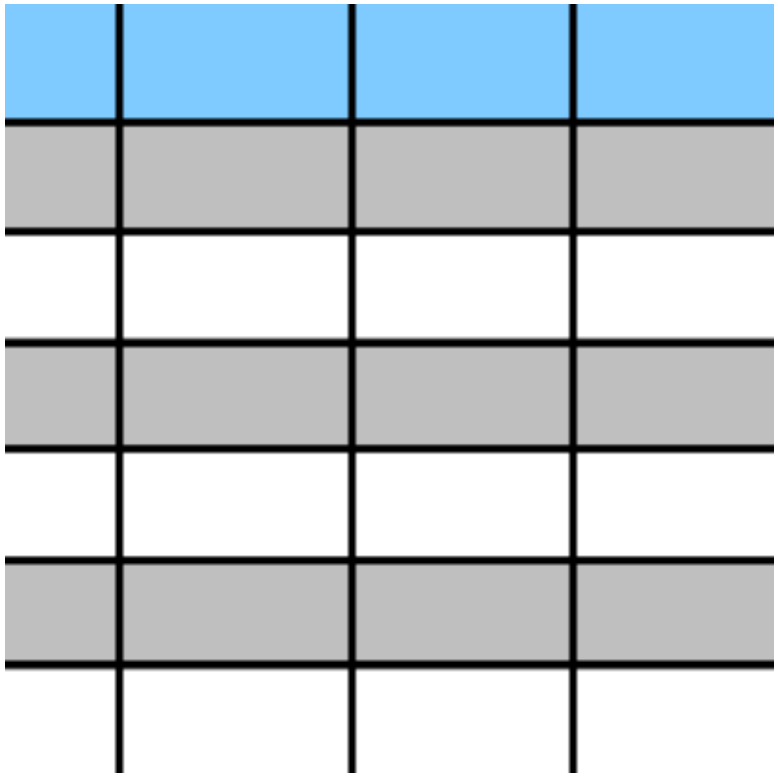
HEALTH
UNIVERSITY OF UTAH

Getting Data

Andrew Redd, PhD.

R Bootcamp 2021



Getting Data

Local Data

Data in R is held locally (i.e. in memory).

Most functions for reading data are simple and obviously named.

- read.csv
    - reads in a comma separated file.
- read.fwf
    - reads fixed width format. ## COVID Data

The first data set we will use was compiled by the John's Hopkins Coronavirus Resource Center and can be obtained from https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases (https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases) or kaggle (https://www.kaggle.com/baguspurnama/covid-confirmed-global). It gives worldwide data for the COVID-19 pandemic from January 2020 through mid July 2021.

The data is split into three files;

- `confirmed.csv` for confirmed cases,

Read in the data

```
confirmed.data.raw.base <- read.csv("data/confirmed.csv")
head(confirmed.data.raw.base)
```

Province.State	Country.Region	Lat	Long	X1.22.20	X1.23.20	X1.24.20	X1.25.20	X1.26.20	X1.27
	Afghanistan	33.93911	67.70995	0	0	0	0	0	
	Albania	41.15330	20.16830	0	0	0	0	0	
	Algeria	28.03390	1.65960	0	0	0	0	0	
	Andorra	42.50630	1.52180	0	0	0	0	0	
	Angola	-11.20270	17.87390	0	0	0	0	0	
	Antigua and Barbuda	17.06080	-61.79640	0	0	0	0	0	

Tidy solution: readr

```
confirmed.data.raw.tidy <- readr::read_csv("data/confirmed.csv")
```

```
Message:##
## -- Column specification --------------------------------
## cols(
##   .default = col_double(),
##   `Province/State` = col_character(),
##   `Country/Region` = col_character()
## )
## i Use `spec()` for the full column specifications.
```

Tidy output

```
head(confirmed.data.raw.tidy)
```

Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20
NA	Afghanistan	33.93911	67.70995	0	0	0	0	0	0
NA	Albania	41.15330	20.16830	0	0	0	0	0	0
NA	Algeria	28.03390	1.65960	0	0	0	0	0	0
NA	Andorra	42.50630	1.52180	0	0	0	0	0	0
NA	Angola	-11.20270	17.87390	0	0	0	0	0	0
NA	Antigua and Barbuda	17.06080	-61.79640	0	0	0	0	0	0

Task

Find a function that reads in Microsoft Excel ‘xlsx’ files. Refer back to the Resources slides (Resources.html) if you need.

2:00

Possible Answers

- `readxl::read_excel()` (https://www.rdocumentation.org/packages/readxl/versions/1.3.1/topics/read_exce
- `officer::read_excel()` (https://www.rdocumentation.org/packages/officer/versions/0.3.5/topics/read_xlsx)
- `openxlsx::read.xlsx()` (https://www.rdocumentation.org/packages/openxlsx/versions/4.1.0.1/topics/read.
- `xlsx::read.xlsx()` (https://www.rdocumentation.org/packages/xlsx/versions/0.6.1/topics/read.xlsx)

Excel solution

```
confirmed.data.raw.xl <- readxl::read_excel("data/confirmed.xlsx")
head(confirmed.data.raw.xl)
```

Province/State	Country/Region	Lat	Long	43852	43853	43854	43855	43856	43857	43858	4
NA	Afghanistan	33.93911	67.70995	0	0	0	0	0	0	0	
NA	Albania	41.15330	20.16830	0	0	0	0	0	0	0	
NA	Algeria	28.03390	1.65960	0	0	0	0	0	0	0	
NA	Andorra	42.50630	1.52180	0	0	0	0	0	0	0	
NA	Angola	-11.20270	17.87390	0	0	0	0	0	0	0	
NA	Antigua and Barbuda	17.06080	-61.79640	0	0	0	0	0	0	0	

Examining data

Useful tools for examining data.

- `str()` - ‘structure’ of the data.
- `glimpse()` - more useful version of str and works on all tibbles.
- `head()` - first n rows.
- `tail()` - last n rows
- `summary()` - will give univariate summaries of variables.

glimpse()

```
glimpse(confirmed.data.raw.tidy)
```

Problem

This is only one set ☹

Purrr to the rescue

```
library(readr)
library(purrr)
(data.files <- list.files("data", pattern=".*[.]csv", full.names=TRUE))
```

```
## [1] "data/confirmed.csv" "data/deaths.csv"
## [3] "data/recovered.csv"
```

```
covid.data.wide.raw <- map(data.files, read_csv)
```

column specification messages suppressed for this slide

Mapping

Map Definition

Apply a function to each element of a list or vector and return the results.

Example

```
map(data.files, read_csv)
```

Call the function `read_csv()` for each element of `data.files`, or which there are three; `confirmed.csv`, `deaths.csv` and `recovered.csv`.

We expect a list of three elements, each a `tibble` or `data.frame` with 449 columns.

Checks

```
length(covid.data.wide.raw)
```

```
## [1] 3
```

```
map(covid.data.wide.raw, class)
```

```
## [[1]]
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
##
## [[2]]
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
##
## [[3]]
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
map(covid.data.wide.raw, dim)
```

Map Variants by output

- `map()` = list,
- `map_lgl()` = logical
- `map_int()` = integers
- `map_dbl()` = numbers
- `map_chr()` = strings
- `map_dfc()` = column bound data
- `map_dfr()` = row bound data

Other *map* variants

- `map2()` & `map2_*()` = take two lists or vectors of the same length and call the given function for each pair of inputs.
- `imap()` & `imap_*()` = index map, functions must accept the element and the index or name of the element. Equivalent to `map2(x, names(x), fun)` or `map2(x, seq_along(x), fun)`
- `pmap()` & `pmap_*()` generalization of map 2 to arbitrary number where the vectors must be provided as a list.
- `map_if()` apply only to elements matching a given predicate, leave others alone.
- `map_at()` apply only at specified indices, leave others alone.

Quiz

What form is the data in?

What form do we want it in?

Task

Collapse it into a single data.frame

Possible solution 1: combind_rows

```
covid.data.wide.raw %>%
    bind_rows(.id="file") %>%
    select(file, 1:10) %>%
    glimpse()
```

```
## Rows: 807
## Columns: 10
## $ file             [3m[38;5;246m<chr>[39m[23m "1", "1", "1", "1", "1", "1", "1"~
## $ `Province/State` [3m[38;5;246m<chr>[39m[23m NA, NA, NA, NA, NA, NA, NA, NA, "~
## $ `Country/Region` [3m[38;5;246m<chr>[39m[23m "Afghanistan", "Albania", "Algeri~
## $ Lat              [3m[38;5;246m<dbl>[39m[23m 33.93911, 41.15330, 28.03390, 42.~
## $ Long             [3m[38;5;246m<dbl>[39m[23m 67.70995, 20.16830, 1.65960, 1.52~
## $ `1/22/20`        [3m[38;5;246m<dbl>[39m[23m 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ `1/23/20`        [3m[38;5;246m<dbl>[39m[23m 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ `1/24/20`        [3m[38;5;246m<dbl>[39m[23m 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ `1/25/20`        [3m[38;5;246m<dbl>[39m[23m 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ `1/26/20`        [3m[38;5;246m<dbl>[39m[23m 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, ~
```

Possible solution 2: Change the mapping function

but also get the right name.

```
covid.data.wide.dfr <-
    data.files %>%
    set_names(.,.) %>%
    map_dfr(read_csv, .id = "file")
```

Time for an aside on the pipe.

Manipulating the pipe

```
data.files %>%
    set_names(.,.)
```

The dot or period `.` has special meaning in the context of the pipe; it specifies where the argument should go.

Since the dot is repeated we use the argument twice. So the statement is equivalent to:

```
    set_names(data.fies, data.files)
```

Pipe functions

When the `.` starts a statement it creates a function of one argument.

```
    . %>% set_names(.,.)
```

```
## Functional sequence with the following components:
##
##  1. set_names(., .)
##
## Use 'functions' to extract the individual functions.
```

This is functionally equivalent to

```
function(x)set_names(x,x)
```

```
## function(x)set_names(x,x)
## <environment: 0x0000024e55964088>
```

Pipe functions are one shorthand for declaring functions.

Lambda functions

Formula based lambda function constitute the second shorthand for declaring functions, and can accept multiple arguments.

yes, R stole this from python

Terminology

Lambda functions

Functions created from formulas(~) with predefined arguments

```
?rlang::is_lambda
```

Arguments

```
rlang::as_function(~set_names(.x, .x))
```

```
## <lambda>
## function (..., .x = ..1, .y = ..2, . = ..1)
## set_names(.x, .x)
## <environment: 0x0000024e55964088>
## attr(,"class")
## [1] "rlang_lambda_function" "function"
```

The usable arguments are:

- `...` for variadic arguments
- `.`, `..1`, or `.x` is always the first argument
- `..2`, or `.y` is the second argument


AND NOW BACK TO
OUR REGULARLY
SCHEDULED
PROGRAMMING

Save it for future use

```
save(covid.data.wide.dfr, file='data/covid.data.wide.dfr.RData')
```

OR

```
saveRDS(covid.data.wide.dfr, file='data/covid.data.wide.dfr.rds')
```

Next Up

Data Wrangling (07-DataWrangling.html)