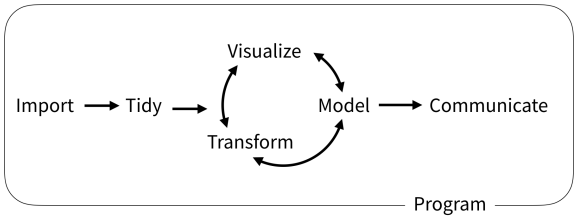
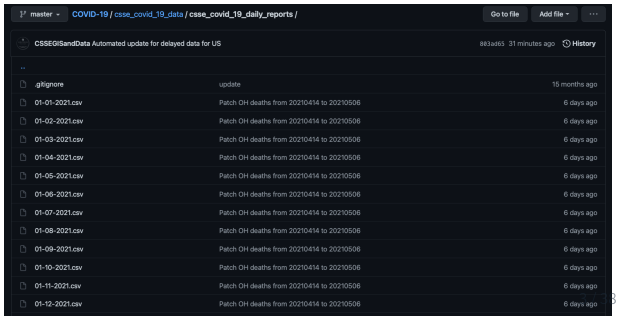
STATS 220

Don’t repeat yourself



Case study: batch download and process COVID-19 data



Keep calm, and import CSVs one by one

```
library(tidyverse) # library(purrr)
covid_0301 <- read_csv("https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/03-01-2020.csv")
covid_0302 <- read_csv("https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/03-02-2020.csv")
covid_0303 <- read_csv("https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/03-03-2020.csv")
(covid_03 <- bind_rows(covid_0301, covid_0302, covid_0303))
```

```
#> # A tibble: 420 x 8
#>   `Province/State` `Country/Region` `Last Update`
#>   <chr>            <chr>            <dttm>
#> 1 Hubei            Mainland China   2020-03-01 10:13:19
#> 2 <NA>             South Korea      2020-03-01 23:43:03
#> 3 <NA>             Italy            2020-03-01 23:23:02
#> 4 Guangdong        Mainland China   2020-03-01 14:13:18
#> 5 Henan            Mainland China   2020-03-01 14:13:18
#> 6 Zhejiang         Mainland China   2020-03-01 10:13:33
#> # … with 414 more rows, and 5 more variables:
#> #   Confirmed <dbl>, Deaths <dbl>, Recovered <dbl>,
#> #   Latitude <dbl>, Longitude <dbl>
```

Data is changing

```
covid_1231 <- read_csv("https://github.com/CSSEGISandData/COVID-
19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/12-31-2020.csv")
covid_1231
```

```
#> # A tibble: 3,980 x 14
#>    FIPS Admin2 Province_State Country_Region
#>   <dbl> <chr>  <chr>          <chr>
#> 1    NA <NA>   <NA>           Afghanistan
#> 2    NA <NA>   <NA>           Albania
#> 3    NA <NA>   <NA>           Algeria
#> 4    NA <NA>   <NA>           Andorra
#> 5    NA <NA>   <NA>           Angola
#> 6    NA <NA>   <NA>           Antigua and Barbuda
#> # … with 3,974 more rows, and 10 more variables:
#> #   Last_Update <dttm>, Lat <dbl>, Long_ <dbl>,
#> #   Confirmed <dbl>, Deaths <dbl>, Recovered <dbl>,
#> #   Active <dbl>, Combined_Key <chr>, Incident_Rate <dbl>,
#> #   Case_Fatality_Ratio <dbl>
```


What are the repeated patterns?

1. everyday different URLs
2. read each csv file into R
3. non-standardised column names

DRY: automate the common tasks above

1 generate a sequence of urls

```
make_url <- function(date) {
  date_chr <- format(date, "%m-%d-%Y")
  setNames(glue::glue("https://github.com/CSSEGISandData/COVID-
19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/{date_chr}.csv"),
  date_chr)
}
```

3 key steps to creating a custom function:

1. pick a **name** for the function, e.g. `make_url`.
2. list the inputs, or **arguments**, to the function inside `function`, e.g. `date`.
3. place the code you have developed in **body** of the function, a `{` block that immediately follows `function(...)`.

Test `make_url()`: does it return what we expected?

```
library(lubridate)
dates <- seq(mdy("03-01-2020"), mdy("12-31-2020"), by = 1)
urls <- make_url(dates)
head(urls, 2)
```

```
#> https://github.com/CSSEGISandData/COVID-
19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/03-01-2020.csv
#> https://github.com/CSSEGISandData/COVID-
19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/03-02-2020.csv
```

```
tail(urls, 2)
```

```
#> https://github.com/CSSEGISandData/COVID-
19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/12-30-2020.csv
#> https://github.com/CSSEGISandData/COVID-
19/raw/master/csse_covid_19_data/csse_covid_19_daily_reports/12-31-2020.csv
```

2 & 3 read data and standardise column names

```
read_covid19 <- function(url) {
  data <- read_csv(url,
    col_types = cols(Last_Update = "_", `Last Update` = "_"))
  rename_with(data, janitor::make_clean_names)
}
read_covid19(urls[1])
```

```
#> # A tibble: 126 x 7
#>   province_state country_region confirmed deaths recovered
#>   <chr>          <chr>              <dbl>  <dbl>     <dbl>
#> 1 Hubei          Mainland China     66907   2761     31536
#> 2 <NA>           South Korea         3736     17        30
#> 3 <NA>           Italy               1694     34        83
#> 4 Guangdong      Mainland China      1349      7      1016
#> 5 Henan          Mainland China      1272     22      1198
#> 6 Zhejiang       Mainland China      1205      1      1046
#> # … with 120 more rows, and 2 more variables:
#> #   latitude <dbl>, longitude <dbl>
```


Why function

- Modularise the code to make it easier to understand the logic.
- Generalise the code to make it reusable later.
- Reduce the duplication and make it less error-prone.

The joy of functional programming (FP)

Hadley Wickham: The joy of functional programming

comes to rescue

copy and paste

```
covid19_csv1 <- read_covid19(urls[1])
covid19_csv2 <- read_covid19(urls[2])
covid19_csv3 <- read_covid19(urls[2])
covid19_csv4 <- read_covid19(urls[4])
covid19_csv5 <- read_covid19(urls[5])
covid19_csv6 <- read_covid19(urls[6])
covid19_csv6 <- read_covid19(urls[7])
covid19_csv8 <- read_covid19(urls[8])
covid19_csv9 <- read_covid19(urls[9])
```

When it's gonna end? 😭

The goal of FP is to make it easy to express repeated actions using high-level verbs.

- `map()`

`map(.x, .f, ...)` for every element of `.x`, apply `.f`

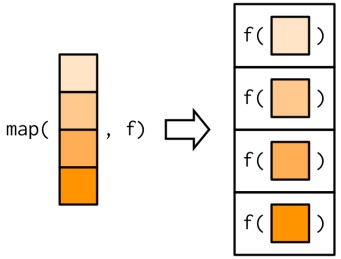
> always returns a list

```
covid19_lst <- map(urls, read_covid19)
covid19_lst[[length(covid19_lst)]]
```

```
#> # A tibble: 3,980 x 13
#>    fips admin2 province_state country_region      lat   long
#>   <dbl> <chr>  <chr>          <chr>             <dbl>  <dbl>
#> 1    NA <NA>   <NA>           Afghanistan        33.9  67.7
#> 2    NA <NA>   <NA>           Albania            41.2  20.2
#> 3    NA <NA>   <NA>           Algeria            28.0   1.66
#> 4    NA <NA>   <NA>           Andorra            42.5   1.52
#> 5    NA <NA>   <NA>           Angola            -11.2  17.9
#> 6    NA <NA>   <NA>           Antigua and Barb…  17.1 -61.8
#> # … with 3,974 more rows, and 7 more variables:
#> #   confirmed <dbl>, deaths <dbl>, recovered <dbl>,
#> #   active <dbl>, combined_key <chr>, incident_rate <dbl>,
#> #   case_fatality_ratio <dbl>
```


- `map()`

A **functional** is a function that takes a function as an input and returns a vector as output.



- `map()`

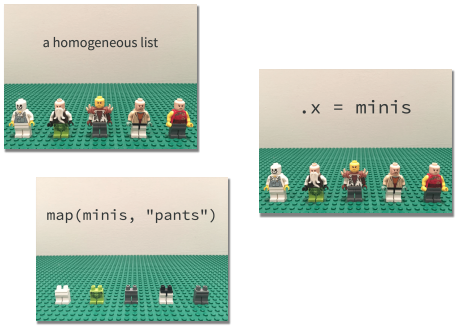


image credit: Jenny Bryan

- `map()`

Bind multiple tibbles by row

```
bind_rows(covid19_lst)
```

```
#> # A tibble: 1,069,141 x 16
#>   province_state country_region confirmed deaths recovered
#>   <chr>          <chr>              <dbl>  <dbl>     <dbl>
#> 1 Hubei          Mainland China     66907   2761     31536
#> 2 <NA>           South Korea         3736     17        30
#> 3 <NA>           Italy               1694     34        83
#> 4 Guangdong      Mainland China      1349      7      1016
#> 5 Henan          Mainland China      1272     22      1198
#> 6 Zhejiang       Mainland China      1205      1      1046
#> # … with 1,069,135 more rows, and 11 more variables:
#> #   latitude <dbl>, longitude <dbl>, fips <dbl>,
#> #   admin2 <chr>, lat <dbl>, long <dbl>, active <dbl>,
#> #   combined_key <chr>, incidence_rate <dbl>,
#> #   case_fatality_ratio <dbl>, incident_rate <dbl>
```

- map()

Hmm, where are the dates?

```
bind_rows(covid19_lst)
```

```
#> # A tibble: 1,069,141 x 16
#>   province_state country_region confirmed deaths recovered
#>   <chr>          <chr>              <dbl>  <dbl>     <dbl>
#> 1 Hubei          Mainland China     66907   2761     31536
#> 2 <NA>           South Korea         3736     17        30
#> 3 <NA>           Italy               1694     34        83
#> 4 Guangdong      Mainland China      1349      7      1016
#> 5 Henan          Mainland China      1272     22      1198
#> 6 Zhejiang       Mainland China      1205      1      1046
#> # … with 1,069,135 more rows, and 11 more variables:
#> #   latitude <dbl>, longitude <dbl>, fips <dbl>,
#> #   admin2 <chr>, lat <dbl>, long <dbl>, active <dbl>,
#> #   combined_key <chr>, incidence_rate <dbl>,
#> #   case_fatality_ratio <dbl>, incident_rate <dbl>
```

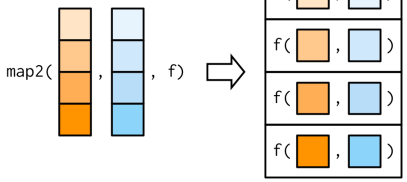

- map()
- **map2()**

`map2(.x, .y, .f)` for every element of `.x` and `.y`, apply `.f`

```
covid19_lst <- map2(urls, dates,
  ~ read_covid19(.x) %>% mutate(date = .y))
bind_rows(covid19_lst)
```

```
#> # A tibble: 1,069,141 x 17
#>   province_state country_region confirmed deaths recovered
#>   <chr>          <chr>              <dbl>  <dbl>     <dbl>
#> 1 Hubei          Mainland China     66907   2761     31536
#> 2 <NA>           South Korea         3736     17        30
#> 3 <NA>           Italy               1694     34        83
#> 4 Guangdong      Mainland China      1349      7      1016
#> 5 Henan          Mainland China      1272     22      1198
#> 6 Zhejiang       Mainland China      1205      1      1046
#> # … with 1,069,135 more rows, and 12 more variables:
#> #   latitude <dbl>, longitude <dbl>, fips <dbl>,
#> #   admin2 <chr>, lat <dbl>, long <dbl>, active <dbl>,
#> #   combined_key <chr>, incidence_rate <dbl>,
#> #   case_fatality_ratio <dbl>, incident_rate <dbl>,
#> #   date <date>
```


- map()
- **map2()**



- map()
- map2()
- **map_dfr()**

`map_*()` is a variant of `map()` that returns an output in the indicated type. `map_dfr()` returns data frames by row-binding.

```
covid19_raw <- map_dfr(urls, read_covid19, .id = "date")
covid19_raw
```

```
#> # A tibble: 1,069,141 x 17
#>   date       province_state country_region confirmed deaths
#>   <chr>      <chr>          <chr>              <dbl>  <dbl>
#> 1 03-01-2020 Hubei          Mainland China     66907   2761
#> 2 03-01-2020 <NA>           South Korea         3736     17
#> 3 03-01-2020 <NA>           Italy               1694     34
#> 4 03-01-2020 Guangdong      Mainland China      1349      7
#> 5 03-01-2020 Henan          Mainland China      1272     22
#> 6 03-01-2020 Zhejiang       Mainland China      1205      1
#> # … with 1,069,135 more rows, and 12 more variables:
#> #   recovered <dbl>, latitude <dbl>, longitude <dbl>,
#> #   fips <dbl>, admin2 <chr>, lat <dbl>, long <dbl>,
#> #   active <dbl>, combined_key <chr>, incidence_rate <dbl>,
#> #   case_fatality_ratio <dbl>, incident_rate <dbl>
```

- `map()`
- `map2()`
- `map_dfr()`
- **`map_*()`**

`map()` variants

- `map(.x, .f)` maps over 1 argument returns a list
- `map_dbl(.x, .f)` returns a double vector
- `map_chr(.x, .f)` returns a character vector
- `map_lgl(.x, .f)` returns a logical vector
- `map_dfr(.x, .f)` returns a data frame created by row-binding
- `map_dfc(.x, .f)` returns a data frame created by col-binding

- `map2_*(.x, .y, .f)` maps over 2 arguments
- `pmap_*(.l, .f)` maps over > 2 arguments

Wrangle a bit

```
covid19_bystate <- covid19_raw %>%
  select(date, country_region:recovered) %>%
  mutate(
    date = mdy(date),
    country_region = case_when(
      country_region == "Korea, South" ~ "South Korea",
      country_region == "Mainland China" ~ "China",
      TRUE ~ country_region)
  )
covid19_bystate
```

```
#> # A tibble: 1,069,141 x 5
#>   date       country_region confirmed deaths recovered
#>   <date>     <chr>              <dbl>  <dbl>     <dbl>
#> 1 2020-03-01 China              66907   2761     31536
#> 2 2020-03-01 South Korea         3736     17        30
#> 3 2020-03-01 Italy               1694     34        83
#> 4 2020-03-01 China               1349      7      1016
#> 5 2020-03-01 China               1272     22      1198
#> 6 2020-03-01 China               1205      1      1046
#> #   with 1,069,135 more rows
```



```
covid19_byregion <- covid19_bystate %>%
  group_by(country_region, date) %>%
  summarise(
    confirmed = sum(confirmed, na.rm = TRUE),
    deaths = sum(deaths, na.rm = TRUE),
    recovered = sum(recovered, na.rm = TRUE)
  ) %>%
  mutate(
    new_confirmed = confirmed - lag(confirmed, na.rm = TRUE),
    new_deaths = deaths - lag(deaths, na.rm = TRUE),
    new_recovered = recovered - lag(recovered, na.rm = TRUE)
  ) %>%
  ungroup()
```

Don't wanna type 🤪, but I typed...

Apply a function `across()` multiple columns

```
covid19_byregion <- covid19_bystate %>%
  group_by(country_region, date) %>%
  summarise(
    across(
      confirmed:recovered,
      sum, na.rm = TRUE)
  ) %>%
  mutate(
    across(
      confirmed:recovered,
      ~ .x - lag(.x, na.rm = TRUE),
      .names = "new_{.col}")
  ) %>%
  ungroup()
```

```
covid19_byregion <- covid19_bystate %>%
  group_by(country_region, date) %>%
  summarise(
    across(
      where(is.numeric),
      sum, na.rm = TRUE)
  ) %>%
  mutate(
    across(
      where(is.numeric),
      ~ .x - lag(.x, na.rm = TRUE),
      .names = "new_{.col}")
  ) %>%
  ungroup()
```



```
covid19_byregion
```

```
#> # A tibble: 56,107 x 8
#>   country_region date       confirmed deaths recovered
#>   <chr>          <date>         <dbl>  <dbl>     <dbl>
#> 1 Afghanistan    2020-03-01         1      0         0
#> 2 Afghanistan    2020-03-02         1      0         0
#> 3 Afghanistan    2020-03-03         2      0         0
#> 4 Afghanistan    2020-03-04         4      0         0
#> 5 Afghanistan    2020-03-05         4      0         0
#> 6 Afghanistan    2020-03-06         4      0         0
#> # … with 56,101 more rows, and 3 more variables:
#> #   new_confirmed <dbl>, new_deaths <dbl>,
#> #   new_recovered <dbl>
```


`nest()` multiple columns into a list-column

```
covid19_lstcol <- covid19_byregion %>%
  nest(data = -country_region)
covid19_lstcol
```

```
#> # A tibble: 236 x 2
#>   country_region      data
#>   <chr>               <list>
#> 1 Afghanistan         <tibble[,7] [306 × 7]>
#> 2 Albania             <tibble[,7] [298 × 7]>
#> 3 Algeria             <tibble[,7] [306 × 7]>
#> 4 Andorra             <tibble[,7] [305 × 7]>
#> 5 Angola              <tibble[,7] [287 × 7]>
#> 6 Antigua and Barbuda <tibble[,7] [294 × 7]>
#> # … with 230 more rows
```

Look inside the list-column

```
covid19_au <- covid19_lstcol$data[[10]]
covid19_au
```

```
#> # A tibble: 306 x 7
#>   date       confirmed deaths recovered new_confirmed
#>   <date>         <dbl>  <dbl>     <dbl>         <dbl>
#> 1 2020-03-01        27      1        11            NA
#> 2 2020-03-02        30      1        11             3
#> 3 2020-03-03        39      1        11             9
#> 4 2020-03-04        52      2        11            13
#> 5 2020-03-05        55      2        21             3
#> 6 2020-03-06        60      2        21             5
#> # … with 300 more rows, and 2 more variables:
#> #   new_deaths <dbl>, new_recovered <dbl>
```

Fit regression models and extract slopes

```
fit_au <- lm(new_recovered ~ lag(new_confirmed, n = 14),
  data = covid19_au)
fit_au
```

```
#>
#> Call:
#> lm(formula = new_recovered ~ lag(new_confirmed, n = 14), data =
covid19_au)
#>
#> Coefficients:
#>                 (Intercept)  lag(new_confirmed, n = 14)
#>                     26.3548                      0.5299
```

```
slope_au <- coef(fit_au)[2]
slope_au
```

```
#> lag(new_confirmed, n = 14)
#>                  0.5299203
```


Function time

```
extract_slope <- function(data) {
  fit <- lm(new_recovered ~ lag(new_confirmed, n = 14), data)
  coef(fit)[2]
}
```

Oops, `map()` isn't happy. 😕

```
covid19_lstcol$data %>%
  map(extract_slope)
```

```
#> Error in lm.fit(x, y, offset = offset, singular.ok =
singular.ok, ...): 0 (non-NA) cases
```


If an error occured, `safely()` captures the error

- returns a named list of `result` and `error`

```
safely_extract_slope <- safely(extract_slope,
  otherwise = NA_real_)
covid19_lstcol$data %>% map(safely_extract_slope)
```

```
#> [[1]]
#> [[1]]$result
#> lag(new_confirmed, n = 14)
#>                  0.6920523
#>
#> [[1]]$error
#> NULL
#>
#>
#> [[2]]
#> [[2]]$result
#> lag(new_confirmed, n = 14)
#>                  0.6435756
```


Pull out results

```
covid19_lstcol$data %>%
  map(safely_extract_slope) %>%
  map("result")
```

```
#> [[1]]
#> lag(new_confirmed, n = 14)
#>                  0.6920523
#>
#> [[2]]
#> lag(new_confirmed, n = 14)
#>                  0.6435756
#>
#> [[3]]
#> lag(new_confirmed, n = 14)
#>                  0.5329582
#>
#> [[4]]
#> lag(new_confirmed, n = 14)
#>                  0.5595175
#>
#> [[5]]
```

```
covid19_lstcol$data %>%
  map(safely_extract_slope) %>%
  map_dbl("result")
```

```
#>   [1]  6.920523e-01  6.435756e-01
#>   [3]  5.329582e-01  5.595175e-01
#>   [5]  5.196389e-01  7.375993e-01
#>   [7]  8.669078e-01  8.971555e-01
#>   [9]             NA  5.299203e-01
#>  [11]  9.182422e-01  9.642226e-01
#>  [13]  2.903094e-01             NA
#>  [15]  8.760548e-01  8.306039e-01
#>  [17]  1.478383e-01  1.038319e+00
#>  [19] -1.417372e-01  3.760244e-01
#>  [21]  2.665434e-01  7.554900e-02
#>  [23]  6.108023e-01  6.446241e-01
#>  [25]  4.912720e-01  6.540097e-01
#>  [27]  4.138478e-01  5.732950e-01
#>  [29]  6.704973e-01  8.998348e-01
#>  [31]  5.736442e-02  5.809416e-01
#>  [33]  1.256359e-01  1.167159e-01
```

Alternatively, `mutate()` with `map()` inside a tibble

```
covid19_lstcol %>%
  mutate(
    slopes = map(data, safely_extract_slope) %>%
      map_dbl("result"))
```

```
#> # A tibble: 236 x 3
#>   country_region      data                  slopes
#>   <chr>               <list>                 <dbl>
#> 1 Afghanistan         <tibble[,7] [306 × 7]>  0.692
#> 2 Albania             <tibble[,7] [298 × 7]>  0.644
#> 3 Algeria             <tibble[,7] [306 × 7]>  0.533
#> 4 Andorra             <tibble[,7] [305 × 7]>  0.560
#> 5 Angola              <tibble[,7] [287 × 7]>  0.520
#> 6 Antigua and Barbuda <tibble[,7] [294 × 7]>  0.738
#> # … with 230 more rows
```


Side-effects

Most functions are called for the value that they return. But some functions are called primarily for their side-effects, for example:

- `print()` displays values/plots to the screen.
- `write_csv()` writes values to the CSV file saved in the disk.

No values/outputs to be captured.

☐ We're going to save each region's data into separate CSV files to make it easier to import in the future.

`group_split()` splits tibble into chunks

```
covid19_lst <- covid19_byregion %>%
  group_by(country_region) %>%
  group_split()
covid19_lst[[1]]
```

```
#> # A tibble: 306 x 8
#>   country_region date       confirmed deaths recovered
#>   <chr>          <date>         <dbl>  <dbl>     <dbl>
#> 1 Afghanistan    2020-03-01         1      0         0
#> 2 Afghanistan    2020-03-02         1      0         0
#> 3 Afghanistan    2020-03-03         2      0         0
#> 4 Afghanistan    2020-03-04         4      0         0
#> 5 Afghanistan    2020-03-05         4      0         0
#> 6 Afghanistan    2020-03-06         4      0         0
#> # … with 300 more rows, and 3 more variables:
#> #   new_confirmed <dbl>, new_deaths <dbl>,
#> #   new_recovered <dbl>
```


`walk()` with side-effects, if no outputs

```
region_names <- unique(covid19_byregion$country_region)
file_name <- glue::glue("data/covid19/{region_names}.csv")
# fs::dir_create("data/covid19")
walk2(covid19_lst, file_name, write_csv)
```

```
fs::dir_ls("data/covid19")
```

```
#> data/covid19/Afghanistan.csv
#> data/covid19/Albania.csv
#> data/covid19/Algeria.csv
#> data/covid19/Andorra.csv
#> data/covid19/Angola.csv
#> data/covid19/Antigua and Barbuda.csv
#> data/covid19/Argentina.csv
#> data/covid19/Armenia.csv
#> data/covid19/Aruba.csv
#> data/covid19/Australia.csv
#> data/covid19/Austria.csv
```

{fs} provides a cross-platform, uniform interface to file system operations.

```
library(fs)
covid19_local <- map_dfr(
  dir_ls("data/covid19", glob = "*.csv"), read_csv)
covid19_local
```

```
#> # A tibble: 56,107 x 8
#>   country_region date       confirmed deaths recovered
#>   <chr>          <date>         <dbl>  <dbl>     <dbl>
#> 1 Afghanistan    2020-03-01         1      0         0
#> 2 Afghanistan    2020-03-02         1      0         0
#> 3 Afghanistan    2020-03-03         2      0         0
#> 4 Afghanistan    2020-03-04         4      0         0
#> 5 Afghanistan    2020-03-05         4      0         0
#> 6 Afghanistan    2020-03-06         4      0         0
#> # … with 56,101 more rows, and 3 more variables:
#> #   new_confirmed <dbl>, new_deaths <dbl>,
#> #   new_recovered <dbl>
```


Reading

- Functions
- Iteration
- Functionals