Week-9: Exploratory Data Analyses

NM2207: Computational Media Literacy

Narayani Vedam, Ph.D.

Department of Communications and New Media

NUS National University of Singapore | Faculty of Arts & Social Sciences

This week

This week

I. `Tidy` data (click here)

II. `Tidy`-ing data (click here)

III. Scraping data (click here)

IV. Access & Collect data using APIs in R (click here)

So far

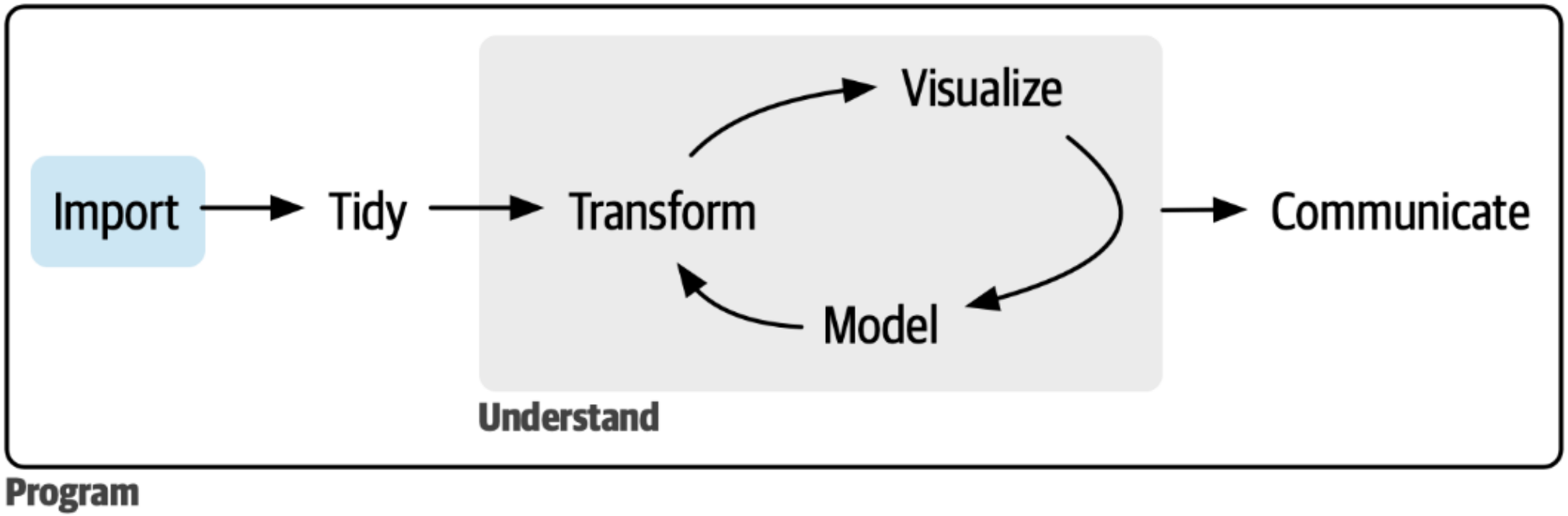


Figure: Pictorial representation of what we have set-out to do in NM2207 (Source:https://r4ds.hadley.nz/import)

To import a datase in a format different from . *csv* look-up https://r4ds.hadley.nz/data-import

`Tidy` data

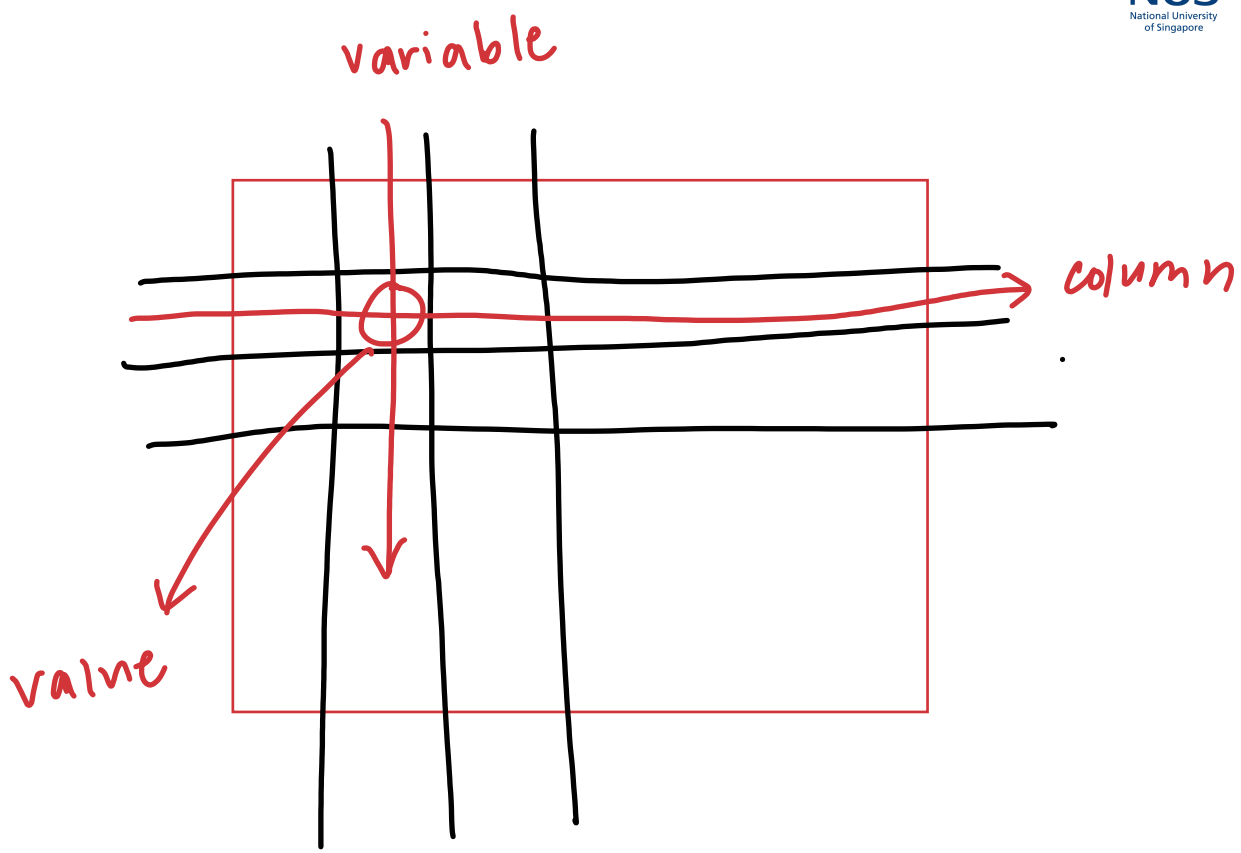
`Tidy` data

> There are three rules that mark a `Tidy` datatset

a. Each **variable** is a column; each column is a variable.

b. Each **observation** is a row; each row is an observation.

c. Each **value** is a cell; each cell is a single value.



`Tidy` data

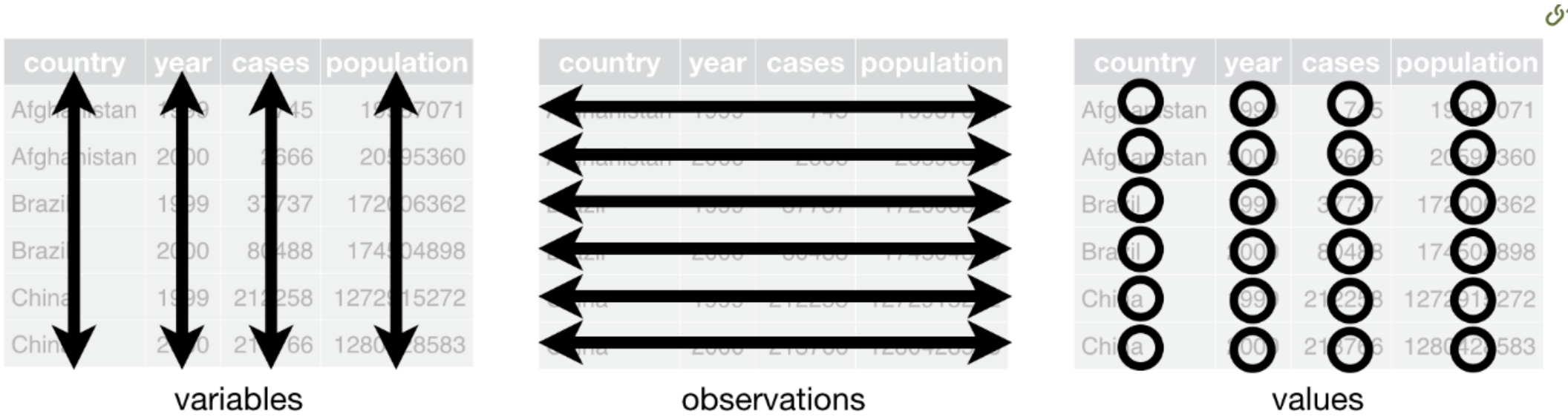


Figure: Arrangement in a Tidy dataset (Source: https://r4ds.hadley.nz/data-tidy#fig-tidy-structure)

Tidy vs Non-Tidy

```
library(tidyverse)

tidydata <- tribble(
~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)

tidydata
```

```
## # A tibble: 6 × 4
##   country      year  cases population
##   <chr>       <dbl>  <dbl>      <dbl>
## 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
```

```
nontidydata <- tribble(
 ~country,~year,~rate,
"Afghanistan",  1999, "745/19987071",
"Afghanistan",  2000, "2666/20595360",
"Brazil",       1999, "37737/172006362",
"Brazil",       2000, "80488/174504898",
"China",        1999, "212258/1272915272",
"China",        2000, "213766/1280428583")

nontidydata
```

```
## # A tibble: 6 × 3
##   country      year rate
##   <chr>       <dbl> <chr>
## 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
```

Why `Tidy` data?

1. If you have a consistent data structure, it's easier to learn the tools that work with it because they have an underlying uniformity.
2. There's a specific advantage to placing variables in columns because it allows R's vectorized nature to shine.

```
tidydata
#> # A tibble: 6 × 4
#>   country      year  cases population
#>   <chr>       <dbl>  <dbl>      <dbl>
#> 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
```

```
tidydata %>%
  group_by(year) %>%
  summarize(total_cases = sum(cases))
```

```
## # A tibble: 2 × 2
##    year total_cases
##   <dbl>       <dbl>
## 1  1999      250740
## 2  2000      296920
```

`Tidy`-ing data

`Tidy`-ing data: Example-1

```
nontidydata
```

```
## # A tibble: 6 × 3
##   country      year rate
##   <chr>       <dbl> <chr>
## 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
```

making a new subset

original dataframe

```
tidieddata <- nontidydata %>%
  separate(rate, into = c("cases",
                          "population"),
           sep = "/")

tidieddata
```

variable to change

seperator symbol

```
## # A tibble: 6 × 4
##   country      year cases  population
##   <chr>       <dbl> <chr>  <chr>
## 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
```

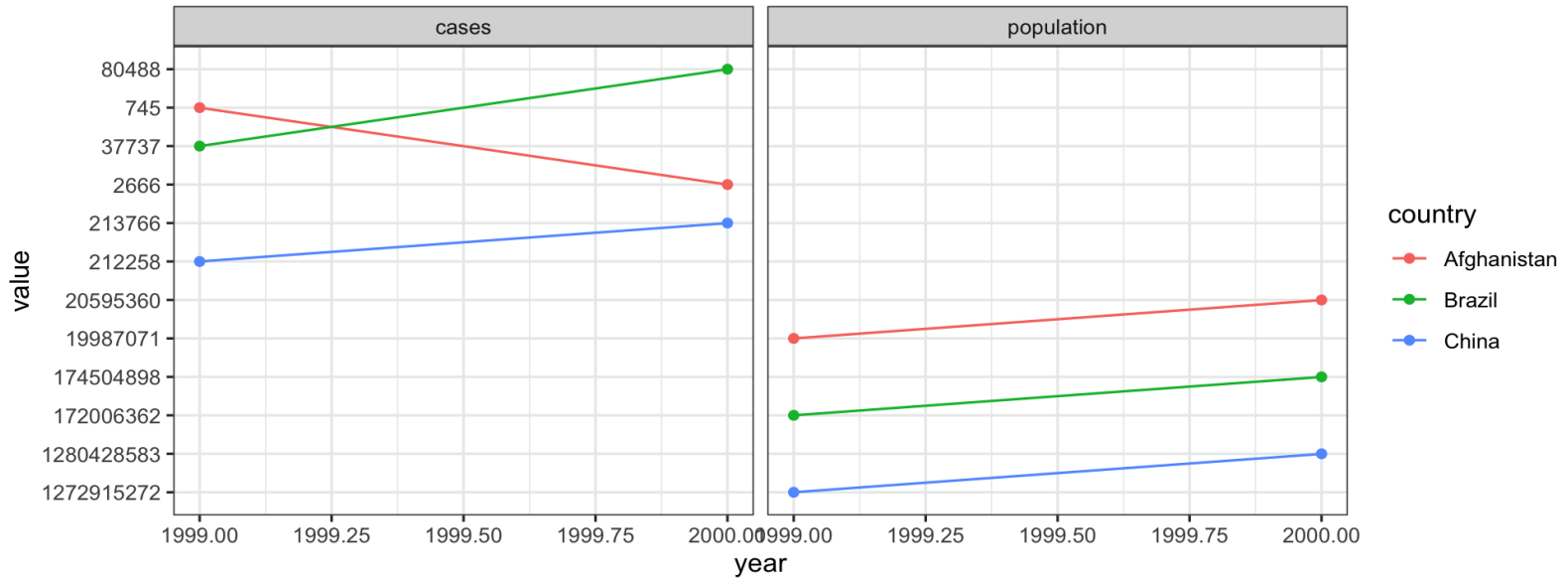
`Tidy`-ing data: Example-1

```
newtidieddata <- tidieddata %>%
  pivot_longer(
    cols = cases:population,
    names_to = "measurement",
    values_to = "value"
  )
newtidieddata
```

```
## # A tibble: 12 × 4
##    country      year measurement value
##    <chr>       <dbl> <chr>       <chr>
##  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
```

`Tidy`-ing data: Example-1

```
ggplot(newtidieddata) +
  aes(x=year,y=value, colour=country) +
  geom_point() +
  geom_line(aes(group = country))+
  facet_wrap(~measurement) +
  theme_bw()
```



`Tidy`-ing data: Example-2

```
df <- tribble(
  ~id,  ~bp1, ~bp2,
   "A",  100,  120,
   "B",  140,  115,
   "C",  120,  125
)
df
```

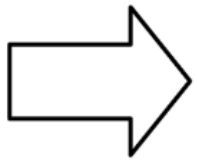
```
## # A tibble: 3 × 3
##   id      bp1   bp2
##   <chr> <dbl> <dbl>
## 1 A       100   120
## 2 B       140   115
## 3 C       120   125
```

```
df %>%
  pivot_longer(
    cols = bp1:bp2,
    names_to = "measurement",
    values_to = "value"
  )
```

```
## # A tibble: 6 × 3
##   id    measurement value
##   <chr> <chr>       <dbl>
## 1 A     bp1           100
## 2 A     bp2           120
## 3 B     bp1           140
## 4 B     bp2           115
## 5 C     bp1           120
## 6 C     bp2           125
```

Reshaping data: `pivot_longer()`

id	bp1	bp2
A	100	120
B	140	115
C	120	125

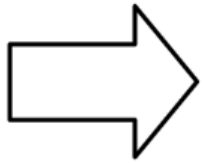


id	measurement	value
A	bp1	100
A	bp2	120
B	bp1	140
B	bp2	115
C	bp1	120
C	bp2	125

Figure: Repetition of column entries (Source:https://r4ds.hadley.nz/data-tidy#fig-tidy-structure)

Reshaping data: `pivot_longer()`

id	bp1	bp2
A	100	120
B	140	115
C	120	125

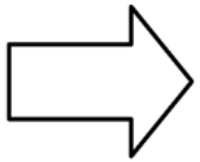


id	measurement	value
A	bp1	100
A	bp2	120
B	bp1	140
B	bp2	115
C	bp1	120
C	bp2	125

Figure: Repetition of previous column names (Source:https://r4ds.hadley.nz/data-tidy#fig-tidy-structure)

Reshaping data: `pivot_longer()`

id	bp1	bp2
A	100	120
B	140	115
C	120	125



id	measurement	value
A	bp1	100
A	bp2	120
B	bp1	140
B	bp2	115
C	bp1	120
C	bp2	125

Figure: Mapping values (Source:https://r4ds.hadley.nz/data-tidy#fig-tidy-structure)

Reshaping data: Example-3

```
newtidieddata
```

```
## # A tibble: 12 × 4
##    country      year measurement value
##    <chr>       <dbl> <chr>       <chr>
##  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
```

opposite

```
newtidieddata %>%
  pivot_wider(names_from="measurement",
              values_from="value")
```

```
## # A tibble: 6 × 4
##   country      year cases  population
##   <chr>       <dbl> <chr>  <chr>
## 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
```

Reshaping data: Example-4

```
df <- tribble(
  ~id, ~measurement, ~value,
  "A",         "bp1",    100,
  "B",         "bp1",    140,
  "B",         "bp2",    115,
  "A",         "bp2",    120,
  "A",         "bp3",    105
)
df
```

```
## # A tibble: 5 × 3
##   id    measurement value
##   <chr> <chr>       <dbl>
## 1 A     bp1           100
## 2 B     bp1           140
## 3 B     bp2           115
## 4 A     bp2           120
## 5 A     bp3           105
```

from, not to

```
df %>%
  pivot_wider(
    names_from = measurement,
    values_from = value
  )
```

```
## # A tibble: 2 × 4
##   id      bp1   bp2   bp3
##   <chr> <dbl> <dbl> <dbl>
## 1 A       100   120   105
## 2 B       140   115    NA
```

Scraping data

Scraping data from the web

- When data used in the webpage has public access
- Is not readily available for download or changing with time
- Even with public data-sets, you are not allowed to scrape personal information like phone number, email address, name, date of birth etc
- Read the ethics of scraping thoroughly before you set out to do it on your own
- Reach out to us if you plan to scrape data for your projects
- We will be scraping the webpage, https://books.toscrape.com/, which has been designed for purposes of scraping

Basics

- You will need to install the package `rvest`

```
install.packages("rvest")
```

- Load the package `rvest`

```
library(rvest)
```

- Read the html elements of the webpage

```
webpage <- read_html("https://books.toscrape.com/")
```

- Access the html elements of the webpage

```
table <-html_elements(webpage,"body")
```

Access & Collect data using APIs in R

What is an API?

> An acronym for Application Program Interface

- It is an intermediary between a dataset (usually a very large one) and us
- It facilitates our access to a dataset, usually hosted on a webpage
- Using the API to do so is also called making an API call
- To call an API, you use the web-address or the url

Calling an API

- Install packages `httr` and `jsonlite`

```
install.packages(c("httr","jsonlite"))
```

- Load packages `httr` and `jsonlite`

```
library(c(httr,jsonlite))
```

- Get the url

```
# current data
current_county_data_url <- "https://api.covidactnow.org/v2/counties.csv?apiKey=YOUR_KEY_HERE"
# historic data
historic_county_data_url <-
"https://api.covidactnow.org/v2/counties.timeseries.csv?apiKey=YOUR_KEY_HERE"
# individual location data
individual_loc_data_url <-
"https://api.covidactnow.org/v2/county/{state}.csv?apiKey=YOUR_KEY_HERE"
```

Calling an API

```
# current data
current_county_data_url <- "https://api.covidactnow.org/v2/counties.csv?apiKey=33382de96fd8441fb6c
raw_data <- GET(current_county_data_url)
raw_data$status
raw_data$content
```

Calling an API

```
# historic data
historic_county_data_url <-
"https://api.covidactnow.org/v2/counties.timeseries.csv?apiKey=33382de96fd8441fb6c1eca82b3bd4ec"
raw_data <- GET(historic_county_data_url)
raw_data$status
raw_data$content
```

Calling an API

```
# individual location data
individual_loc_data_url <-
"https://api.covidactnow.org/v2/county/{49}.csv?apiKey=33382de96fd8441fb6c1eca82b3bd4ec"
raw_data <- GET(individual_loc_data_url)
raw_data$status
raw_data$content
```

Thanks!

Slides created via the R packages:

xaringan
gadenbuie/xaringanthemer.