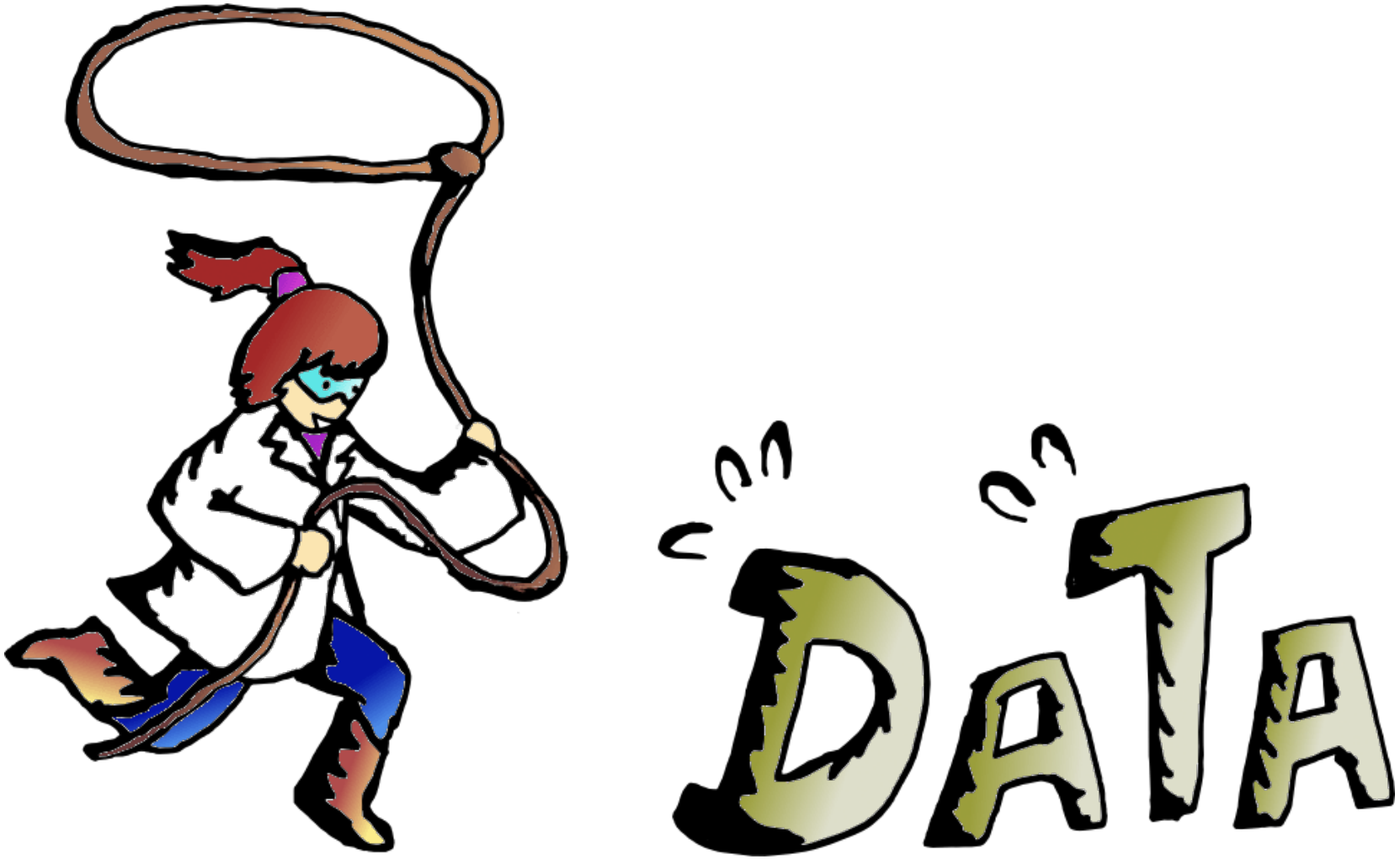
HEALTH
UNIVERSITY OF UTAH

Data Wrangling

Andrew Redd, PhD.

R Bootcamp 2020


DATA

Manipulations

- Data integrity
- Reshaping
- Filtering
- Merging
- Summarizing

Packages that we will need

```
# Make tidyverse load quietly
options(tidyverse.quiet = TRUE)
library(tidyverse)   #< General use
library(tidyr)       #< Reshaping
library(wbstats)     #< World bank data.
library(countrycode) #< Country coding
library(assertthat)  #< Results checking
library(lubridate)   #< Date manipulations
```

Loading data

for `.RData` files use `load()`

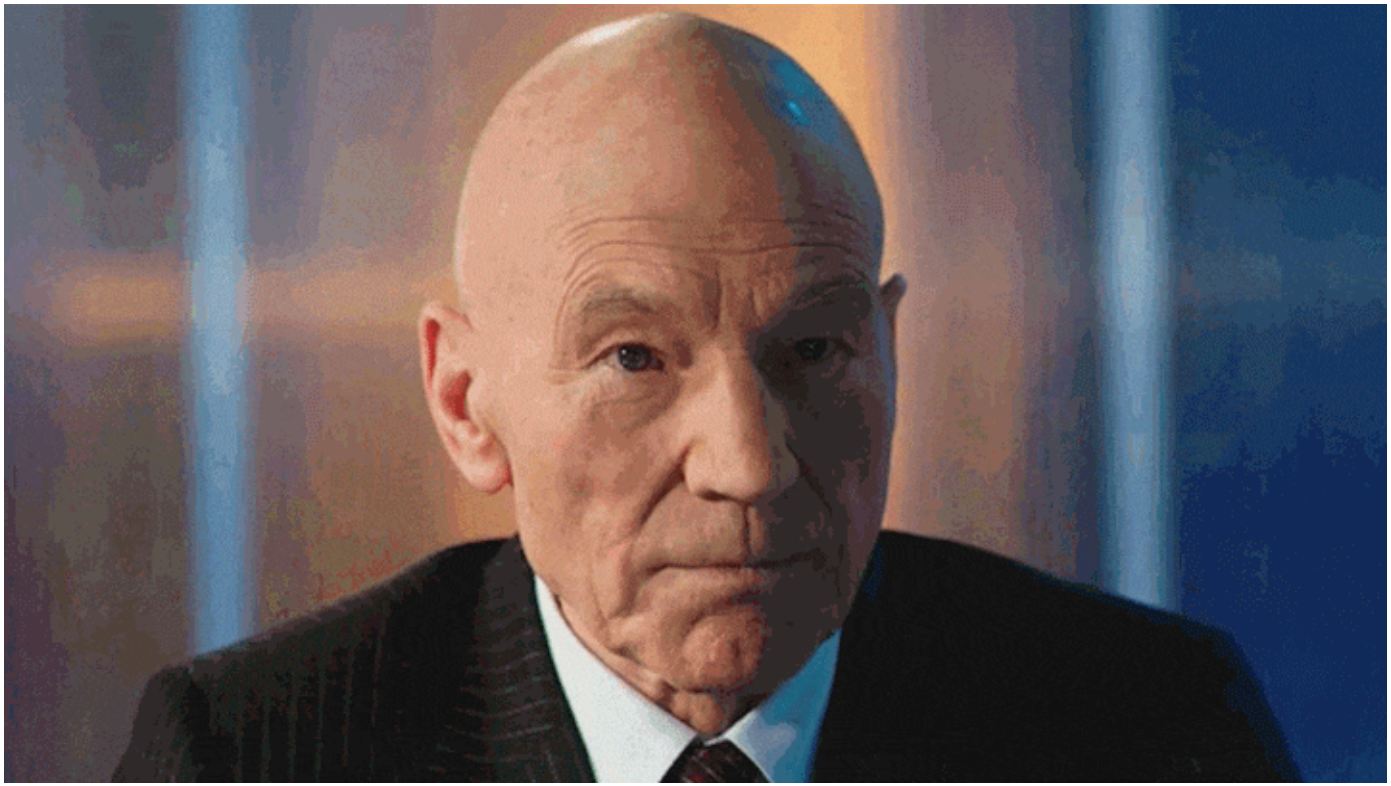
```
load("data/covid.data.wide.dfr.RData")
```

for `.rds` files use `readRDS()` and capture the results in a variable.

```
data <- readRDS("data/covid.data.wide.dfr.rds")
```

Always check your data

Any problems with the data?



1st problem: Reshaping data

- Wide Data
  - multiple observations for one unit are in columns
- Long Data
  - multiple observations for one unit are in rows.

Wide <-> Long

use `tidyr::pivot_longer()` (https://rdocumentation.org/packages/tidyr/versions/1.1.3/topics/pivot_longer) or `tidyr::pivot_wider()` (https://rdocumentation.org/packages/tidyr/versions/1.1.3/topics/pivot_wider).

```
covid.data.long.1 <-
tidyr::pivot_longer( covid.data.wide.dfr
                   , cols=matches("\\d+/\\d+/\\d+")
                   , names_to="Date"
                   , values_to="Count")
```

Time for an aside on tidy selectors.

Tidy selectors

The statement `. %>% select(file, 1:10) %>% ...` states to give back only the column named `file` then columns at indices 1 through 10.

Note `file` is not quoted. `select` and other tidy functions uses tidy selection, a form of lazy evaluation where the arguments are evaluated in the context of the data.

- `:` - sequence, i.e. all variables between given variables
- `!` - compliment of what is provided.
- `-` - drop specified variable

Tidy selector functions

- `everything()` - everything not already specified.
- `last_col()` - last variable, or nth last variable
- String matches: `starts_with()`, `ends_with()`, `contains()`(exact match), and `matches()`(regular expression match)
- `num_range()` example: X1, X2, X3, … could be selected with `num_range('X')`
- escaping the lazy evaluation
  - `all_of()` and `any_of()`, Example: `select(my.data, any_of(my.vars))` would select any variables from my.data whose names were present in the variable `my.vars`.
  - `where()`, give a predicate to make the determination if variable should be kept or not. Ex. `where(is.numeric)` would select all numeric variables.


AND NOW BACK TO
OUR REGULARLY
SCHEDULED
PROGRAMMING

Wide <-> Long

use `tidyr::pivot_longer()` (https://rdocumentation.org/packages/tidyr/versions/1.1.3/topics/pivot_longer) or `tidyr::pivot_wider()` (https://rdocumentation.org/packages/tidyr/versions/1.1.3/topics/pivot_wider).

```
covid.data.long.1 <-
tidyr::pivot_longer( covid.data.wide.dfr
                   , cols=matches("\\d+/\\d+/\\d+")
                   , names_to="Date"
                   , values_to="Count")
```

```
glimpse(covid.data.long.1)
```

```
## Rows: 359,115
## Columns: 7
## $ file             <chr> "data/confirmed.csv", "data/confi~
## $ `Province/State` <chr> NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ `Country/Region` <chr> "Afghanistan", "Afghanistan", "Af~
## $ Lat              <dbl> 33.93911, 33.93911, 33.93911, 33.~
## $ Long             <dbl> 67.70995, 67.70995, 67.70995, 67.~
## $ Date             <chr> "1/22/20", "1/23/20", "1/24/20", ~
## $ Count            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
```

2nd problem: fixing variables

Tasks:

1. Convert file names to better categories
2. Fix date to be an actual date

- Use
  - `mutate` (https://www.rdocumentation.org/packages/dplyr/versions/0.7.8/topics/mutate to add/alter variables,
  - `gsub` (https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/grep) with regular expressions (https://cheatography.com/davechild/cheat-sheets/regular-expressions/) for string manipulation,
  - and `lubridate` (https://www.rdocumentation.org/packages/lubridate/versions/1.7.10) for dates.

`mutate()` variants

- `mutate()` - modify/add variables
- `mutate_at()` - modify a set of variables.
- `mutate_if()` - modify variables meeting a criteria
- `transmute()` - create a new set of variables based on previous.

Make COVID data Long

```
covid.data.long.2 <-
    mutate( covid.data.long.1
          # Add Metric derived from file
          , Metric = file %>% basename() %>% gsub(".csv", "", ., fixed=TRUE)
          # Alter Date in place
          , Date = lubridate::mdy(Date)
          )
```

```
glimpse(covid.data.long.2)
```

```
## Rows: 359,115
## Columns: 8
## $ file             <chr> "data/confirmed.csv", "data/confi~
## $ `Province/State` <chr> NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ `Country/Region` <chr> "Afghanistan", "Afghanistan", "Af~
## $ Lat              <dbl> 33.93911, 33.93911, 33.93911, 33.~
## $ Long             <dbl> 67.70995, 67.70995, 67.70995, 67.~
## $ Date             <date> 2020-01-22, 2020-01-23, 2020-01-~
## $ Count            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Metric           <chr> "confirmed", "confirmed", "confir~
```

Now make it wider

```
covid.data.long.3 <-
    covid.data.long.2 %>%
    select(-file) %>% #< important to drop in order to get the right result
    tidyr::pivot_wider(names_from=Metric, values_from=Count)
```

```
glimpse(covid.data.long.3)
```

```
## Rows: 124,600
## Columns: 8
## $ `Province/State` <chr> NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ `Country/Region` <chr> "Afghanistan", "Afghanistan", "Af~
## $ Lat              <dbl> 33.93911, 33.93911, 33.93911, 33.~
## $ Long             <dbl> 67.70995, 67.70995, 67.70995, 67.~
## $ Date             <date> 2020-01-22, 2020-01-23, 2020-01-~
## $ confirmed        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ deaths           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ recovered        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
```

`select()` - Choosing variables

Key Function
Use `select()` to choose the variables desired.

Basic Usage

```
select(data, ...)
```

Over the next few examples we will explore the forms `...` can take

`select()` - Variable Names

the easiest is with variable names:

```
covid.data.long.2 %>%
    select(`Country/Region`, `Province/State`
          , Date, Metric, Count) %>%
    head
```

Country/Region	Province/State	Date	Metric	Count
Afghanistan	NA	2020-01-22	confirmed	0
Afghanistan	NA	2020-01-23	confirmed	0
Afghanistan	NA	2020-01-24	confirmed	0
Afghanistan	NA	2020-01-25	confirmed	0
Afghanistan	NA	2020-01-26	confirmed	0
Afghanistan	NA	2020-01-27	confirmed	0

`select()` - Dropping by Variable Names

You can select everything **but** a variable with the minus operator

```
covid.data.long.2 %>% select(-file) %>% head()
```

Province/State	Country/Region	Lat	Long	Date	Count	Metric
NA	Afghanistan	33.93911	67.70995	2020-01-22	0	confirmed
NA	Afghanistan	33.93911	67.70995	2020-01-23	0	confirmed
NA	Afghanistan	33.93911	67.70995	2020-01-24	0	confirmed
NA	Afghanistan	33.93911	67.70995	2020-01-25	0	confirmed
NA	Afghanistan	33.93911	67.70995	2020-01-26	0	confirmed
NA	Afghanistan	33.93911	67.70995	2020-01-27	0	confirmed

`select()` - By the numbers

You can select by variable position as well.

```
covid.data.long.3 %>% select(2, 3, 6:9) %>% head()
```

```
## Error: Can't subset columns that don't exist.
## x Location 9 doesn't exist.
## i There are only 8 columns.
```

`select()` - by variable range

Use single colon `:` with variable names to select variables named and eveything in between:

```
covid.data.long.3 %>%
    select(`Province/State`, `Country/Region`, Date
          , confirmed:recovered) %>% head()
```

Province/State	Country/Region	Date	confirmed	deaths	recovered
NA	Afghanistan	2020-01-22	0	0	0
NA	Afghanistan	2020-01-23	0	0	0
NA	Afghanistan	2020-01-24	0	0	0
NA	Afghanistan	2020-01-25	0	0	0
NA	Afghanistan	2020-01-26	0	0	0
NA	Afghanistan	2020-01-27	0	0	0

`select()` - by helpers

selection helpers are also provided:

```
covid.data.long.3 %>% select(contains("/")) %>% head()
```

Province/State	Country/Region
NA	Afghanistan
NA	Afghanistan
NA	Afghanistan
NA	Afghanistan
NA	Afghanistan
NA	Afghanistan

`select()` - The helpers

The available helpers are:

- `starts_with()`
- `ends_with()`
- `contains()` - must match literally
- `matches()` - Regular expression match
- `num_range()` - numerical ranged variables with a prefix
- `all_of()` - must match all of given variables named in a vector.
- `any_of()` - select any variables present in given vector, but no error if not present.
- `everything()` - Matches all variables, useful when reordering variables.
- `last_col()` - The last column

`select()` - Multiple

You may use multiple forms together.

```
covid.data.long.3 %>%
    select(2:3, where(is.Date), confirmed:recovered) %>%
    head()
```

Country/Region	Lat	Date	confirmed	deaths	recovered
Afghanistan	33.93911	2020-01-22	0	0	0
Afghanistan	33.93911	2020-01-23	0	0	0
Afghanistan	33.93911	2020-01-24	0	0	0
Afghanistan	33.93911	2020-01-25	0	0	0
Afghanistan	33.93911	2020-01-26	0	0	0
Afghanistan	33.93911	2020-01-27	0	0	0

Subsetting data

Key Function
Subset data with the `filter()` function.

The base R version is subset, but it is FAR less robust.

It takes the form of

```
filter(data, expr1, expr2, ...)
```

where `data` is the data set, and `expr1`, `expr2`, … are the criteria expressions evaluated *in the context of the data*. Data must meet *all* ctriteria to remain.

`filter()` Example

Subset data to only confirmed cases for Nigeria.

```
covid.data.long.3 %>%
    select(2:3, Date:recovered) %>%
    filter( `Country/Region` == "US") %>%
    head()
```

Country/Region	Lat	Date	confirmed	deaths	recovered
US	40	2020-01-22	1	0	0
US	40	2020-01-23	1	0	0
US	40	2020-01-24	2	0	0
US	40	2020-01-25	2	0	0
US	40	2020-01-26	5	0	0
US	40	2020-01-27	5	0	0

`filter()` Or

to perform an `or` use the single |

```
covid.data.long.3 %>%
    select(2:3, Date:recovered) %>%
    filter( (`Country/Region` == "US") | (`Country/Region` == "Canada")
          ) %>% head()
```

Country/Region	Lat	Date	confirmed	deaths	recovered
Canada	53.9333	2020-01-22	0	0	NA
Canada	53.9333	2020-01-23	0	0	NA
Canada	53.9333	2020-01-24	0	0	NA
Canada	53.9333	2020-01-25	0	0	NA
Canada	53.9333	2020-01-26	0	0	NA
Canada	53.9333	2020-01-27	0	0	NA

`filter()` and

Independent statements to filter are combined assuming an `and`. You can make an and explicit with a `&`.

an alternate form would be to use `%in%`

```
covid.data.long.3 %>%
    select(2:3, Date:recovered) %>%
    filter( `Country/Region` %in% c('US', 'Canada', 'Mexico')
          , is.na(`Province/State`)
          ) %>% head()
```

```
## Error: Problem with `filter()` input `..2`.
## i Input `..2` is `is.na(`Province/State`)`.
## x object 'Province/State' not found
```

`distinct()` - normalizing

From the previous filter example note that report date is repeated week after week.
:::{.keyfunction} To get only distinct observations, use `distinct()`. :::

distinct() - normalizing

```
covid.data.long.2 %>%
    filter( `Country/Region` %in% c('US', 'Canada', 'Mexico')
          , is.na(`Province/State`)
          ) %>%
    select(2:3, Metric) %>%
    distinct()
```

Province/State	Country/Region	Metric
NA	Mexico	confirmed
NA	US	confirmed
NA	Mexico	deaths
NA	US	deaths
NA	Canada	recovered
NA	Mexico	recovered
NA	US	recovered

Sorting Data

Key Function
To sort data use `arrange()`

`sort()` is the base version but again, less robust.

Arrange allows you to give sorting criteria.

arrange() Example

```
covid.data.long.2 %>%
    arrange(Date, `Country/Region`, `Province/State`) %>%
    select(3:2, Date, Metric, Count) %>%
    head()
```

Country/Region	Province/State	Date	Metric	Count
Afghanistan	NA	2020-01-22	confirmed	0
Afghanistan	NA	2020-01-22	deaths	0
Afghanistan	NA	2020-01-22	recovered	0
Albania	NA	2020-01-22	confirmed	0
Albania	NA	2020-01-22	deaths	0
Albania	NA	2020-01-22	recovered	0

Combining data

Key Function
Use the **`join`** family of functions to merge data together:

- `inner_join(a, b)` - keep only rows that match both `a` and `b`.
- `left_join(a, b)` - keep all rows of `a` and add columns in `b` to the rows that match. Unmatched rows will contain missing values.
- `right_join(a, b)` - same as left but swap `a` and `b`.
- `full_join(a, b)` - keep all rows of both `a` and `b`.
- `semi_join(a, b)` - keep all rows of `a` that match `b`, but don't add columns from `b`.
- `anti_join(a, b)` - keep only those rows of `a` that **don't** match `b`.

Operations have these parameters:

- `by` - variables to join on, defaults to common variables
- `suffix` - sufixes to add to distinguish common variables that are not part of `by`

World Bank Data

The `wbstats` package provides access to the world bank database.

```
library(wbstats)
wb_search('population', extra=TRUE)
```

```
(wb.pop.data <- wb_data(indicator ="SP.POP.TOTL", start_date = 2020, end_date = 2021))
```

Run these commands, investigate the output and then let's discuss.

Example: Add country information to COVID data

Which join do we want to use?

```
covid.data.long.4 <-
    left_join(covid.data.long.3, wb.pop.data
              , by=c('Country/Region'='country')
              )
```

Now we need to investigate…

Summarization

Summarization

Key Function

`summarise(data, ...)`

Take the data and summarise it by performing the `...` operations to it.

```
covid.data.long.4 %>%
    summarize( `Total` = n()
             , 'N Missing' = sum(is.na(SP.POP.TOTL))
             , 'Number of countries' = n_distinct(`Country/Region`)
             , "# of Reporting dates" = n_distinct(Date)
             , max.cases = max(confirmed, na.rm=TRUE)
             , max.deaths = max(deaths, na.rm=TRUE)
             )
```

Total	N Missing	Number of countries	# of Reporting dates	max.cases	max.deaths
124600	12460	192	445	31151495	561783

Grouped Summarization

Key Function

`group_by(data, ...)`

Take the data and group it by variables specified in `...`,
all subsequent operations should be done by group.

Grouped Summarization

```
covid.data.long.4 %>%
    group_by(`Country/Region`) %>%
    summarize( `Total` = n()
             , 'N Missing' = sum(is.na(SP.POP.TOTL))
             , 'Number of countries' = n_distinct(`Country/Region`)
             , "# of Reporting dates" = n_distinct(Date)
             , max.cases = max(confirmed, na.rm=TRUE)
             , max.deaths = max(deaths, na.rm=TRUE)
             )
```

Country/Region	Total	N Missing	Number of countries	# of Reporting dates	max.cases	max.deaths
Afghanistan	445	0	1	445	57144	2521
Albania	445	0	1	445	128155	2310
Algeria	445	0	1	445	118378	3126
Andorra	445	0	1	445	12497	120
Angola	445	0	1	445	23331	550

Exercise

Find what didn’t match?

2:00

Solution

```
covid.data.long.4 %>%
    group_by(`Country/Region`) %>%
    summarize( `Total` = n()
             , 'N Missing' = sum(is.na(SP.POP.TOTL))
             , "# of Reporting dates" = n_distinct(Date)
             , max.cases = max(confirmed, na.rm=TRUE)
             , max.deaths = max(deaths, na.rm=TRUE)
             ) %>%
    filter(`N Missing` > 0)
```

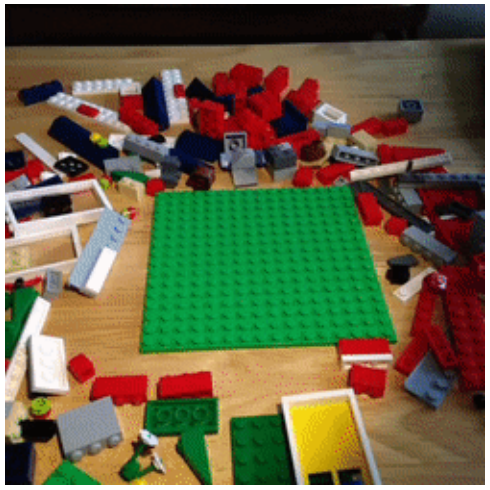
Country/Region	Total	N Missing	# of Reporting dates	max.cases	max.deaths
Bahamas	445	445	445	9364	189
Brunei	445	445	445	219	3
Burma	445	445	445	142572	3206
Congo (Brazzaville)	445	445	445	10084	137
Congo (Kinshasa)	445	445	445	28542	745

Question

What should we do with our data?

This data set on it’s own is not very interesting.

Let’s build something interesting.



Get the desired population data

- `SP.URB.TOTL.ZS` - Percentage of Population in Urban Areas (in % of Total Population)
- `SP.POP.TOTL.MA.ZS` - Population, male (% of total)
- `SP.POP.TOTL` - Population, total
- `EN.POP.DNST` - Population density (people per sq km)
- `IN.POV.HCR.EST.TOTL` - Poverty HCR Estimates (%) - Total
- `NY.GDP.PCAP.CD` - GDP per capita (current US$)

```
pop.vars <- c( 'SP.URB.TOTL.ZS', 'SP.POP.TOTL.MA.ZS'
             , 'SP.POP.TOTL', 'EN.POP.DNST'
             , 'IN.POV.HCR.EST.TOTL', 'NY.GDP.PCAP.CD')
pop.data <- wb_data(indicator = pop.vars
                  , start_date = 2020, end_date=2020)
```

Look at the data

1. What format is it in?
2. Are there any problems?
3. Did we get get everything we expected?

Join together

```
covid.data.long.final <-
    covid.data.long.3 %>%
    left_join( pop.data
             , by=c('Country/Region'='country')
             )
```

Exercise/break

15:00