An introduction to R

Dr. Jonathan Leslie
Head of Data Science at Pivigo

jonleslie
@jlesliedata
jonathan.leslie@pivigo.com

2018-08-15
Science to Data Science - London 2018

pivigo
THE DATA SCIENCE HUB

Table of Contents

Part 1

- Why R?
- A tour of RStudio
- Getting started with R
- Data wrangling

Part 2

- Data visualization
- Data tidying
- Exploratory data analysis

Why R?

- Open source
- Powerful statistics
- Outstanding for plotting
- Shiny dashboards
- Community

Return to Table of Contents

Acknowledgement

Dr. Chester Ismay
Senior Curriculum Lead at DataCamp

ismayc
@old_man_chester
chester@datacamp.com

Slides available at http://bit.ly/ness-infer

#rstats

Return to Table of Contents

R4DS learning community

Return to Table of Contents

A tour of RStudio

Return to Table of Contents

Designed for the novice / Nice for the practioner

For much of this book, we will assume that you are using R via RStudio. First time users often confuse the two. At its simplest:

- R is like a car's engine
- RStudio is like a car's dashboard

R: Engine	RStudio: Dashboard

Return to Table of Contents

Designed for the novice / Nice for the practioner

A good analogy for R packages is they are like apps you can download onto a mobile phone:

R: A new phone	R Packages: Apps you can download
	GET IT ON Google Play Download on the App Store

Return to Table of Contents

GO TO RSTUDIO

Return to Table of Contents

Prior Installation

Make sure you have the (current) up-to-date R, RStudio, and R packages

- Beginner's Guide on ModernDive.com

- R (version 3.4.4)
- RStudio (version 1.1.442)
- Run this in the RStudio Console

```
pkgs <- c("tidyverse", "moderndive", "gapminder",
          "nycflights13", "fivethirtyeight", "janitor",
          "ggplot2movies", "remotes")

install.packages(pkgs)
remotes::install_github("andrewpbray/infer")
```

Return to Table of Contents

R for Data Science

- Link for the book is here

Return to Table of Contents

Freely available information

ModernDive

A Modern Dive into Data with R

- Webpage: http://moderndive.com
- Developmental version at https://moderndive.netlify.com
- GitHub Repo
- Please signup for our mailing list!

Return to Table of Contents

Getting started with R

Return to Table of Contents

R Data Types

Return to Table of Contents

The bare minimum needed for understanding today

Vector/variable

- Type of vector (`int`, `num` or `dbl`, `chr`, `lgl`, `date`)

Return to Table of Contents

The bare minimum needed for understanding today

Vector/variable

- Type of vector (`int`, `num` or `dbl`, `chr`, `lgl`, `date`)

Data frame

- Vectors of (potentially) different types
- Each vector has the same number of rows

Return to Table of Contents

The bare minimum needed for understanding today

```
library(tibble)
library(lubridate)
ex1 <- data_frame(
    vec1 = c(1980, 1990, 2000, 2010),
    vec2 = c(1L, 2L, 3L, 4L),
    vec3 = c("low", "low", "high", "high"),
    vec4 = c(TRUE, FALSE, FALSE, FALSE),
    vec5 = ymd(c("2017-05-23", "1776/07/04",
                 "1983-05/31", "1908/04-01"))
  )
ex1
```

Return to Table of Contents

The bare minimum needed for understanding today

```r
library(tibble)
library(lubridate)
ex1 <- data_frame(
    vec1 = c(1980, 1990, 2000, 2010),
    vec2 = c(1L, 2L, 3L, 4L),
    vec3 = c("low", "low", "high", "high"),
    vec4 = c(TRUE, FALSE, FALSE, FALSE),
    vec5 = ymd(c("2017-05-23", "1776/07/04",
                 "1983-05/31", "1908/04-01"))
  )
ex1
```

```
# A tibble: 4 x 5
   vec1  vec2 vec3  vec4  vec5
  <dbl> <int> <chr> <lgl> <date>
1  1980     1 low   TRUE  2017-05-23
2  1990     2 low   FALSE 1776-07-04
3  2000     3 high  FALSE 1983-05-31
4  2010     4 high  FALSE 1908-04-01
```

Return to Table of Contents

Welcome to the tidyverse!

The `tidyverse` is a collection of R packages that share common philosophies and are designed to work together.

Return to Table of Contents

First motivating example for today

- Inspired by the late, great Hans Rosling

Return to Table of Contents

Data Wrangling

The `gapminder` package

```r
library(gapminder)
gapminder
```

```
# A tibble: 1,704 x 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Afghanistan Asia       1957    30.3  9240934      821.
 3 Afghanistan Asia       1962    32.0 10267083      853.
 4 Afghanistan Asia       1967    34.0 11537966      836.
 5 Afghanistan Asia       1972    36.1 13079460      740.
 6 Afghanistan Asia       1977    38.4 14880372      786.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# ... with 1,694 more rows
```

Return to Table of Contents

Base R versus the `tidyverse`

- The mean life expectancy across all years for Asia

Return to Table of Contents

Base R versus the `tidyverse`

- The mean life expectancy across all years for Asia

```
# Base R
asia <- gapminder[gapminder$continent == "Asia", ]
mean(asia$lifeExp)
```

```
[1] 60.065
```

Return to Table of Contents

Base R versus the `tidyverse`

- The mean life expectancy across all years for Asia

```
# Base R
asia <- gapminder[gapminder$continent == "Asia", ]
mean(asia$lifeExp)
```

```
[1] 60.065
```

```
library(dplyr)
gapminder %>% filter(continent == "Asia") %>%
  summarize(mean_exp = mean(lifeExp))
```

```
[1] 60.065
```

Return to Table of Contents

The pipe `%>%`

Return to Table of Contents

The pipe `%>%`

- A way to chain together commands
- Can be read as "and then" when reading over code

Return to Table of Contents

The pipe `%>%`

- A way to chain together commands
- Can be read as "and then" when reading over code

```
library(dplyr)
gapminder %>% filter(continent == "Asia") %>%
  summarize(mean_exp = mean(lifeExp))
```

Return to Table of Contents

The Five Main Verbs (5MV) of data wrangling

- `filter()`
- `summarize()`
- `group_by()`
- `mutate()`
- `arrange()`

Return to Table of Contents

`filter()`

- Select a subset of the rows of a data frame.
- Arguments are "filters" that you'd like to apply.

Return to Table of Contents

`filter()`

- Select a subset of the rows of a data frame.
- Arguments are "filters" that you'd like to apply.

```
library(gapminder); library(dplyr)
gap_2007 <- gapminder %>% filter(year == 2007)
head(gap_2007, 4)
```

```
# A tibble: 4 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       2007    43.8 31889923      975.
2 Albania     Europe     2007    76.4  3600523     5937.
3 Algeria     Africa     2007    72.3 33333216     6223.
4 Angola      Africa     2007    42.7 12420476     4797.
```

- Use `==` to compare a variable to a value

Return to Table of Contents

Logical operators

- Use `|` to check for any in multiple filters being true:

Return to Table of Contents

Logical operators

- Use `|` to check for any in multiple filters being true:

```
gapminder %>%
  filter(year == 2002 | continent == "Asia") %>%
  sample_n(8)
```

Return to Table of Contents

Logical operators

- Use `|` to check for any in multiple filters being true:

```
gapminder %>%
  filter(year == 2002 | continent == "Asia") %>%
  sample_n(8)
```

```
# A tibble: 8 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Iraq        Asia       1997    58.8 20775703     3076.
2 Korea, Rep. Asia       1997    74.6 46173816    15994.
3 Bangladesh  Asia       1952    37.5 46886859      684.
4 France      Europe     2002    79.6 59925035    28926.
5 Kuwait      Asia       1957    58.0   212846   113523.
6 Iran        Asia       1962    49.3 22874000     4187.
7 Myanmar     Asia       1982    58.1 34680442      424
8 Canada      Americas   2002    79.8 31902268    33329.
```

Return to Table of Contents

Logical operators

- Use `,` to check for all of multiple filters being true:

Return to Table of Contents

Logical operators

- Use `,` to check for all of multiple filters being true:

```
gapminder %>%
  filter(year == 2002, continent == "Asia")
```

```
# A tibble: 8 x 6
  country          continent  year lifeExp        pop gdpPercap
  <fct>            <fct>     <int>   <dbl>      <int>     <dbl>
1 Afghanistan      Asia       2002    42.1   25268405      727.
2 Bahrain          Asia       2002    74.8     656397    23404.
3 Bangladesh       Asia       2002    62.0  135656790     1136.
4 Cambodia         Asia       2002    56.8   12926707      896.
5 China            Asia       2002    72.0 1280400000     3119.
6 Hong Kong, China Asia       2002    81.5    6762476    30209.
7 India            Asia       2002    62.9 1034172547     1747.
8 Indonesia        Asia       2002    68.6  211060000     2874.
```

Return to Table of Contents

Logical operators

- Use `%in%` to check for any being true (shortcut to using `|` repeatedly with `==`)

Return to Table of Contents

Logical operators

- Use `%in%` to check for any being true (shortcut to using `|` repeatedly with `==`)

```
gapminder %>%
  filter(country %in% c("Argentina", "Belgium", "Mexico"),
         year %in% c(1987, 1992))
```

Return to Table of Contents

Logical operators

- Use `%in%` to check for any being true (shortcut to using `|` repeatedly with `==`)

```
gapminder %>%
  filter(country %in% c("Argentina", "Belgium", "Mexico"),
         year %in% c(1987, 1992))
```

```
# A tibble: 6 x 6
  country   continent  year lifeExp      pop gdpPercap
  <fct>     <fct>     <int>   <dbl>    <int>     <dbl>
1 Argentina Americas   1987    70.8 31620918     9140.
2 Argentina Americas   1992    71.9 33958947     9308.
3 Belgium   Europe     1987    75.4  9870200    22526.
4 Belgium   Europe     1992    76.5 10045622    25576.
5 Mexico    Americas   1987    69.5 80122492     8688.
6 Mexico    Americas   1992    71.5 88111030     9472.
```

Return to Table of Contents

`summarize()`

- Any numerical summary that you want to apply to a column of a data frame is specified within `summarize()`.

```
stats_1997 <- gapminder %>%
  filter(year == 1997) %>%
  summarize(max_exp = max(lifeExp),
            sd_exp = sd(lifeExp))
stats_1997
```

Return to Table of Contents

`summarize()`

- Any numerical summary that you want to apply to a column of a data frame is specified within `summarize()`.

```
stats_1997 <- gapminder %>%
  filter(year == 1997) %>%
  summarize(max_exp = max(lifeExp),
            sd_exp = sd(lifeExp))
stats_1997
```

```
# A tibble: 1 x 2
  max_exp sd_exp
    <dbl>  <dbl>
1    80.7   11.6
```

Return to Table of Contents

Combining `summarize()` with `group_by()`

When you'd like to determine a numerical summary for all levels of a different categorical variable

```
max_exp_1997_by_cont <- gapminder %>%
  filter(year == 1997) %>%
  group_by(continent) %>%
  summarize(max_exp = max(lifeExp),
            sd_exp = sd(lifeExp))
max_exp_1997_by_cont
```

Return to Table of Contents

Combining `summarize()` with `group_by()`

When you'd like to determine a numerical summary for all levels of a different categorical variable

```
max_exp_1997_by_cont <- gapminder %>%
  filter(year == 1997) %>%
  group_by(continent) %>%
  summarize(max_exp = max(lifeExp),
            sd_exp = sd(lifeExp))
max_exp_1997_by_cont
```

```
# A tibble: 5 x 3
  continent max_exp sd_exp
  <fct>       <dbl>  <dbl>
1 Africa       74.8  9.10
2 Americas     78.6  4.89
3 Asia         80.7  8.09
4 Europe       79.4  3.10
5 Oceania      78.8  0.905
```

Return to Table of Contents

`mutate()`

- Allows you to
  1. create a new variable with a specific value OR
  2. create a new variable based on other variables OR
  3. change the contents of an existing variable

Return to Table of Contents

`mutate()`

- Allows you to
  1. create a new variable with a specific value OR
  2. create a new variable based on other variables OR
  3. change the contents of an existing variable

```
gap_plus <- gapminder %>% mutate(just_one = 1)
head(gap_plus, 4)
```

```
# A tibble: 4 x 7
  country     continent  year lifeExp      pop gdpPercap just_one
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>    <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.        1
2 Afghanistan Asia       1957    30.3  9240934      821.        1
3 Afghanistan Asia       1962    32.0 10267083      853.        1
4 Afghanistan Asia       1967    34.0 11537966      836.        1
```

Return to Table of Contents

`mutate()`

- Allows you to
  1. create a new variable with a specific value OR
  2. create a new variable based on other variables OR
  3. change the contents of an existing variable

Return to Table of Contents

`mutate()`

- Allows you to
  1. create a new variable with a specific value OR
  2. create a new variable based on other variables OR
  3. change the contents of an existing variable

```
gap_w_gdp <- gapminder %>% mutate(gdp = pop * gdpPercap)
sample_n(gap_w_gdp, 4)
```

```
# A tibble: 4 x 7
  country   continent  year lifeExp      pop gdpPercap           gdp
  <fct>     <fct>     <int>   <dbl>    <int>     <dbl>         <dbl>
1 Comoros   Africa     1967    46.5   217378     1876.    407807572.
2 Paraguay  Americas   1992    68.2  4483945     4196.  18816476471.
3 Mauritius Africa     1967    61.6   789309     2475.   1953845681.
4 Chile     Americas   1997    75.8 14599929    10118. 147722858046.
```

Return to Table of Contents

`mutate()`

- Allows you to
  1. create a new variable with a specific value OR
  2. create a new variable based on other variables OR
  3. change the contents of an existing variable

Return to Table of Contents

`mutate()`

- Allows you to
  1. create a new variable with a specific value OR
  2. create a new variable based on other variables OR
  3. change the contents of an existing variable

```
gap_weird <- gapminder %>% mutate(pop = pop + 1000)
head(gap_weird, 4)
```

```
# A tibble: 4 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <dbl>     <dbl>
1 Afghanistan Asia       1952    28.8  8426333      779.
2 Afghanistan Asia       1957    30.3  9241934      821.
3 Afghanistan Asia       1962    32.0 10268083      853.
4 Afghanistan Asia       1967    34.0 11538966      836.
```

Return to Table of Contents

`arrange()`

- Reorders the rows in a data frame based on the values of one or more variables

Return to Table of Contents

`arrange()`

- Reorders the rows in a data frame based on the values of one or more variables

```
gapminder %>% arrange(year, country) %>% head(10)
```

```
# A tibble: 10 x 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Albania     Europe     1952    55.2  1282697     1601.
 3 Algeria     Africa     1952    43.1  9279525     2449.
 4 Angola      Africa     1952    30.0  4232095     3521.
 5 Argentina   Americas   1952    62.5 17876956     5911.
 6 Australia   Oceania    1952    69.1  8691212    10040.
 7 Austria     Europe     1952    66.8  6927772     6137.
 8 Bahrain     Asia       1952    50.9   120447     9867.
 9 Bangladesh  Asia       1952    37.5 46886859      684.
10 Belgium     Europe     1952    68    8730405     8343.
```

Return to Table of Contents

`arrange()`

- Can also put into descending order

Return to Table of Contents

arrange()

- Can also put into descending order

```
gapminder %>%
  filter(year > 2000) %>%
  arrange(desc(lifeExp)) %>%
  head(10)
```

```
# A tibble: 10 x 6
   country          continent  year lifeExp       pop gdpPercap
   <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
 1 Japan            Asia       2007    82.6 127467972    31656.
 2 Hong Kong, China Asia       2007    82.2   6980412    39725.
 3 Japan            Asia       2002    82   127065841    28605.
 4 Iceland          Europe     2007    81.8    301931    36181.
 5 Switzerland      Europe     2007    81.7   7554661    37506.
 6 Hong Kong, China Asia       2002    81.5   6762476    30209.
 7 Australia        Oceania    2007    81.2  20434176    34435.
 8 Spain            Europe     2007    80.9  40448191    28821.
 9 Sweden           Europe     2007    80.9   9031088    33860.
10 Israel           Asia       2007    80.7   6426679    25523.
```

Return to Table of Contents

Don't mix up `arrange` and `group_by`

- `group_by` is used (mostly) with `summarize` to calculate summaries over groups
- `arrange` is used for sorting

Return to Table of Contents

Don't mix up `arrange` and `group_by`

This doesn't really do anything useful

```
gapminder %>% group_by(year)
```

```
# A tibble: 1,704 x 6
# Groups:   year [12]
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Afghanistan Asia       1957    30.3  9240934      821.
 3 Afghanistan Asia       1962    32.0 10267083      853.
 4 Afghanistan Asia       1967    34.0 11537966      836.
 5 Afghanistan Asia       1972    36.1 13079460      740.
 6 Afghanistan Asia       1977    38.4 14880372      786.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# ... with 1,694 more rows
```

Return to Table of Contents

Don't mix up `arrange` and `group_by`

But this does

```
gapminder %>% arrange(year)
```

```
# A tibble: 1,704 x 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Albania     Europe     1952    55.2  1282697     1601.
 3 Algeria     Africa     1952    43.1  9279525     2449.
 4 Angola      Africa     1952    30.0  4232095     3521.
 5 Argentina   Americas   1952    62.5 17876956     5911.
 6 Australia   Oceania    1952    69.1  8691212    10040.
 7 Austria     Europe     1952    66.8  6927772     6137.
 8 Bahrain     Asia       1952    50.9   120447     9867.
 9 Bangladesh  Asia       1952    37.5 46886859      684.
10 Belgium     Europe     1952    68    8730405     8343.
# ... with 1,694 more rows
```

Return to Table of Contents

Practice

Use the 5MV to answer problems from R data packages, e.g., [`nycflights13::weather`]

1. What is the maximum arrival delay for each carrier departing JFK? [`nycflights13::flights`]
2. Which carrier has the worst delays (mean arrival delay)?
3. What airline corresponds to the "F9" carrier code? Use [`nycflights13::airlines`]

Return to Table of Contents

Challenge

What is the following code doing?

```
flights %>%
  select(year:day, ends_with("delay")) %>%
  group_by(year, month, day) %>%
  filter(rank(desc(arr_delay)) < 10)
```

Return to Table of Contents

Data Visualization

- What are the variables here?
- What is the observational unit?
- How are the variables mapped to aesthetics?

Return to Table of Contents

Grammar of Graphics

Wilkinson (2005) laid out the proposed "Grammar of Graphics"

Return to Table of Contents

Grammar of Graphics in R

Wickham implemented the grammar in R
in the `ggplot2` package

Return to Table of Contents

Edward Tufte pioneered the field of data visualisation

Return to Table of Contents

Return to Table of Contents

Another recommendation:

Online preview for FUNDAMENTALS OF DATA VISUALISATION
Claus O. Wilke
https://serialmentor.com/dataviz/

Return to Table of Contents

What is a statistical graphic?

Return to Table of Contents

What is a statistical graphic?

A `mapping` of
`data` variables

Return to Table of Contents

What is a statistical graphic?

A `mapping` of
`data` variables

to
`aes()` thetic attributes

Return to Table of Contents

What is a statistical graphic?

A `mapping` of
`data` variables

to
`aes()`thetic attributes

of
`geom_`etric objects.

Return to Table of Contents

Back to Basics

Return to Table of Contents

Old school

- Sketch the graphics below on paper, where the `x`-axis is variable `A` and the `y`-axis is variable `B`

```
# A tibble: 4 x 4
      A     B     C D
  <dbl> <dbl> <dbl> <chr>
1  1980     1     3 cold
2  1990     2     2 cold
3  2000     3     1 hot
4  2010     4     2 hot
```

1. A scatter plot
2. A scatter plot where the `color` of the points corresponds to `D`
3. A scatter plot where the `size` of the points corresponds to `C`

Return to Table of Contents

Reproducing the plots in `ggplot2`

1. A scatterplot

```
library(ggplot2)
ggplot(data = simple_ex, mapping = aes(x = A, y = B)) +
  geom_point()
```

Return to Table of Contents

Reproducing the plots in `ggplot2`

1. A scatterplot

```
library(ggplot2)
ggplot(data = simple_ex, mapping = aes(x = A, y = B)) +
  geom_point()
```

Return to Table of Contents

Reproducing the plots in `ggplot2`

2. A scatter plot where the `color` of the points corresponds to `D`

```
library(ggplot2)
ggplot(data = simple_ex, mapping = aes(x = A, y = B)) +
  geom_point(mapping = aes(color = D))
```

Return to Table of Contents

Reproducing the plots in `ggplot2`

2. A scatter plot where the `color` of the points corresponds to `D`

```
library(ggplot2)
ggplot(data = simple_ex, mapping = aes(x = A, y = B)) +
  geom_point(mapping = aes(color = D))
```

Return to Table of Contents

Reproducing the plots in `ggplot2`

3. A scatter plot where the `size` of the points corresponds to `C`

```
library(ggplot2)
ggplot(data = simple_ex, mapping = aes(x = A, y = B, size = C)) +
  geom_point()
```

Return to Table of Contents

Reproducing the plots in `ggplot2`

3. A scatter plot where the `size` of the points corresponds to `C`

```
library(ggplot2)
ggplot(data = simple_ex, mapping = aes(x = A, y = B, size = C)) +
  geom_point()
```

Return to Table of Contents

Another example

Do cars with bigger engines have lower fuel efficiency?

```
mpg
```

```
# A tibble: 234 x 11
   manufacturer model displ  year   cyl trans drv     cty   hwy fl
   <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr>
 1 audi         a4      1.8  1999     4 auto… f        18    29 p
 2 audi         a4      1.8  1999     4 manu… f        21    29 p
 3 audi         a4      2    2008     4 manu… f        20    31 p
 4 audi         a4      2    2008     4 auto… f        21    30 p
 5 audi         a4      2.8  1999     6 auto… f        16    26 p
 6 audi         a4      2.8  1999     6 manu… f        18    26 p
 7 audi         a4      3.1  2008     6 auto… f        18    27 p
 8 audi         a4 q…   1.8  1999     4 manu… 4        18    26 p
 9 audi         a4 q…   1.8  1999     4 auto… 4        16    25 p
10 audi         a4 q…   2    2008     4 manu… 4        20    28 p
# ... with 224 more rows, and 1 more variable: class <chr>
```

Return to Table of Contents

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

Return to Table of Contents

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

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Return to Table of Contents

Exercises

1. Run `ggplot(data = mpg)`. What do you see?
2. How many rows are in mpg? How many columns?
3. What does the `drv` variable describe? Read the help for `?mpg` to find out.
4. Make a scatterplot of hwy vs cyl.
5. What happens if you make a scatterplot of class vs drv? Why is the plot not useful?

Return to Table of Contents

Aesthetic mapping

- An aesthetic is a visual property of the objects in your plot.
- Aesthetics include things like the size, the shape, or the color of your points.

Return to Table of Contents

Mapping to colour

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

Return to Table of Contents

Mapping to shape

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

Return to Table of Contents

Mapping to size

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

Return to Table of Contents

Facets

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

Return to Table of Contents

Geometric objects

A geom is the geometrical object that a plot uses to represent data.

Return to Table of Contents

Same data, different `geom`s

Exercise: try to create each of these plots
(hint: use `geom_smooth` for plot on the right)

Return to Table of Contents

Can you create this one?

Return to Table of Contents

Can you create this one?

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

Return to Table of Contents

Introducing global mapping

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

Return to Table of Contents

Introducing global mapping

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

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

Return to Table of Contents

Introducing global mapping

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

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

```
ggplot(mpg, aes(displ,hwy)) +
  geom_point() +
  geom_smooth()
```

Return to Table of Contents

Local mappings overwrite global mappings for that that layer only

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_point(data = filter(mpg, class == "2seater"), colour = "red")
```

Return to Table of Contents

The Five-Named Graphs

The 5NG of data viz

- Scatterplot: `geom_point()`
- Line graph: `geom_line()`
- Histogram: `geom_histogram()`
- Boxplot: `geom_boxplot()`
- Bar graph: `geom_bar()`

Return to Table of Contents

More examples

Return to Table of Contents

Histogram

```
library(nycflights13)
ggplot(data = weather, mapping = aes(x = humid)) +
  geom_histogram(bins = 20, color = "black", fill = "darkorange")
```

Return to Table of Contents

Boxplot (broken)

```
library(nycflights13)
ggplot(data = weather, mapping = aes(x = month, y = humid)) +
  geom_boxplot()
```

Return to Table of Contents

Boxplot (fixed)

```
library(nycflights13)
ggplot(data = weather, mapping = aes(x = factor(month), y = humid)) +
  geom_boxplot()
```

Return to Table of Contents

Bar graph

```
library(fivethirtyeight)
ggplot(data = bechdel, mapping = aes(x = clean_test)) +
  geom_bar()
```

Return to Table of Contents

How about over time?

- One more variable to create with `dplyr`

```
library(dplyr)
year_bins <- c("'70-'74", "'75-'79", "'80-'84", "'85-'89",
               "'90-'94", "'95-'99", "'00-'04", "'05-'09",
               "'10-'13")
bechdel <- bechdel %>%
  mutate(five_year = cut(year,
                         breaks = seq(1969, 2014, 5),
                         labels = year_bins))
```

Return to Table of Contents

How about over time? (Stacked)

```
library(fivethirtyeight)
library(ggplot2)
ggplot(data = bechdel,
       mapping = aes(x = five_year, fill = clean_test)) +
  geom_bar() +
  scale_fill_brewer(type = "qual") # set colors
```

Return to Table of Contents

How about over time? (Side-by-side)

```
library(fivethirtyeight)
library(ggplot2)
ggplot(data = bechdel,
       mapping = aes(x = five_year, fill = clean_test)) +
  geom_bar(position = "dodge") +
  scale_fill_brewer(type = "qual")
```

Return to Table of Contents

How about over time? (Stacked proportional)

```
library(fivethirtyeight)
library(ggplot2)
ggplot(data = bechdel,
       mapping = aes(x = five_year, fill = clean_test)) +
  geom_bar(position = "fill", color = "black") +
  scale_fill_brewer(type = "qual")
```

Return to Table of Contents

Practice

Produce appropriate 5NG with R package & data set in [ ], e.g., [`nycflights13::weather`]

1. How does `age` predict `recline_rude`?
   [`fivethirtyeight::flying`]
2. Distribution of log base 10 scale of `budget_2013`
   [`fivethirtyeight::bechdel`]
3. How does `budget` predict `rating`?
   [`ggplot2movies::movies`]

Return to Table of Contents

HINTS

Return to Table of Contents

Determining the appropriate plot

Return to Table of Contents

Some good resources for data visualisation

- FUNDAMENTALS OF DATA VISUALISATION Claus O. Wilke
  - https://serialmentor.com/dataviz/
- awesome-python
  - https://github.com/vinta/awesome-python#data-visualization
- Flowing data
  - http://flowingdata.com/
- Tidy Tuesday
  - https://github.com/rfordatascience/tidytuesday

Return to Table of Contents

Review

- Write the code to produce this plot for 2007 data

Return to Table of Contents

```r
library(gapminder)
library(tidyverse)
gapminder_2007 <- gapminder %>% filter(year == 2007)
ggplot(data = gapminder_2007,
       mapping = aes(x = gdpPercap, y = lifeExp,
                     size = pop, color = continent)) +
  geom_point()
```

Return to Table of Contents

Data Tidying

Tidy Data?

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

The third point means we don't mix apples and oranges.

Return to Table of Contents

What is Tidy Data?

1. Each observation forms a row. In other words, each row corresponds to a single instance of an observational unit
2. Each variable forms a column:
    - Some variables may be used to identify the observational units.
    - For organizational purposes, it's generally better to put these in the left-hand columns
3. Each type of observational unit forms a table.

Return to Table of Contents

Differentiating between neat data and tidy data

- Colloquially, they mean the same thing
- But in our context, one is a subset of the other.

Neat data is

- easy to look at,
- organized nicely, and
- in table form.

Return to Table of Contents

Differentiating between neat data and tidy data

- Colloquially, they mean the same thing
- But in our context, one is a subset of the other.

Neat data is

- easy to look at,
- organized nicely, and
- in table form.

Tidy data is neat but also abides by a set of three rules.

Return to Table of Contents

Consider these representations of the same data

```
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

```
# A tibble: 12 x 4
   country      year type            count
   <chr>       <int> <chr>           <int>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       1999 cases           37737
 6 Brazil       1999 population  172006362
 7 Brazil       2000 cases           80488
 8 Brazil       2000 population  174504898
 9 China        1999 cases          212258
10 China        1999 population 1272915272
11 China        2000 cases          213766
12 China        2000 population 1280428583
```

Return to Table of Contents

What if you wanted to plot cases v. year?

```
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

```
# A tibble: 12 x 4
   country      year type            count
   <chr>       <int> <chr>           <int>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       1999 cases           37737
 6 Brazil       1999 population  172006362
 7 Brazil       2000 cases           80488
 8 Brazil       2000 population  174504898
 9 China        1999 cases          212258
10 China        1999 population 1272915272
11 China        2000 cases          213766
12 China        2000 population 1280428583
```

Return to Table of Contents

A solution

```
library(ggplot2)
ggplot(table1, aes(year, cases)) +
  geom_line(aes(group = country), colour = "grey50") +
  geom_point(aes(colour = country))
```

Return to Table of Contents

Other useful examples

Compute rate per 10,000

```
table1 %>%
  mutate(rate = cases / population * 10000)
```

```
# A tibble: 6 x 5
  country      year  cases population  rate
  <chr>       <int>  <int>      <int> <dbl>
1 Afghanistan  1999    745   19987071 0.373
2 Afghanistan  2000   2666   20595360 1.29
3 Brazil       1999  37737  172006362 2.19
4 Brazil       2000  80488  174504898 4.61
5 China        1999 212258 1272915272 1.67
6 China        2000 213766 1280428583 1.67
```

Return to Table of Contents

Other useful examples

Compute cases per year

```
table1 %>%
  count(year, wt = cases)
```

```
# A tibble: 2 x 2
   year      n
  <int>  <int>
1  1999 250740
2  2000 296920
```

Return to Table of Contents

Two common problems:

- One variable might be spread across multiple columns.
- One observation might be scattered across multiple rows.

Return to Table of Contents

Variable spread across multiple columns

A common problem is a dataset where some of the column names are not names of variables, but values of a variable.

```
# A tibble: 3 x 3
  country     `1999` `2000`
  <chr>        <int>  <int>
1 Afghanistan    745   2666
2 Brazil       37737  80488
3 China       212258 213766
```

Return to Table of Contents

Variable spread across multiple columns

A common problem is a dataset where some of the column names are not names of variables, but values of a variable.

```
# A tibble: 3 x 3
  country     `1999` `2000`
  <chr>        <int>  <int>
1 Afghanistan    745   2666
2 Brazil       37737  80488
3 China       212258 213766
```

Introducing `gather()`

Return to Table of Contents

`gather()` columns into a new pair of variables

```
# A tibble: 3 x 3
  country     `1999` `2000`
  <chr>        <int>  <int>
1 Afghanistan    745   2666
2 Brazil       37737  80488
3 China       212258 213766
```

```
table4a %>%
  gather(`1999`, `2000`, key = "year", value = "cases")
```

```
# A tibble: 6 x 3
  country     year   cases
  <chr>       <chr>  <int>
1 Afghanistan 1999     745
2 Brazil      1999   37737
3 China       1999  212258
4 Afghanistan 2000    2666
5 Brazil      2000   80488
6 China       2000  213766
```

Return to Table of Contents

What happened

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Return to Table of Contents

An observation is scattered across multiple rows

```
# A tibble: 12 x 4
   country      year type            count
   <chr>       <int> <chr>           <int>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       1999 cases           37737
 6 Brazil       1999 population  172006362
 7 Brazil       2000 cases           80488
 8 Brazil       2000 population  174504898
 9 China        1999 cases          212258
10 China        1999 population 1272915272
11 China        2000 cases          213766
12 China        2000 population 1280428583
```

Return to Table of Contents

An observation is scattered across multiple rows

```
# A tibble: 12 x 4
   country      year type            count
   <chr>       <int> <chr>           <int>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       1999 cases           37737
 6 Brazil       1999 population  172006362
 7 Brazil       2000 cases           80488
 8 Brazil       2000 population  174504898
 9 China        1999 cases          212258
10 China        1999 population 1272915272
11 China        2000 cases          213766
12 China        2000 population 1280428583
```

Introducing `spread()`

Return to Table of Contents

```
# A tibble: 12 x 4
   country      year type            count
   <chr>       <int> <chr>           <int>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       1999 cases           37737
 6 Brazil       1999 population  172006362
 7 Brazil       2000 cases           80488
 8 Brazil       2000 population  174504898
 9 China        1999 cases          212258
10 China        1999 population 1272915272
11 China        2000 cases          213766
12 China        2000 population 1280428583
```

We need two parameters:

- The column that contains variable names, the key column. Here, it's type.
- The column that contains values from multiple variables, the value column. Here it's count.

Return to Table of Contents

`spread()` columns into new variables

```
table2 %>%
    spread(key = type, value = count)
```

```
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

Return to Table of Contents

What happened?

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Return to Table of Contents

Is this tidy?

```
# A tibble: 12 x 4
    year title                      clean_test budget_2013
   <int> <chr>                      <ord>            <int>
 1  1995 Apollo 13                  ok            99370665
 2  2005 Brokeback Mountain         notalk        16583160
 3  2010 Diary of a Wimpy Kid       ok            16023478
 4  1984 Dune                       dubious      100864980
 5  1984 Ghostbusters               notalk        67243320
 6  2003 How to Lose a Guy in 10 Days men         63304348
 7  2011 Iris                       ok             5696299
 8  2004 Sideways                   ok            20964279
 9  2000 Songcatcher                ok             2435235
10  2004 Team America: World Police men           24663858
11  2010 Tron Legacy                notalk       213646368
12  2011 War Horse                  notalk        72498355
```

Return to Table of Contents

How about this? Is this tidy?

```
# A tibble: 12 x 13
   country `1952` `1957` `1962` `1967` `1972` `1977` `1982` `1987`
   <chr>    <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>
 1 Albania     -9     -9     -9     -9     -9     -9     -9     -9
 2 Argent…     -9     -1     -1     -9     -9     -9     -8      8
 3 Armenia     -9     -7     -7     -7     -7     -7     -7     -7
 4 Austra…     10     10     10     10     10     10     10     10
 5 Austria     10     10     10     10     10     10     10     10
 6 Azerba…     -9     -7     -7     -7     -7     -7     -7     -7
 7 Belarus     -9     -7     -7     -7     -7     -7     -7     -7
 8 Belgium     10     10     10     10     10     10     10     10
 9 Bhutan     -10    -10    -10    -10    -10    -10    -10    -10
10 Bolivia     -4     -3     -3     -4     -7     -7      8      9
11 Brazil       5      5      5     -9     -9     -4     -3      7
12 Bulgar…     -7     -7     -7     -7     -7     -7     -7     -7
# ... with 4 more variables: `1992` <int>, `1997` <int>,
#   `2002` <int>, `2007` <int>
```

Return to Table of Contents

Why is tidy data important?

- Think about trying to plot democracy score across years in the simplest way possible with the data on the previous slide.

Return to Table of Contents

Why is tidy data important?

- Think about trying to plot democracy score across years in the simplest way possible with the data on the previous slide.
- It would be much easier if the data looked like what follows instead so we could put
  - `year` on the `x`-axis and
  - `dem_score` on the `y`-axis.

Return to Table of Contents

Tidy is good

```
library(tidyr)
dem_score_tidy <- dem_score %>%
  gather(-country, key = "year", value = "dem_score") %>%
  mutate(year = as.integer(year))
dem_score_tidy %>% sample_n(10) %>% arrange(country)
```

Return to Table of Contents

Tidy is good

```
library(tidyr)
dem_score_tidy <- dem_score %>%
  gather(-country, key = "year", value = "dem_score") %>%
  mutate(year = as.integer(year))
dem_score_tidy %>% sample_n(10) %>% arrange(country)
```

```
# A tibble: 10 x 3
   country     year dem_score
   <chr>      <int>     <int>
 1 Canada      1997        10
 2 Costa Rica  1962        10
 3 Denmark     2002        10
 4 Georgia     1977        -7
 5 Montenegro  1972        -7
 6 Norway      2007        10
 7 Panama      1962         4
 8 Sweden      1972        10
 9 Taiwan      1952        -8
10 Taiwan      1987        -1
```

Return to Table of Contents

Let's plot it

- Plot the line graph for 4 countries using `ggplot`

```
dem_score4 <- dem_score_tidy %>%
  filter(country %in% c("Australia", "Pakistan", "Portugal", "Uruguay"))
ggplot(data = dem_score4, mapping = aes(x = year, y = dem_score)) +
  geom_line(mapping = aes(color = country), size = 2)
```

Return to Table of Contents

Beginning steps

Frequently the first thing to do when given a dataset is to

- check that the data is tidy (if not, convert it!)
- identify the observational unit,
- specify the variables, and
- give the types of variables you are presented with.

This will help with

- choosing the appropriate plot,
- summarizing the data, and
- understanding which inferences can be applied.

Return to Table of Contents

Exercise

Is this tidy? How would you make it tidy?

```
preg <- tribble(
  ~pregnant, ~male, ~female,
  "yes",     NA,    10,
  "no",      20,    12
)
preg
```

```
# A tibble: 2 x 3
  pregnant  male female
  <chr>    <dbl>  <dbl>
1 yes         NA     10
2 no          20     12
```

Return to Table of Contents

```
# A tibble: 2 x 3
  pregnant  male female
  <chr>    <dbl>  <dbl>
1 yes         NA     10
2 no          20     12
```

Return to Table of Contents

```
# A tibble: 2 x 3
  pregnant  male female
  <chr>    <dbl>  <dbl>
1 yes         NA     10
2 no          20     12
```

```
gather(preg, sex, count, male, female) %>%
  mutate(pregnant = pregnant == "yes",
         female = sex == "female") %>%
  select(-sex)
```

```
# A tibble: 4 x 3
  pregnant count female
  <lgl>    <dbl> <lgl>
1 TRUE        NA FALSE
2 FALSE       20 FALSE
3 TRUE        10 TRUE
4 FALSE       12 TRUE
```

Return to Table of Contents

Exploratory data analysis

The `ggplots::diamonds` dataset

```r
library(tidyverse)
diamonds
```

```
# A tibble: 53,940 x 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
 7 0.24  Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8 0.26  Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9 0.22  Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10 0.23  Very Good H     VS1      59.4    61   338  4     4.05  2.39
# ... with 53,930 more rows
```

Return to Table of Contents

First steps

- What is the structure of the data?
- Is there a data dictionary?
- Visualise distributions
  - Categorical variables
  - Continuous variable
  - Create a dataframe `smaller` containing only those diamonds that are less than 3 carats
  - Explore distribution of `cut` using `geom_freqpoly` on the `smaller` dataframe

Return to Table of Contents

Are there any unusual patterns?

- What might explain them?
- Can you see evidence of outliers?

Return to Table of Contents

Exercise

- Explore the distributions of the variables `x`, `y` and `z`.
- Do you see anything unusual?

Return to Table of Contents

Covariation

- Distributions describe the variation within a variable
- Covariation describes the variation between variables
- Try creating boxplots for `price` v. `cut`
- Try for `hwy` mileage v. vehicle `class` in the `mpg` dataset

Return to Table of Contents

`price` versus `cut`

Return to Table of Contents

`price` versus `cut`

```
ggplot(diamonds, aes(cut, price)) +
  geom_boxplot()
```

Return to Table of Contents

`hwy` versus `class`

```
ggplot(mpg, aes(class, hwy)) +
  geom_boxplot()
```

Return to Table of Contents

`hwy` versus `class`

```
ggplot(mpg, aes(class, hwy)) +
  geom_boxplot() +
  coord_flip()
```

Return to Table of Contents

The previous slides explored covariation between a categorical variable and a continuous variable.

- What if you have two categorical variables?
- For example, `color` and `cut`?

Return to Table of Contents

geom_tile

```
diamonds %>%
  count(color, cut) %>%
  ggplot(aes(color, cut)) +
  geom_tile(mapping = aes(fill = n))
```

Return to Table of Contents

What if you have two continuous variables?

Return to Table of Contents

Thank you!

DataCamp ModernDive infer

Thanks for attending! Contact me: Email or Twitter

- Special thanks to
  - Albert Y. Kim
  - Andrew Bray
  - Alison Hill
- Slides created via the R package xaringan by Yihui Xie
- Slides' source code at https://github.com/ismayc/talks/