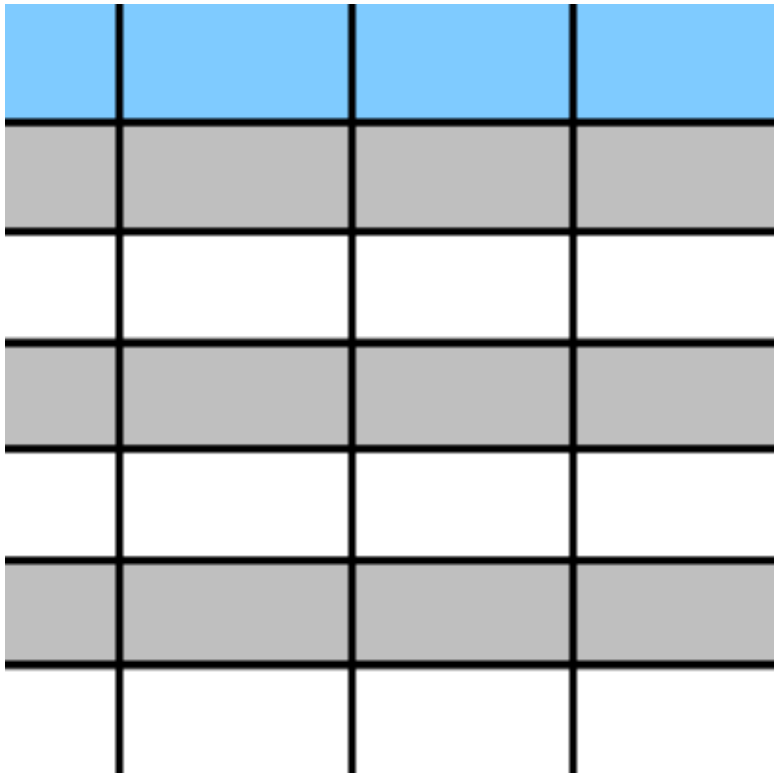
Getting Data

Andrew Redd, PhD.

R Bootcamp 2020



Getting Data

Local Data

Most 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.

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)

Ebola Data

The first data set we will use comes from the Humanitarian Data Exchange (https://data.humdata.org/dataset/ebola-cases-2014).
It consists of the number of Ebola cases and deaths broken down by confirmed, probable and suspected.

This is a good case of slightly messy data that will need a bit of cleaning before it is really usable.

Read in the data

```
ebola.data.raw <- readxl::read_excel("data/ebola-cases-and-deaths-who-gar-sitrep.xlsx")
head(ebola.data.raw)
```

Country	Case def.	Total cases	Cases last 21 days	Total deaths	Country Report Date
Guinea	Confirmed	3351	0	2083	2015-12-27
Guinea	Probable	453	NA	453	NA
Guinea	Suspected	0	NA	NA	NA
Guinea	All	3804	0	2536	NA
Liberia	Confirmed	3151	NA	NA	2015-05-09
Liberia	Probable	1879	NA	NA	NA

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(ebola.data.raw)
```

```
## Observations: 44
## Variables: 6
## $ Country               <chr> "Guinea", "Guinea", "Guin...
## $ `Case def.`           <chr> "Confirmed", "Probable", ...
## $ `Total cases`         <dbl> 3351, 453, 0, 3804, 3151,...
## $ `Cases last 21 days`  <dbl> 0, NA, NA, 0, NA, NA, NA,...
## $ `Total deaths`        <dbl> 2083, 453, NA, 2536, NA, ...
## $ `Country Report Date` <dttm> 2015-12-27, NA, NA, NA, ...
```

Problem

This is only one sheet 😕

Read all the data

```
library(readxl)
library(purrr)
data.file <- "data/ebola-cases-and-deaths-who-gar-sitrep.xlsx"
sheets <- excel_sheets(data.file)
all.ebola.raw.data <- map(sheets, read_excel, path=data.file) %>%
    set_names(sheets)
```

Variants

- `map()` = list,
- `map_lgl()` = logical,
- `map_int()` = integers,
- `map_dbl()` = numbers,
- `map_chr()` = strings.

Quiz

What form is the all.ebola.raw.data in?

What form do we want it in?

Task

Collapse it into a single data.frame

Possible solution

Change the reading function

```
all.ebola.raw.data <-
    excel_sheets(data.file) %>% #< sheet names
    map_dfr(read_excel, path=data.file, .id = "sheet")
```

ERROR:

Column `Total deaths` can't be converted from numeric to character

Another solution - Fix the output

1. Investigate why it failed.

```
all.ebola.raw.data %>% keep(~is.character(.$`Total deaths`))
```

Terminology

Lambda functions

Functions created from formulas (R) where the . is the argument.

Not everything in R supports lambda functions but they can be made explicit by wrapping it in `as_function()`.

```
?rlang::is_lambda
```

Problems

1. Some variables are ‘character’ that should be ‘numeric’
2. Not all sheets have the same number of columns.

Another solution - Fix the output

2. Find the common variables

```
common.vars <- all.ebola.raw.data %>%
    # a. Get the variable names for for each sheet
    map(tbl_vars) %>%
    # b. Only keep the variables that are common to all
    reduce(intersect)
```

Key Function

`reduce()`

Take the first two elements apply the function, then take the output of that and the next element and apply the function, and so on until all elements have had the function applied. Return value of the last call.

Another solution - Fix the output

3. Combine together

```
all.ebola.raw.data.df <- all.ebola.raw.data %>%
    map(select, !!!common.vars) %>%
    map( mutate_at, vars(starts_with("Total"))
       , ~as.integer(na_if(., '..'))) %>%
    bind_rows(.id = "SheetName")
```

Terminology

Lazy evaluation

The select function expects unquoted variable names because it supports ‘lazy evaluation’. To over come this we need `!!!` or `!!` to indicate that the name should be interpreted as an object to be evaluated rather than a variable name directly.

Time for some clean up

```
ebola.data <- all.ebola.raw.data.df %>%
    # 1. Carry last value forward on date.
    mutate(`Country Report Date` = zoo::na.locf(`Country Report Date`)) %>%
    # 2. Fill in the missing values with 0.
    mutate_at(vars(starts_with("total")), coalesce, 0L)
```

Concept

Note the `0L` which indicates an integer zero. If performed with the bare `0` the operation would fail with

ERROR:

Argument 2 must be an integer vector, not a double vector

Save it for future use

```
save(ebola.data, file='data/ebola.data.RData')
```

OR

```
saveRDS(ebola.data, file='data/ebola.data.rds')
```