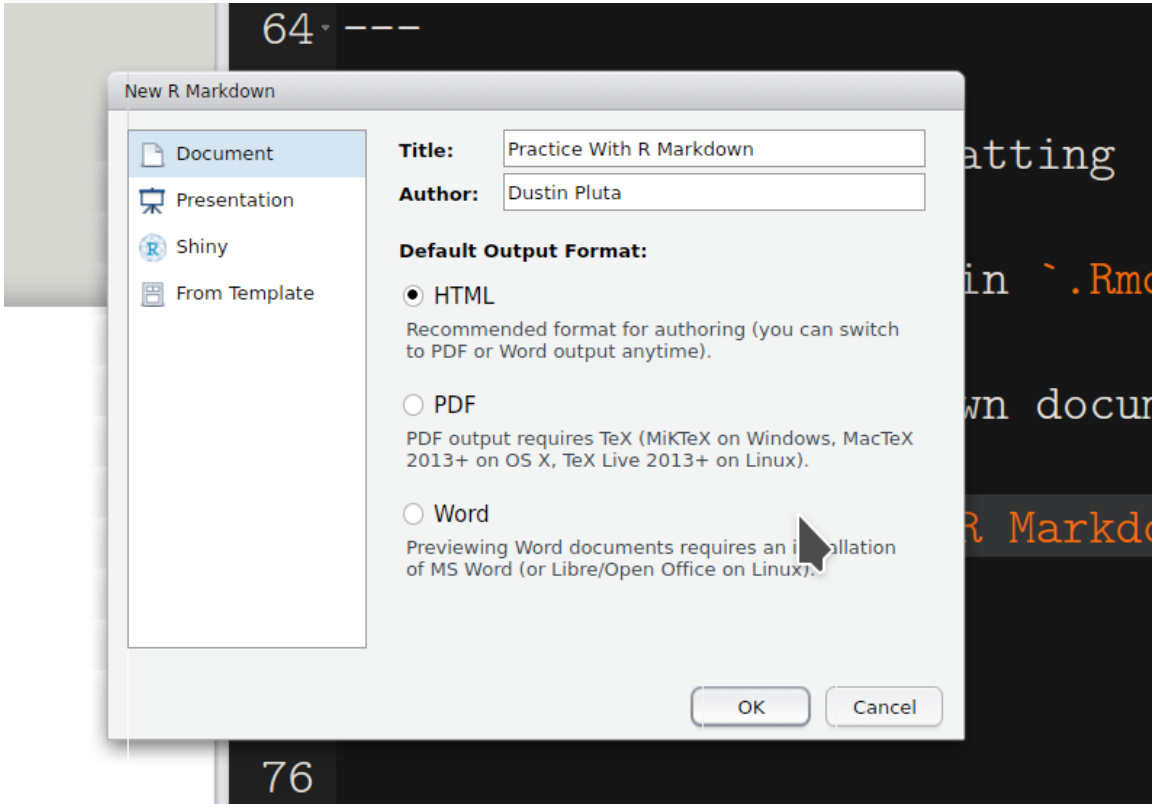
2. *For collaborating with other data scientists (including future you!), who are interested in both your conclusions, and how you reached them (i.e. the code).*

3. *As an environment in which to do data science, as a modern day lab notebook where you can capture not only what you did, but also what you were thinking.*

Getting Started with R Markdown

- R Markdown files end in `.Rmd`
- Create a new R markdown document in RStudio:
    - `File > New File > R Markdown...`

Getting Started with R Markdown



Getting Started with R Markdown

- The default R Markdown template gives some examples of basic R Markdown features

Getting Started with R Markdown

Getting Started with R Markdown

Getting Started with R Markdown

Getting Started with R Markdown

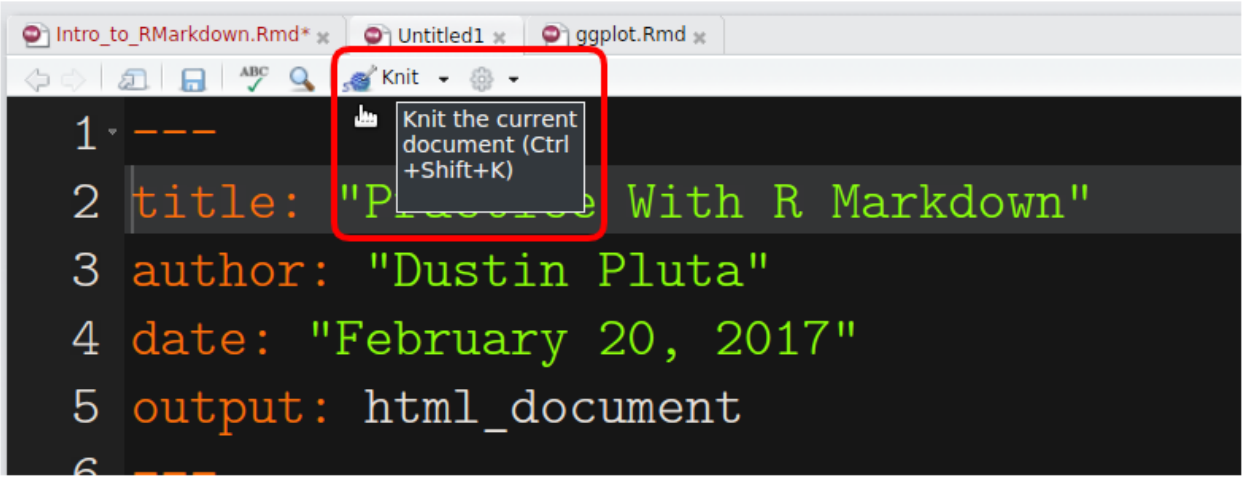
Getting Started with R Markdown

Getting Started with R Markdown

Getting Started with R Markdown

Getting Started with R Markdown

- Compile or "knit" the R Markdown document to the desired format (either html, pdf, or Word document)



Getting Started with R Markdown

Practice With R Markdown

Dustin Pluta

February 20, 2017

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

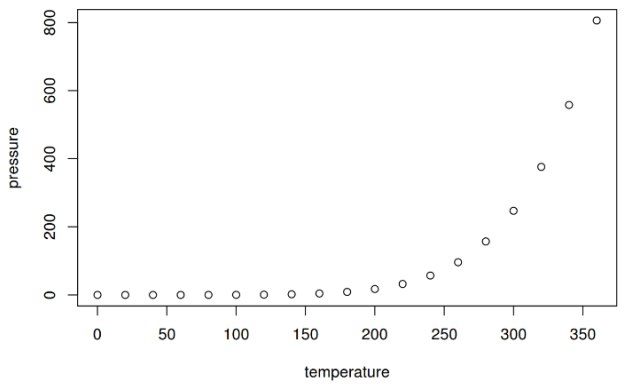
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

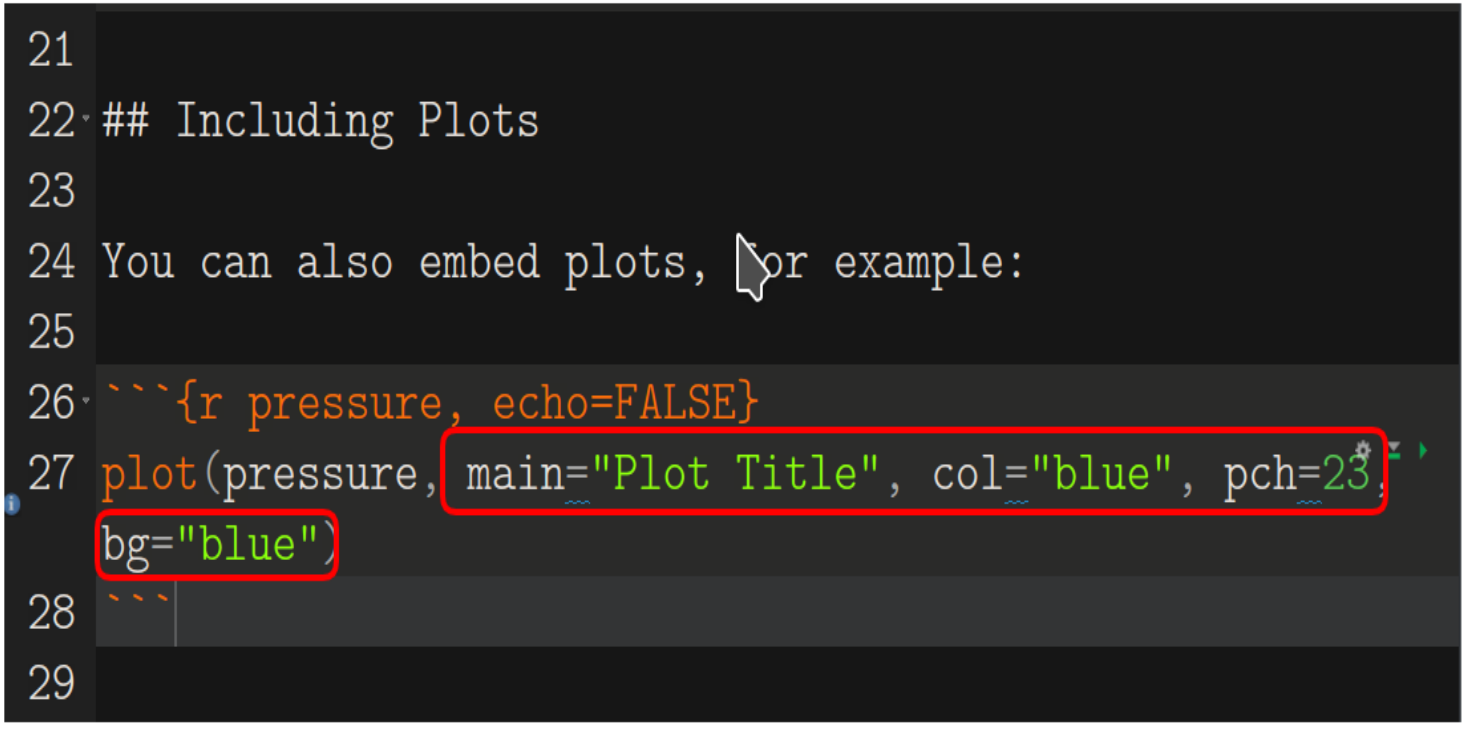
Including Plots

You can also embed plots, for example:



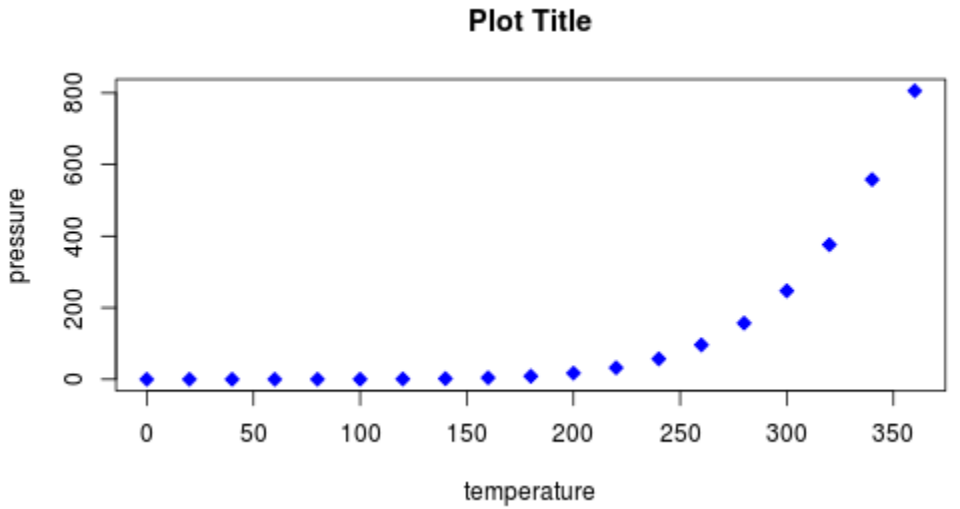
Getting Started with R Markdown

- Let's modify the plot to include a title, and make the points blue.



Getting Started with R Markdown

```
plot(pressure, main = "Plot Title", pch = 23, col = "blue", bg = "blu
```



Practice With R Markdown

R Markdown Cheat Sheet

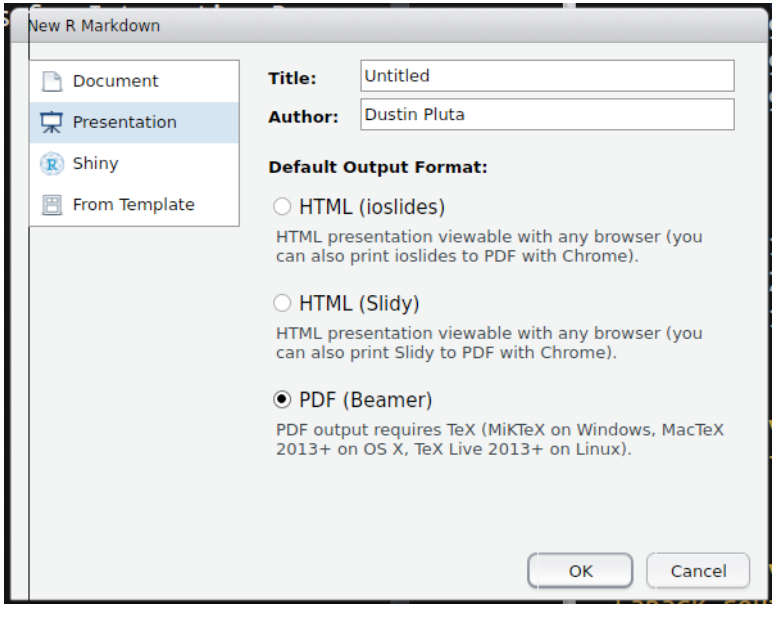
1. R Markdown Basic Example: Shows simple plot with `cars` data set.
2. R Markdown Exercise Set 1: More examples and some exercises to try on the Iris and IMDB data sets.

More R Markdown Features

1. Presentations: beamer, ioslides, slidy, xaringan
2. knitr
3. Blogdown
4. Bookdown
5. Interactive Documents

Presentations

- You can easily create academic presentations using 4 different formats
  - beamer (pdf)
  - ioslides (html)
  - slidy (html)
  - xaringan (html)



knitr

- R Markdown can make full use of Latex through the `knitr` package
- `knitr` lets you easily display mathematical formulas and other Latex formatting in your Markdown document
- For example, math can be inserted inline like $\alpha^2 + \beta^2 = \gamma^2$ or in display mode:

$$Y = X\beta + \varepsilon$$

$$\int_{\mathbb{R}} \sum_{i=1}^{n} \nabla \ell_i d\mu$$

Blogdown

Making a Website Using Blogdown, Hugo, and GitHub pages/

Example Blogdown Blog: Simply Statistics

Next Steps

- RMarkdown_Basic_Example.Rmd
- Iris_Example.Rmd
- RMarkdown_Exercise_Set1.Rmd
- Interactive_RMarkdown_Example.Rmd
- RMarkdown_Exercise_Set2.RMD
- IMDB_Dashboard.Rmd

Some Resources for R

- `dplyr` and Data Wrangling Cheat Sheet
- R Markdown Cheat Sheet
- Data Carpentry Lessons for R
- `dplyr` Tutorial
- Advanced R
- R for Data Science
- Coursera Data Science Specialization