# tidyverse overview

Andrew Ghazi

March 25th 2021

# While people are joining the meeting

1. Install R from cran.r-project.org
2. Install Rstudio from rstudio.com
3. Install the tidyverse with this R command: `install.packages("tidyverse")`
4. Get the slides (.html) and/or code (.R) if you'd like to follow along: github.com/andrewGhazi/tidy_overview

# What is the tidyverse?

- A dialect of R
- A suite of R packages with a shared design philosophy
- Install it with this R command:
  `install.packages('tidyverse')`

# Why use the tidyverse?

- Shared design philosophy –> consistency
- Design goal: Lower your cognitive burden
- Easy to learn, write, and read
- Other people use it

# A note

Lecture notes show up like this.

```
print('R code looks like this.')
```

```
## [1] "R code looks like this."
```

```
print('Console output has two pound symbols in front.')
```

```
## [1] "Console output has two pound symbols in front."
```

```
x = 5
```

Assignment prints nothing.

# Load the tidyverse

The startup message lists loaded packages and overwritten functions.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

# Help

To look at the documentation for any function, use `?` or `help()`

How to read help: https://socviz.co/appendix.html#a-little-more-about-r

# Basic concepts

# Data frames

- Used to store ordered collections of variables

```
diabetes
```

```
## # A tibble: 532 x 8
##    npreg   glu    bp  skin   bmi   ped   age diabetic
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
##  1     5    86    68    28  30.2 0.364    24 No
##  2     7   195    70    33  25.1 0.163    55 Yes
##  3     5    77    82    41  35.8 0.156    35 No
##  4     0   165    76    43  47.9 0.259    26 No
##  5     0   107    60    25  26.4 0.133    23 No
##  6     5    97    76    27  35.6 0.378    52 Yes
##  7     3    83    58    31  34.3 0.336    25 No
##  8     1   193    50    16  25.9 0.655    24 No
##  9     3   142    80    15  32.4 0.2      63 No
## 10     2   128    78    37  43.3 1.22     31 Yes
## # ... with 522 more rows
```

# tidy data

- Columns are variables
- Rows are observations

```
## # A tibble: 532 x 8
##    npreg   glu    bp  skin   bmi   ped   age diabetic
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
##  1     5    86    68    28  30.2 0.364    24 No
##  2     7   195    70    33  25.1 0.163    55 Yes
##  3     5    77    82    41  35.8 0.156    35 No
##  4     0   165    76    43  47.9 0.259    26 No
##  5     0   107    60    25  26.4 0.133    23 No
##  6     5    97    76    27  35.6 0.378    52 Yes
##  7     3    83    58    31  34.3 0.336    25 No
##  8     1   193    50    16  25.9 0.655    24 No
##  9     3   142    80    15  32.4 0.2      63 No
## 10     2   128    78    37  43.3 1.22     31 Yes
## # ... with 522 more rows
```

# Pipes

- Pipes look like this: `%>%`
- They take the input from the left side, and hand it to the right side:

```
c(1,2,3,4) %>% mean
```

```
## [1] 2.5
```

- Verbalize as: “and then”.
- “Set up this vector and then take the mean”.

# Pipe example

Why are pipes useful?

# Pipe example

Why are pipes useful?

# Pipe example

Why are pipes useful?

# Pipe example

Compare two ways of doing the same thing:

```
input = "potatoes"

stick(mash(boil(input)), where = 'stew')
```

```
## [1] "stewed, mashed, boiled potatoes"
```

# Pipe example

Compare two ways of doing the same thing:

```
input = "potatoes"

stick(mash(boil(input)), where = 'stew')
```

```
## [1] "stewed, mashed, boiled potatoes"
```

```
input %>%
  boil() %>%
  mash() %>%
  stick(where = 'stew')
```

```
## [1] "stewed, mashed, boiled potatoes"
```

# Pipe example

Why are pipes useful?

Because chained verbs look like English sentences.

```
input %>%
  boil() %>%
  mash() %>%
  stick(where = 'stew')
```

Piped code is easier to write *and easier to read*.

# package overview

# Import - `readr`

- Get the data into R as a data frame.
- Read data from a file on your computer or from a URL
- `read_csv()`, `read_tsv()`, `read_delim()`
- related package: `readxl`
- Read in the diabetes example dataset:

```
data_path = "data/diabetes.tsv"
diabetes = read_tsv(data_path)
```

# Diabetes dataset

- Adapted from R package `MASS` (not a tidyverse package)
- Describes incidence of diabetes in 532 Pima Indian women along with several health-related variables

```
diabetes
```

```
## # A tibble: 532 x 8
##    npreg   glu    bp  skin   bmi   ped   age diabetic
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
##  1     5    86    68    28  30.2 0.364    24 No
##  2     7   195    70    33  25.1 0.163    55 Yes
##  3     5    77    82    41  35.8 0.156    35 No
##  4     0   165    76    43  47.9 0.259    26 No
##  5     0   107    60    25  26.4 0.133    23 No
##  6     5    97    76    27  35.6 0.378    52 Yes
##  7     3    83    58    31  34.3 0.336    25 No
##  8     1   193    50    16  25.9 0.655    24 No
##  9     3   142    80    15  32.4 0.2      63 No
## 10     2   128    78    37  43.3 1.22     31 Yes
## # ... with 522 more rows
```

# Manipulate - `dplyr`

- Package for basic data manipulation
- Most important functions:
  `filter()`, `select()`, `arrange()`, `mutate()`, `group_by()`, `summarise()`

# dplyr::filter()

- Subset rows by a condition

```
diabetes %>%
  filter(npreg == 0)
```

```
## # A tibble: 77 x 8
##    npreg   glu    bp  skin   bmi   ped   age diabetic
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
##  1     0   165    76    43  47.9 0.259    26 No
##  2     0   107    60    25  26.4 0.133    23 No
##  3     0   137    40    35  43.1 2.29     33 Yes
##  4     0   139    62    17  22.1 0.207    21 No
##  5     0   101    64    17  21   0.252    21 No
##  6     0   140    65    26  42.6 0.431    24 Yes
##  7     0   121    66    30  34.3 0.203    33 Yes
##  8     0   102    86    17  29.3 0.695    27 No
##  9     0   119    66    27  38.8 0.259    22 No
## 10     0    86    68    32  35.8 0.238    25 No
## # ... with 67 more rows
```

# dplyr::filter()

- Subset rows by a condition

```
diabetes %>%
  filter(bmi > 30)
```

```
## # A tibble: 342 x 8
##    npreg   glu    bp  skin   bmi   ped   age diabetic
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
##  1     5    86    68    28  30.2 0.364    24 No
##  2     5    77    82    41  35.8 0.156    35 No
##  3     0   165    76    43  47.9 0.259    26 No
##  4     5    97    76    27  35.6 0.378    52 Yes
##  5     3    83    58    31  34.3 0.336    25 No
##  6     3   142    80    15  32.4 0.2      63 No
##  7     2   128    78    37  43.3 1.22     31 Yes
##  8     0   137    40    35  43.1 2.29     33 Yes
##  9     9   154    78    30  30.9 0.164    45 No
## 10     1   189    60    23  30.1 0.398    59 Yes
## # ... with 332 more rows
```

# dplyr::select()

- Subset columns
- Choose columns by name, index, or condition

```
diabetes %>%
  select(diabetic, npreg, age)
```

```
## # A tibble: 532 x 3
##    diabetic npreg   age
##    <chr>    <dbl> <dbl>
##  1 No           5    24
##  2 Yes          7    55
##  3 No           5    35
##  4 No           0    26
##  5 No           0    23
##  6 Yes          5    52
##  7 No           3    25
##  8 No           1    24
##  9 No           3    63
## 10 Yes          2    31
## # ... with 522 more rows
```

# dplyr::select()

- Subset columns
- Choose columns by name, index, or condition

```
diabetes %>%
  select(8, 1, 7)
```

```
## # A tibble: 532 x 3
##    diabetic npreg   age
##    <chr>    <dbl> <dbl>
##  1 No           5    24
##  2 Yes          7    55
##  3 No           5    35
##  4 No           0    26
##  5 No           0    23
##  6 Yes          5    52
##  7 No           3    25
##  8 No           1    24
##  9 No           3    63
## 10 Yes          2    31
## # ... with 522 more rows
```

# dplyr::arrange()

- Reorder rows by a variable

```
diabetes %>% arrange(age)
```

```
## # A tibble: 532 x 8
##    npreg   glu    bp  skin   bmi   ped   age diabetic
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
##  1     4    99    76    15  23.2 0.223    21 No
##  2     1   109    60     8  25.4 0.947    21 No
##  3     0   139    62    17  22.1 0.207    21 No
##  4     0   101    64    17  21   0.252    21 No
##  5     1    99    58    10  25.4 0.551    21 No
##  6     1    97    64    19  18.2 0.299    21 No
##  7     1   114    66    36  38.1 0.289    21 No
##  8     1    96    64    27  33.2 0.289    21 No
##  9     2   100    64    23  29.7 0.368    21 No
## 10     2   115    64    22  30.8 0.421    21 No
## # ... with 522 more rows
```

Quiz: What's the highest blood pressure observed in this data? (hint: desc())

# `dplyr::mutate()`

- Add new columns
- Structure:

```
input_df %>%
  mutate(col_name = col_values)
```

# dplyr::mutate()

- Add new columns
- Example: Calculate birth year as a function of age

```
diabetes %>%
  mutate(birth_year = 2021 - age)
```

```
## # A tibble: 532 x 9
##    npreg   glu    bp  skin   bmi   ped   age diabetic birth_year
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>         <dbl>
##  1     5    86    68    28  30.2 0.364    24 No             1997
##  2     7   195    70    33  25.1 0.163    55 Yes            1966
##  3     5    77    82    41  35.8 0.156    35 No             1986
##  4     0   165    76    43  47.9 0.259    26 No             1995
##  5     0   107    60    25  26.4 0.133    23 No             1998
##  6     5    97    76    27  35.6 0.378    52 Yes            1969
##  7     3    83    58    31  34.3 0.336    25 No             1996
##  8     1   193    50    16  25.9 0.655    24 No             1997
##  9     3   142    80    15  32.4 0.2      63 No             1958
## 10     2   128    78    37  43.3 1.22     31 Yes            1990
## # ... with 522 more rows
```

# dplyr::group_by()

- Group a data frame
- Example: mothers vs non-mothers:

```
diabetes %>%
  mutate(is_mother = npreg > 0) %>%
  group_by(is_mother)
```

```
## # A tibble: 532 x 9
## # Groups:   is_mother [2]
##    npreg   glu    bp  skin   bmi   ped   age diabetic is_mother
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>    <lgl>
##  1     5    86    68    28  30.2 0.364    24 No       TRUE
##  2     7   195    70    33  25.1 0.163    55 Yes      TRUE
##  3     5    77    82    41  35.8 0.156    35 No       TRUE
##  4     0   165    76    43  47.9 0.259    26 No       FALSE
##  5     0   107    60    25  26.4 0.133    23 No       FALSE
##  6     5    97    76    27  35.6 0.378    52 Yes      TRUE
##  7     3    83    58    31  34.3 0.336    25 No       TRUE
##  8     1   193    50    16  25.9 0.655    24 No       TRUE
##  9     3   142    80    15  32.4 0.2      63 No       TRUE
## 10     2   128    78    37  43.3 1.22     31 Yes      TRUE
## # ... with 522 more rows
```

# dplyr::summarise()

- Compute summary values by group

```
diabetes %>%
  mutate(is_mother = npreg > 0) %>%
  group_by(is_mother) %>%
  summarise(mean_age = mean(age),
            prop_diabetic = sum(diabetic == "Yes")/ n())
```

```
## # A tibble: 2 x 3
##   is_mother mean_age prop_diabetic
## * <lgl>        <dbl>         <dbl>
## 1 FALSE         25.7         0.351
## 2 TRUE          32.6         0.330
```

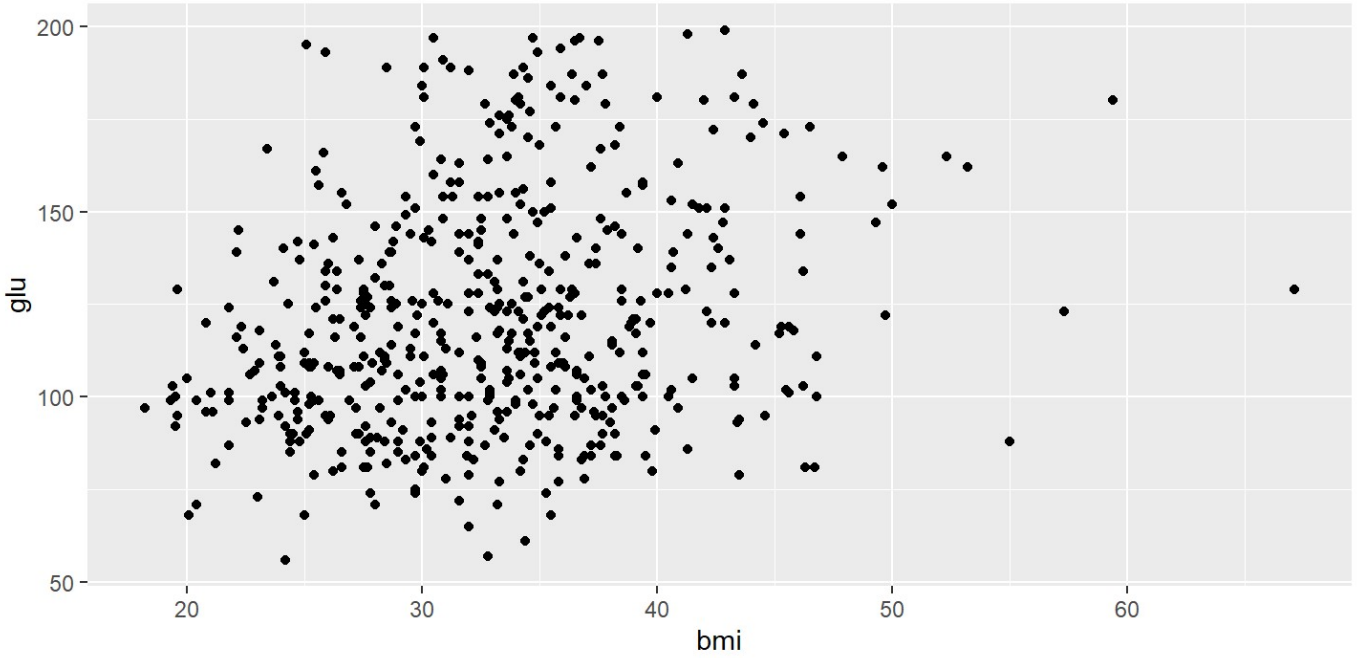
# Visualize - `ggplot2`

- “Grammar of graphics”
- Core concepts:
  - Variables from tidy input data
  - Aesthetic mappings between variables and graphical elements: `aes()`
  - Graphical elements via `geom_*()` e.g. `geom_point()`

# Visualize - `ggplot2` example

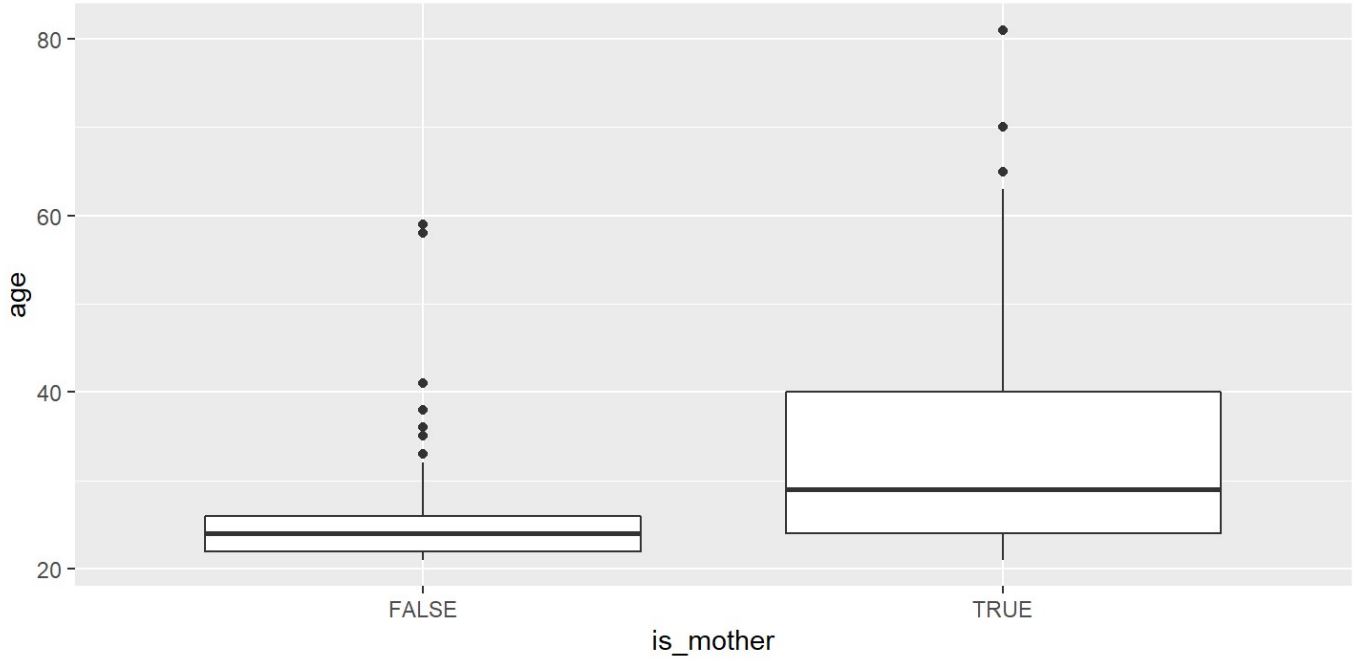
- Tidy input, aesthetic mapping, and a geom

```
diabetes %>%
  ggplot(aes(x = bmi, y = glu)) +
  geom_point()
```

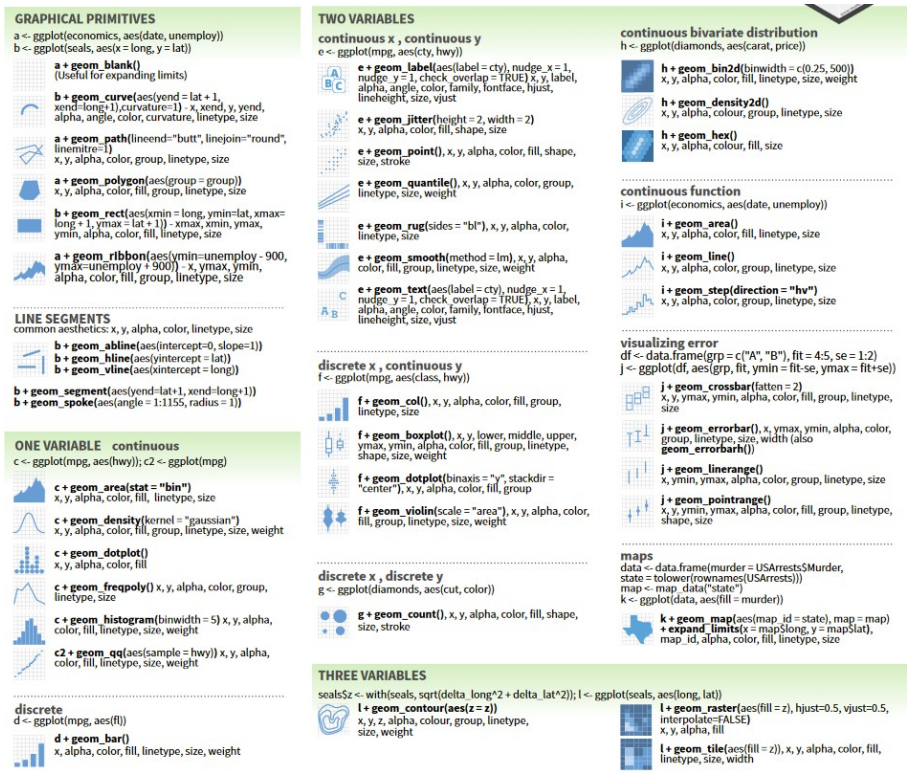


# Visualize - `ggplot2` example

```
diabetes %>%
  mutate(is_mother = npreg > 0) %>%
  ggplot(aes(x = is_mother, y = age)) +
  geom_boxplot()
```



# Visualize - ggplot2's suite of geoms



# Report - `rmarkdown`

- Easily interweave code, $\LaTeX$ and text.
- Render as reports (.html, .docx, .pdf), slides (.html, .ppt), or web applications (Shiny)
- Example: these slides
- Your PI will love you
- Demonstration at end of session 1 if there's time

# COVID data

- New York Times COVID-19 data on Github
- Provides COVID-19 data by county, state, and nation-wide
- https://github.com/nytimes/covid-19-data

# Example: US COVID-19 state-level data

```
nyt_url = "https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv"
state_covid = read_csv(nyt_url)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   date = col_date(format = ""),
##   state = col_character(),
##   fips = col_character(),
##   cases = col_double(),
##   deaths = col_double()
## )
```

# Example: US COVID-19 state-level data

```
state_covid
```

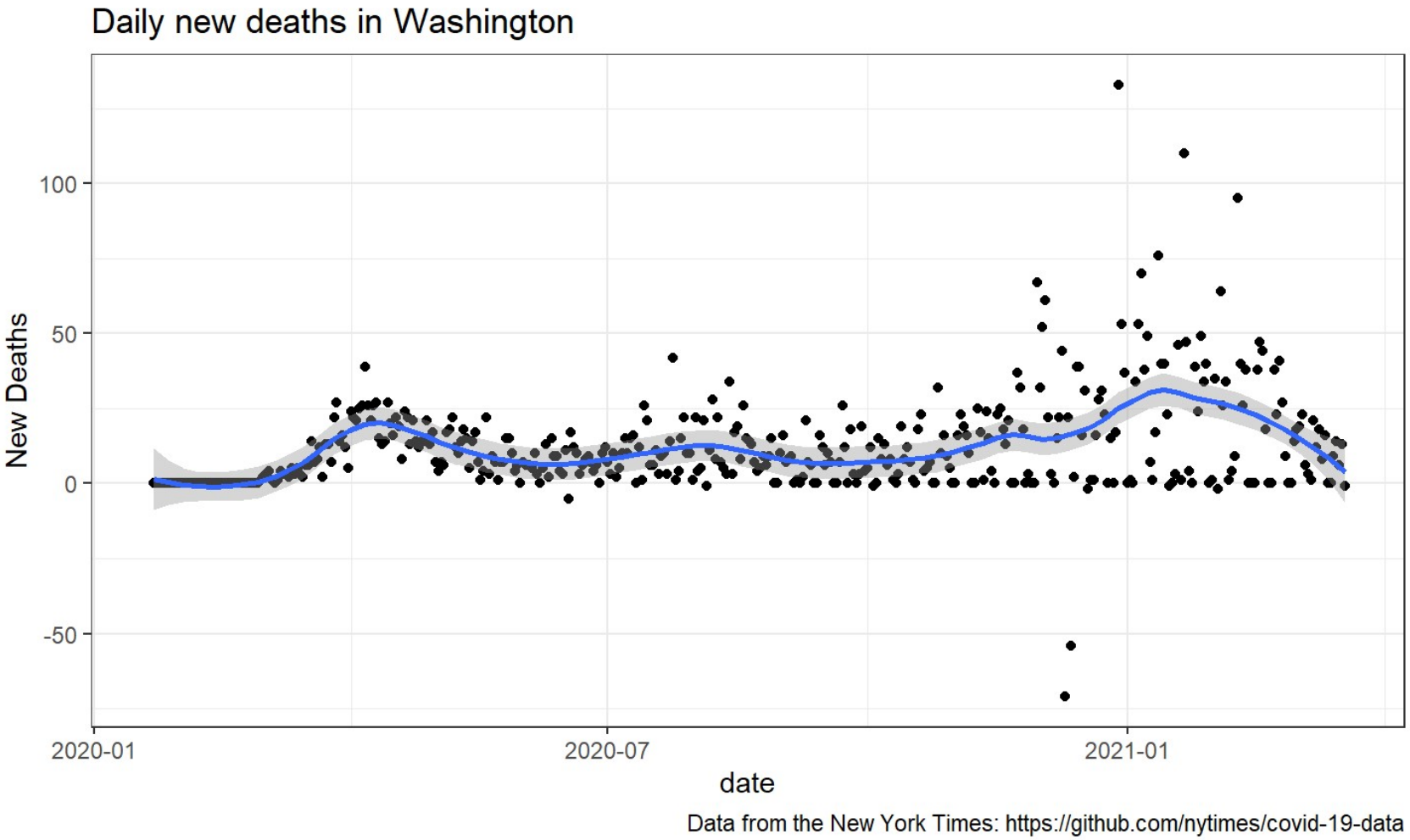
```
## # A tibble: 21,024 x 5
##    date       state      fips  cases deaths
##    <date>     <chr>      <chr> <dbl>  <dbl>
##  1 2020-01-21 Washington 53        1      0
##  2 2020-01-22 Washington 53        1      0
##  3 2020-01-23 Washington 53        1      0
##  4 2020-01-24 Illinois   17        1      0
##  5 2020-01-24 Washington 53        1      0
##  6 2020-01-25 California 06        1      0
##  7 2020-01-25 Illinois   17        1      0
##  8 2020-01-25 Washington 53        1      0
##  9 2020-01-26 Arizona    04        1      0
## 10 2020-01-26 California 06        2      0
## # ... with 21,014 more rows
```

# Example: US COVID-19

```
state_covid %>%
  filter(state == "Washington") %>%
  mutate(new_deaths = c(NA, diff(deaths))) %>%
  ggplot(aes(date, new_deaths)) +
  geom_point() +
  geom_smooth(method = "loess", span = .2) +
  labs(title = "Daily new deaths in Washington",
       y = "New Deaths",
       caption = "Data from the New York Times: https://github.com/nytimes/covid-19-data") +
  theme_bw()
```

- What will the output of this command look like?

# Example: US COVID-19



# Day 2

# purrr

- Functional programming

# map()

- Apply a function to each element of a list
- Example:

```
map(1:3, sqrt)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 1.414214
##
## [[3]]
## [1] 1.732051
```

# map()

- Apply a function to each element of a list
- Example:

```
map(1:3, sqrt)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 1.414214
##
## [[3]]
## [1] 1.732051
```

```
map(datasets, ml_algorithm)
```

# Anonymous functions

- Tilde + `.x`
- Use to easily define single-use functions

# `map_*()` variants

- `map()` always returns a list
- Variants return a vector of a specific type

```
map_dbl(1:5, ~.x^2 + 1)
```

```
## [1]  2  5 10 17 26
```

```
map_lgl(1:5, ~.x == 3)
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE
```

```
map_chr(letters[1:5], ~paste0(rep(.x, 3), collapse=''))
```

```
## [1] "aaa" "bbb" "ccc" "ddd" "eee"
```

# purrr example: run many models

- Define each combination of two explanatory variables

```
formula_df = names(diabetes)[1:7] %>%
  combn(m = 2) %>%
  t %>%
  as_tibble %>%
  set_names(c('x1', 'x2')) %>%
  mutate(formula = paste("diabetic ~ ", x1, " + ", x2, sep = ''))
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_re
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
formula_df
```

```
## # A tibble: 21 x 3
##    x1    x2    formula
##    <chr> <chr> <chr>
##  1 npreg glu   diabetic ~ npreg + glu
##  2 npreg bp    diabetic ~ npreg + bp
##  3 npreg skin  diabetic ~ npreg + skin
##  4 npreg bmi   diabetic ~ npreg + bmi
##  5 npreg ped   diabetic ~ npreg + ped
```

# purrr example: run many models

- Run a logistic regression on each combination of explanatory variables

```
formula_df %>%
  mutate(glm_result = map(formula,
                          ~glm(.x, data = diabetes, family = 'binomial')))
```

# purrr example: run many models

- Extract the AIC of each model fit

```
formula_df %>%
  mutate(glm_result = map(formula,
                          ~glm(.x, data = diabetes, family = 'binomial')),
         aic = map_dbl(lm_result,
                       ~.x$aic))
```

# purrr example: run many models

- See which model had the lowest AIC

```
formula_df %>%
  mutate(glm_result = map(formula,
                          ~glm(.x, data = diabetes, family = 'binomial')),
         aic = map_dbl(lm_result,
                       ~.x$aic)) %>%
  arrange(aic)
```

# Organize - `tidyr`

- `pivot_longer()` and `pivot_wider()`
- `separate()` and `unite()`

# Data tidying example: WHO TB data

- WHO tuberculosis case data
- realistically messy!
- Try running these commands:

```
?tidyr::who
dim(who)
names(who)
who[1:5, 1:8]
who
```

- We will tidy this data in 5 steps

# WHO TB column names

- First part: “new” or “old”
- Second part: TB type (relapse, extrapulmonary, smear-positive, smear-negative)
- Third part: Sex and age group (e.g. f2534)

# TB tidying 1/5: pivot wide to long

```
who1 = who %>%
  pivot_longer(new_sp_m014:newrel_f65,
               names_to = "key",
               values_to = "values",
               values_drop_na = TRUE)
who1
```

```
## # A tibble: 76,046 x 6
##    country     iso2  iso3   year key          values
##    <chr>       <chr> <chr> <int> <chr>         <int>
##  1 Afghanistan AF    AFG    1997 new_sp_m014       0
##  2 Afghanistan AF    AFG    1997 new_sp_m1524     10
##  3 Afghanistan AF    AFG    1997 new_sp_m2534      6
##  4 Afghanistan AF    AFG    1997 new_sp_m3544      3
##  5 Afghanistan AF    AFG    1997 new_sp_m4554      5
##  6 Afghanistan AF    AFG    1997 new_sp_m5564      2
##  7 Afghanistan AF    AFG    1997 new_sp_m65        0
##  8 Afghanistan AF    AFG    1997 new_sp_f014       5
##  9 Afghanistan AF    AFG    1997 new_sp_f1524     38
## 10 Afghanistan AF    AFG    1997 new_sp_f2534     36
## # ... with 76,036 more rows
```

# TB tidying 2/5: fix inconsistent names

```
who2 = who1 %>%
  mutate(key = str_replace(key, "newrel", "new_rel"))
who2
```

```
## # A tibble: 76,046 x 6
##    country     iso2  iso3   year key          values
##    <chr>       <chr> <chr> <int> <chr>         <int>
##  1 Afghanistan AF    AFG    1997 new_sp_m014       0
##  2 Afghanistan AF    AFG    1997 new_sp_m1524     10
##  3 Afghanistan AF    AFG    1997 new_sp_m2534      6
##  4 Afghanistan AF    AFG    1997 new_sp_m3544      3
##  5 Afghanistan AF    AFG    1997 new_sp_m4554      5
##  6 Afghanistan AF    AFG    1997 new_sp_m5564      2
##  7 Afghanistan AF    AFG    1997 new_sp_m65        0
##  8 Afghanistan AF    AFG    1997 new_sp_f014       5
##  9 Afghanistan AF    AFG    1997 new_sp_f1524     38
## 10 Afghanistan AF    AFG    1997 new_sp_f2534     36
## # ... with 76,036 more rows
```

# TB tidying 3/5: separate identifier components

```
who3 = who2 %>%
  separate(key, into = c("newold", "tb_type", "sexage"))
```

# TB tidying 4/5: separate sex and age group

```
who4 = who3 %>%
  separate(sexage, into = c("sex", "age_group"), sep = 1)
```

# TB tidying 5/5: remove redundant columns

```
who_final = who4 %>%
  select(-newold, -matches("iso"))
```

# TB tidying: Final command

```
tidyr::who %>%
  pivot_longer(new_sp_m014:newrel_f65,
               names_to = "key",
               values_to = "values",
               values_drop_na = TRUE) %>%
  mutate(key = str_replace(key, "newrel", "new_rel")) %>%
  separate(key, into = c("newold", "tb_type", "sexage")) %>%
  separate(sexage, into = c("sex", "age_group"), sep = 1) %>%
  select(-newold, -matches("iso"))
```

```
## # A tibble: 76,046 x 6
##    country      year tb_type sex   age_group values
##    <chr>       <int> <chr>   <chr> <chr>      <int>
##  1 Afghanistan  1997 sp      m     014            0
##  2 Afghanistan  1997 sp      m     1524          10
##  3 Afghanistan  1997 sp      m     2534           6
##  4 Afghanistan  1997 sp      m     3544           3
##  5 Afghanistan  1997 sp      m     4554           5
##  6 Afghanistan  1997 sp      m     5564           2
##  7 Afghanistan  1997 sp      m     65             0
##  8 Afghanistan  1997 sp      f     014            5
##  9 Afghanistan  1997 sp      f     1524          38
## 10 Afghanistan  1997 sp      f     2534          36
## # ... with 76,036 more rows
```

# Joins if there's time

# Everything else

- `forcats`
- `stringr`
- `lubridate`
- `tidymodels`

# Resources

- R for Data Science - https://r4ds.had.co.nz/
- Cheat Sheets - https://rstudio.com/resources/cheatsheets/