ANT 6973: DATA VISUALIZATION AND EXPLORATION

IMPORTING AND ORGANIZING DATA

TODAY'S TOPICS

- Pipes
- File paths (`here`)
- Reading from flat files (`readr`)
- Reading from spreadsheets (`readxl`)

PIPES

PIPES

- Pass thing on left into function on the right (as the first argument)

```
x %>% f() becomes f(x)
```

```
x %>% f(y) becomes f(x, y)
```

PIPES

- Why use this? Suppose you have a *sequence* of operations to perform on your data.

```
my_data <- 1:10
```

- *"Take the data, first exponentiate each number, then take the square root of each number, then take the mean."*

PIPES

- Option 1: execute each function one by one, using the previous result in each subsequent function

```
result_1 <- exp(my_data)
result_2 <- sqrt(result_1)
result_final <- mean(result_2)
```

PIPES

- Option 2: nest the functions; functions will be evaluated from inner-most to outer-most

```
result_final <- mean(sqrt(exp(my_data)))
```

PIPES

- Option 3: chain the functions together in their natural sequence using pipes

```
result_final <- my_data %>%
  exp() %>%
  sqrt() %>%
  mean()
```

- More human-readable:

"Take the data, then exponentiate each number, then take the square root of each number, then take the mean."

PIPES

- All tidyverse packages (except for ggplot2) are designed to facilitate pipe-based workflows
    - The first argument of most functions is the "data."
- ggplot2 is an older package that predates the pipe
    - It's too late to change ggplot2 now, and it has its own pipe of sorts (the "+" operator)

PIPES


```
library("tidyverse")
library("gapminder")

count(filter(gapminder, year == 2007), continent)
```

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

PIPES


```
library("tidyverse")
library("gapminder")

count(filter(gapminder, year == 2007), continent)
```

Data

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

PIPES


```
library("tidyverse")
library("gapminder")

count(filter(gapminder, year == 2007), continent)
```

Step 1

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

PIPES


```
library("tidyverse")
library("gapminder")

count(filter(gapminder, year == 2007), continent)
```

Step 2

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

PIPES

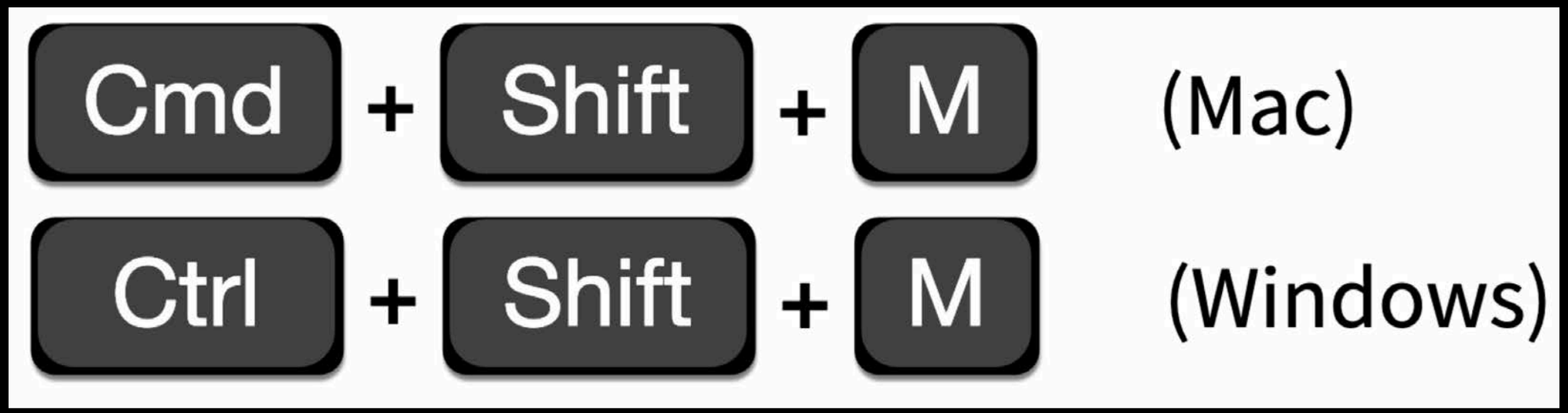

```
library("tidyverse")
library("gapminder")

gapminder %>%
  filter(year == 2007) %>%
  count(continent)
```

← Data
← Step 1
← Step 2

continent	n
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

PIPES

- Shortcut to type %>%



DATA IMPORT

PACKAGES FOR DATA IMPORT

`readr`

Core part of


```
library("tidyverse")
```

`readxl`

Not part of core

```
library("readxl")
```

PACKAGES FOR DATA IMPORT

haven

Import SAS, SPSS and STATA data files

And many more…

PREVIOUSLY...

- Import from a .csv file using `read_csv()`

```
my_data <- read_csv("path/to/data.csv")
```

New R object where data will be stored

`readr` function

Path of data file relative to project home

WRITE FILES

- Use the function `write_csv()` to write a data frame to disk.

```
write_csv(my_data, file = "path/my_data.csv")
```

`readr` function

Name of R object to export

Name & path of file you want to create
note: folder must exist

LET'S DETOUR A BIT TO LEARN MORE ABOUT PATHS...

- Go to this week's assignments on the course website.

- Download paths-project.zip and unzip the file somewhere on your computer.

- **Open the project!**

- Follow along as we complete the activity.

- *Nothing to turn in!*

LET'S GET BACK TO DATA IMPORT

`readr`

- Fast way of reading rectangular data like comma separated file (csv) and tab separated files (tsv).

FUNCTIONS

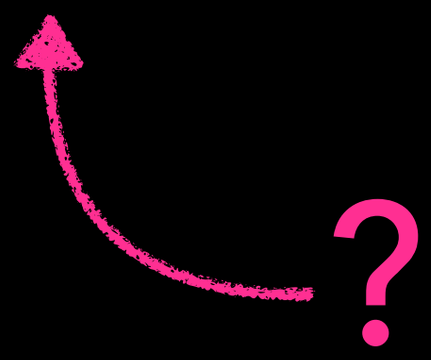
Function	Reads
read_csv()	Comma separated values
read_csv2()	Semi-colon separated values
read_delim()	General delimited files (e.g., "|")
read_fwf()	Fixed width files
read_table()	Space separated
read_table2()	Any number of whitespace characters
read_tsv()	Tab delimited values

FUNCTIONS

Function	Reads
read_csv()	Comma separated values
read_csv2()	Semi-colon separated values
read_delim()	General delimited files (e.g., "|")
read_fwf()	Fixed width files
read_table()	Space separated
read_table2()	Any number of whitespace characters
read_tsv()	Tab delimited values

WHY USE NOT USE BASE R?

- ~10X faster than base R functions like `read.csv()`

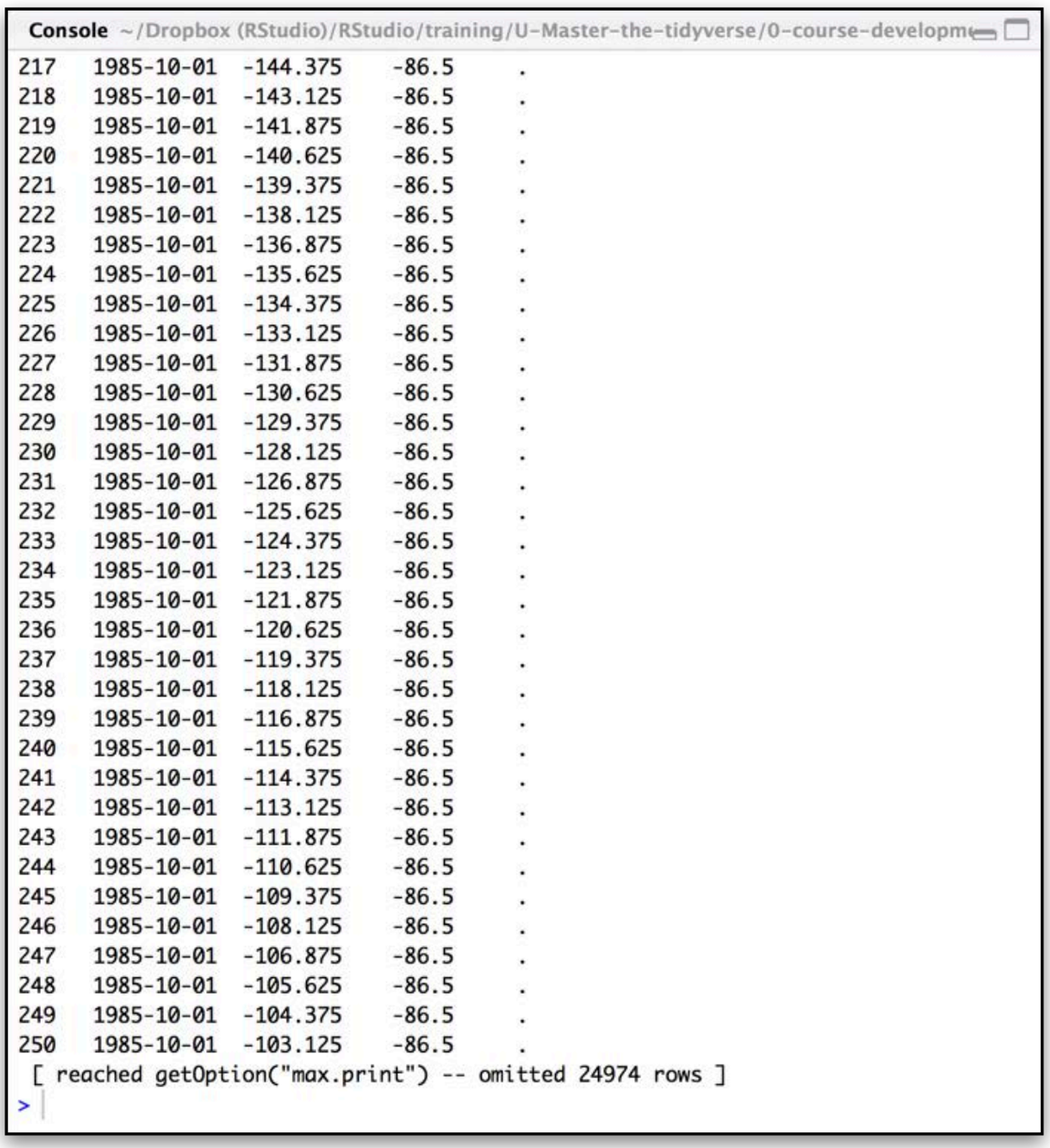


- Returns a tibble rather than a data.frame

TIBBLE

- A type of data frame used throughout the tidyverse

`read.csv() -> data.frame`

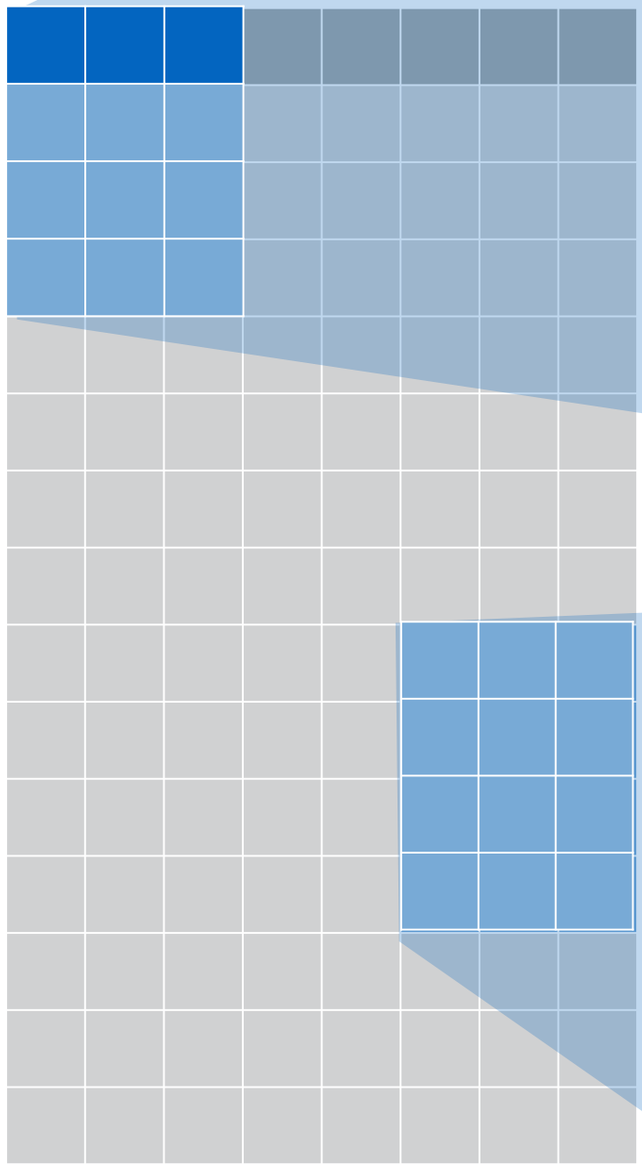


read.csv() -> data.frame

```
Console ~/Dropbox (RStudio)/RStudio/training/U-Master-the-tidyverse/0-course-developm
217   1985-10-01  -144.375    -86.5     .
218   1985-10-01  -143.125    -86.5     .
219   1985-10-01  -141.875    -86.5     .
220   1985-10-01  -140.625    -86.5     .
221   1985-10-01  -139.375    -86.5     .
222   1985-10-01  -138.125    -86.5     .
223   1985-10-01  -136.875    -86.5     .
224   1985-10-01  -135.625    -86.5     .
225   1985-10-01  -134.375    -86.5     .
226   1985-10-01  -133.125    -86.5     .
227   1985-10-01  -131.875    -86.5     .
228   1985-10-01  -130.625    -86.5     .
229   1985-10-01  -129.375    -86.5     .
230   1985-10-01  -128.125    -86.5     .
231   1985-10-01  -126.875    -86.5     .
232   1985-10-01  -125.625    -86.5     .
233   1985-10-01  -124.375    -86.5     .
234   1985-10-01  -123.125    -86.5     .
235   1985-10-01  -121.875    -86.5     .
236   1985-10-01  -120.625    -86.5     .
237   1985-10-01  -119.375    -86.5     .
238   1985-10-01  -118.125    -86.5     .
239   1985-10-01  -116.875    -86.5     .
240   1985-10-01  -115.625    -86.5     .
241   1985-10-01  -114.375    -86.5     .
242   1985-10-01  -113.125    -86.5     .
243   1985-10-01  -111.875    -86.5     .
244   1985-10-01  -110.625    -86.5     .
245   1985-10-01  -109.375    -86.5     .
246   1985-10-01  -108.125    -86.5     .
247   1985-10-01  -106.875    -86.5     .
248   1985-10-01  -105.625    -86.5     .
249   1985-10-01  -104.375    -86.5     .
250   1985-10-01  -103.125    -86.5     .
 [ reached getOption("max.print") -- omitted 24974 rows ]
>
```

read_csv() -> tibble

```
Console ~/Dropbox (RStudio)/RStudio/training/U-Master-the-tidyverse/0-course-developm
> nimbus
# A tibble: 25,224 x 4
         date longitude latitude ozone
       <dttm>     <dbl>    <dbl> <chr>
 1 1985-10-01  -179.375    -87.5     .
 2 1985-10-01  -178.125    -87.5     .
 3 1985-10-01  -176.875    -87.5     .
 4 1985-10-01  -175.625    -87.5     .
 5 1985-10-01  -174.375    -87.5     .
 6 1985-10-01  -173.125    -87.5     .
 7 1985-10-01  -171.875    -87.5     .
 8 1985-10-01  -170.625    -87.5     .
 9 1985-10-01  -169.375    -87.5     .
10 1985-10-01  -168.125    -87.5     .
# ... with 25,214 more rows
>
```



A large table to display

```
# A tibble: 234 × 6
   manufacturer      model displ
          <chr>      <chr> <dbl>
1          audi         a4   1.8
2          audi         a4   1.8
3          audi         a4   2.0
4          audi         a4   2.0
5          audi         a4   2.8
6          audi         a4   2.8
7          audi         a4   3.1
8          audi a4 quattro   1.8
9          audi a4 quattro   1.8
10         audi a4 quattro   2.0
# ... with 224 more rows, and 3
#   more variables: year <int>,
#   cyl <int>, trans <chr>
```

Dimensions

Column types

Concise view of data that fits on screen.

All other columns listed

tibble display

```
156 1999    6   auto(l4)
157 1999    6   auto(l4)
158 2008    6   auto(l4)
159 2008    8   auto(s4)
160 1999    4 manual(m5)
161 1999    4   auto(l4)
162 2008    4 manual(m5)
163 2008    4 manual(m5)
164 2008    4   auto(l4)
165 2008    4   auto(l4)
166 1999    4   auto(l4)
 [ reached getOption("max.print") --
omitted 68 rows ]
```

Large number of rows printed before giving up.

Many rows and variables hidden.

data frame display

ACTIVITY: OZONE

- Go to this week's assignments on the course website.

- Download ozone.Rmd and follow the instructions to complete the assignment.

- Submit the html report.

AND FINALLY...

`readxl`

data organization organizing data in spreadsheets

My collaborators sometimes ask me, “In what form would you like the data?” My response is always, “In its current form!” If the data need to be reformatted, it’s much better for me to write a script than for them to do a bunch of cut-and-paste. I’m a strong proponent of data analysts being able to handle any data files they might receive.

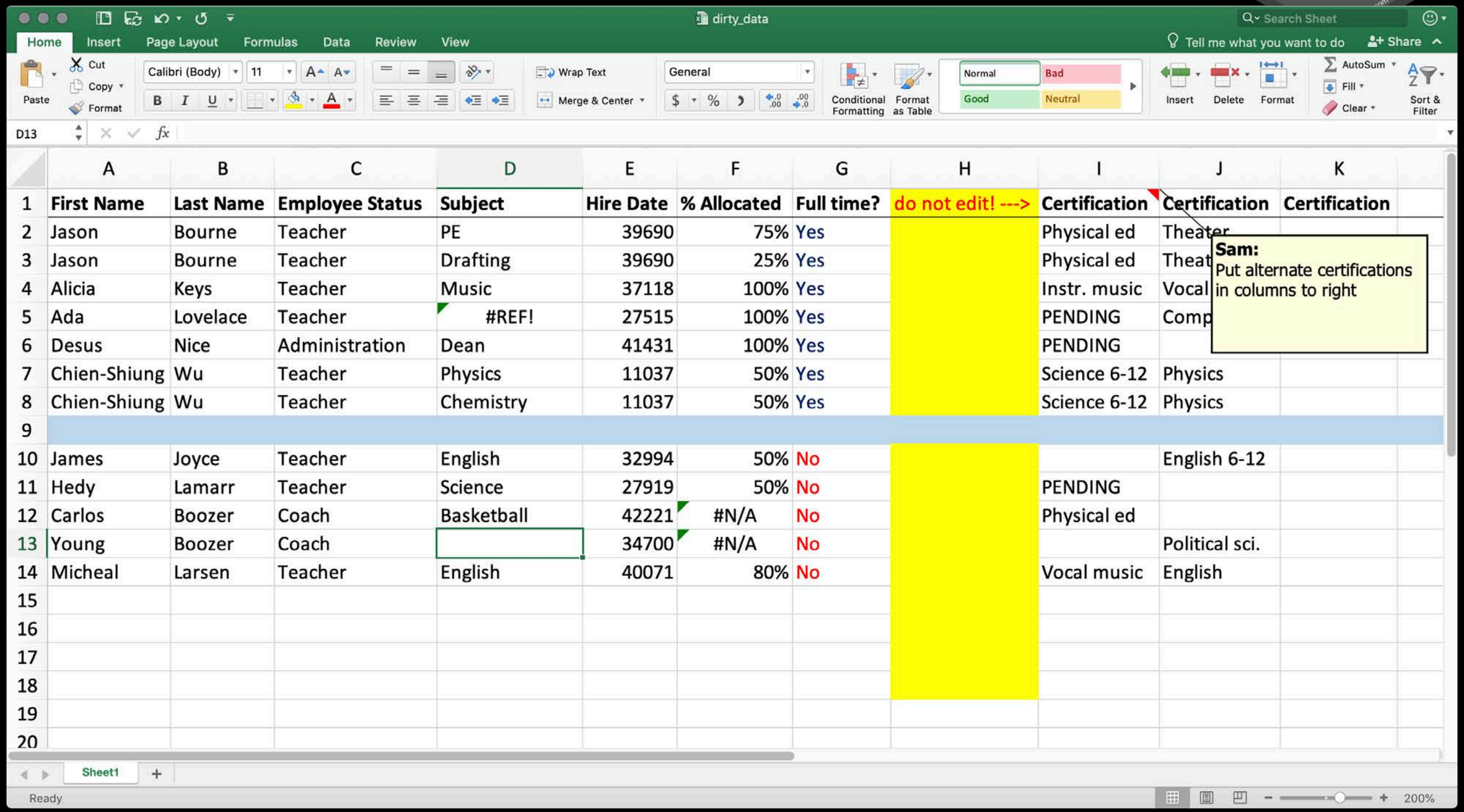
But in many cases, I have to spend **a lot** of time writing scripts to rearrange the layout of the data. And how would you like your data analysts to spend their time? Reorganizing data, or really analyzing data?

Most of my collaborators enter and store their data in spreadsheets, and mostly Microsoft Excel. Before starting to enter data into a spreadsheet, it’s good to spend some time thinking about the layout. The way that you organize the data in spreadsheets can have a big impact on your data analyst’s quality of life.

This is a tutorial on that topic: *how to organize data in spreadsheets*. For complex, high-dimensional data, it may be better to use a formal database. But for many projects, spreadsheets are perfectly fine. But data in spreadsheets can be pretty and easy to work with, or they can be a sloppy mess requiring serious downstream reorganization efforts. We want to avoid the latter.

I don’t think these ideas comes naturally to anyone. So if you’re not happy with the structure of your current data files, don’t despair! And also don’t apply tedious and potentially error-prone hand-editing to revise the arrangement. Rather, apply these principles when designing the layout for your next dataset, to help make analyses easier.

UGH...



	A	B	C	D	E	F	G	H	I	J	K
1	First Name	Last Name	Employee Status	Subject	Hire Date	% Allocated	Full time?	do not edit! --->	Certification	Certification	Certification
2	Jason	Bourne	Teacher	PE	39690	75%	Yes		Physical ed	Theater	
3	Jason	Bourne	Teacher	Drafting	39690	25%	Yes		Physical ed	Theat	
4	Alicia	Keys	Teacher	Music	37118	100%	Yes		Instr. music	Vocal	
5	Ada	Lovelace	Teacher	#REF!	27515	100%	Yes		PENDING	Comp	
6	Desus	Nice	Administration	Dean	41431	100%	Yes		PENDING		
7	Chien-Shiung	Wu	Teacher	Physics	11037	50%	Yes		Science 6-12	Physics	
8	Chien-Shiung	Wu	Teacher	Chemistry	11037	50%	Yes		Science 6-12	Physics	
9											
10	James	Joyce	Teacher	English	32994	50%	No			English 6-12	
11	Hedy	Lamarr	Teacher	Science	27919	50%	No		PENDING		
12	Carlos	Boozer	Coach	Basketball	42221	#N/A	No		Physical ed		
13	Young	Boozer	Coach		34700	#N/A	No			Political sci.	
14	Micheal	Larsen	Teacher	English	40071	80%	No		Vocal music	English	
15											
16											
17											
18											
19											
20											

READXL: A PRIMER

deaths

Search Sheet

Home Insert Page Layout Formulas Data Review View Table Tell me what you want to do Share

Paste Calibri (Body) 12 General Conditional Formatting Format as Table Cell Styles Insert Delete Format Sort & Filter

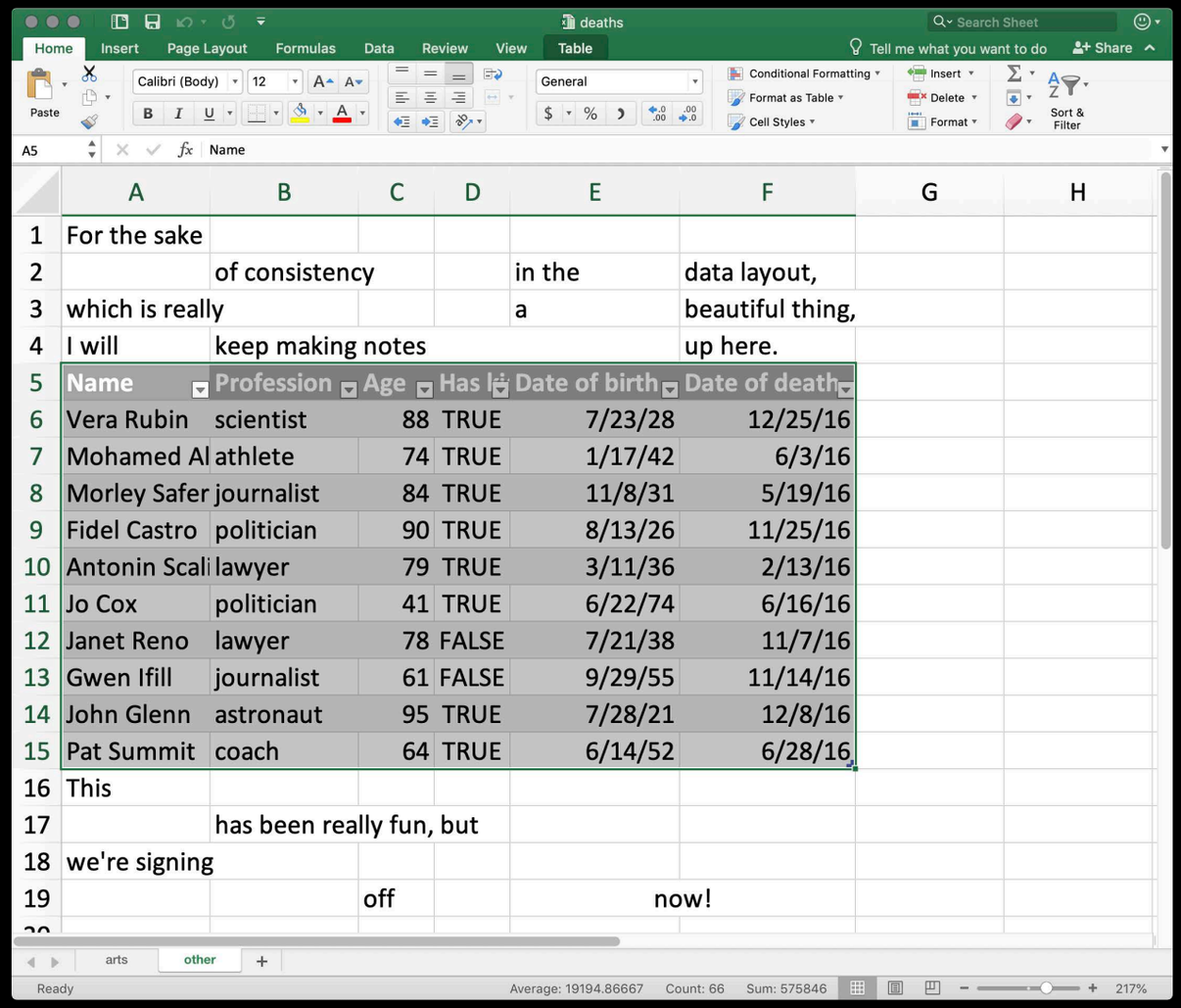
A5 Name

	A	B	C	D	E	F	G	H
1	For the sake							
2		of consistency			in the	data layout,		
3	which is really				a	beautiful thing,		
4	I will	keep making notes				up here.		
5	Name	Profession	Age	Has l	Date of birth	Date of death		
6	Vera Rubin	scientist	88	TRUE	7/23/28	12/25/16		
7	Mohamed Al	athlete	74	TRUE	1/17/42	6/3/16		
8	M							
9	Fi							
10	Ar							
11	Jo Cox	politician	41	TRUE	6/22/74	6/16/16		
12	Janet Reno	lawyer	78	FALSE	7/21/38	11/7/16		
13	Gwen Ifill	journalist	61	FALSE	9/29/55	11/14/16		
14	John Glenn	astronaut	95	TRUE	7/28/21	12/8/16		
15	Pat Summit	coach	64	TRUE	6/14/52	6/28/16		
16	This							
17		has been really fun, but						
18	we're signing							
19			off		now!			

`readxl` is all about the data rectangle

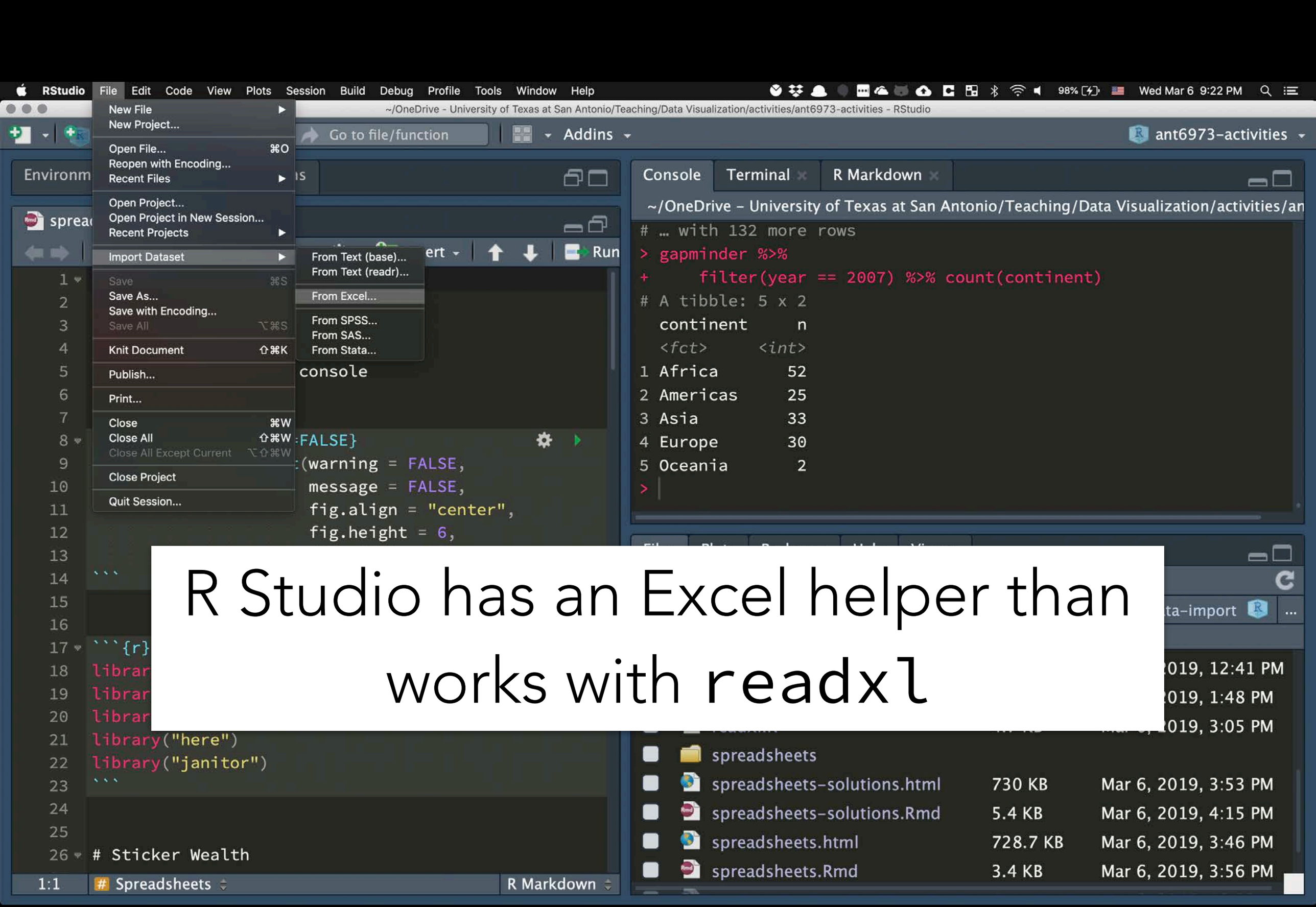
arts other +

Ready Average: 19194.86667 Count: 66 Sum: 575846 217%




```
read_excel("deaths.xlsx", skip = 4)
read_excel("deaths.xlsx", n_max = 15)
read_excel("deaths.xlsx", skip = 4, n_max = 10)
read_excel("deaths.xlsx", range = "A5:F15")
read_excel("deaths.xlsx", range = cell_rows(5:15))
read_excel("deaths.xlsx", range = cell_cols("A:F"))
```

R Studio has an Excel helper than works with `readxl`



~/OneDrive - University of Texas at San Antonio/Teaching/Data Visualization/activities/ant6973-activities - RStudio

Import Excel Data

File/Url:

~/OneDrive - University of Texas at San Antonio/Teaching/Data Visualization/activities/ant6973-activities/08_data-import/spreadsheets/deaths.xlsx

Browse...

Data Preview:

Name (character)	Profession (character)	Age (numeric)	Has kids (double)	Date of birth (date)	Date of death (date)
David Bowie	musician	69	1	Character	2016-01-10
Carrie Fisher	actor	60	1	Numeric	2016-12-27
Chuck Berry	musician	90	1	Date	2017-03-18
Bill Paxton	actor	61	1	Include	2017-02-25
Prince	musician	57	1	Skip	2016-04-21
Alan Rickman	actor	69	0		2016-01-14
Florence Henderson	actor	82	1	1934-02-14	2016-11-24
Harper Lee	author	89	0	1926-04-28	2016-02-19
Zsa Zsa Gábor	actor	99	1	1917-02-06	2016-12-18
George Michael	musician	53	0	1963-06-25	2016-12-25

Previewing first 50 entries.

Import Options:

Name: deaths

Sheet: Default

Range: A1:D10

Max Rows: 10

Skip: 4

NA:

First Row as Names

Open Data Viewer

Code Preview:

```
library(readxl)
deaths <- read_excel("08_data-import/spreadsheets/deaths.xlsx",
    col_types = c("text", "text", "numeric",
        "numeric", "date", "date"), skip = 4,
    n_max = 10)
View(deaths)
```

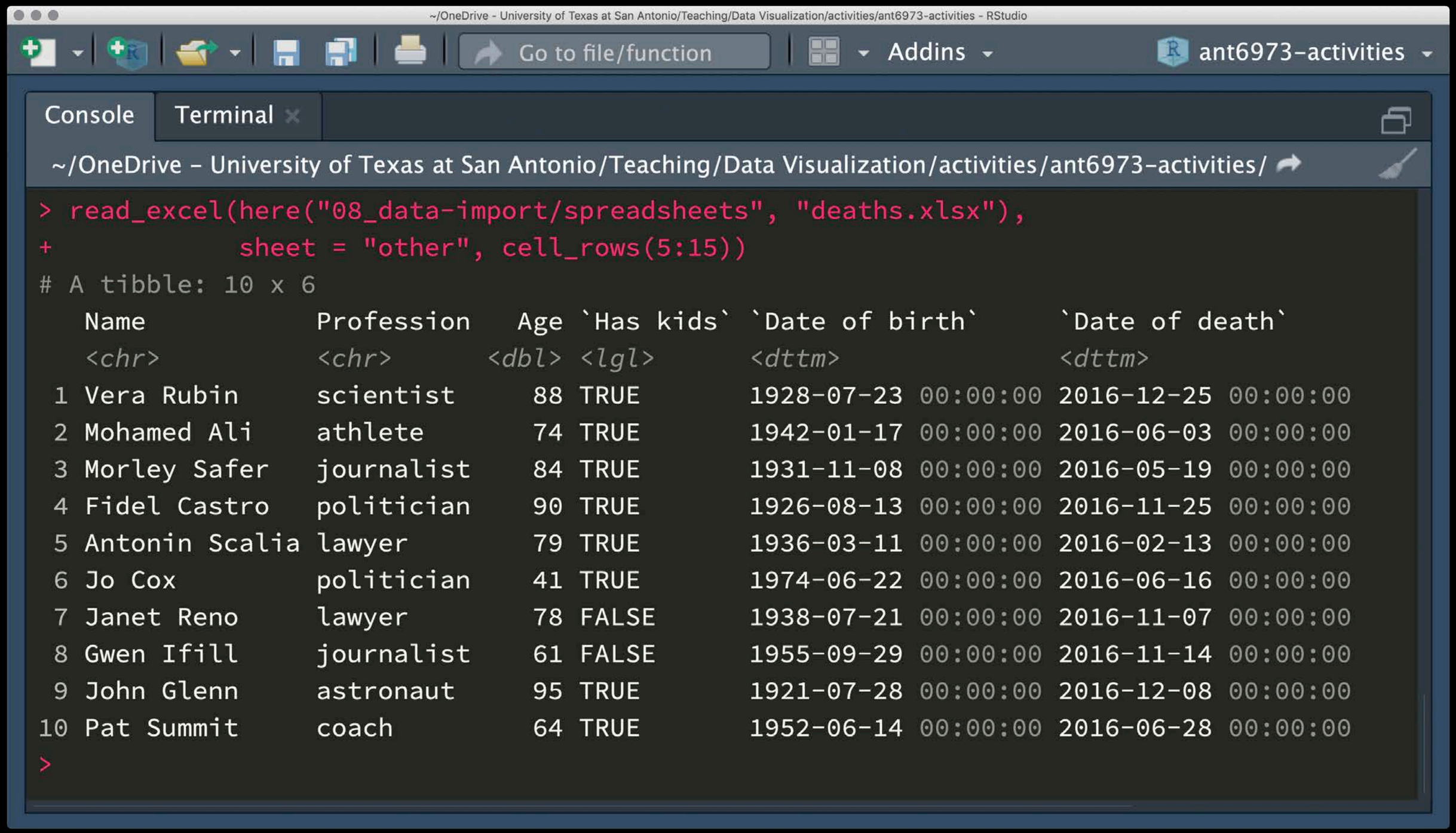
Reading Excel files using readxl

Import

Cancel

spreadsheets.Rmd 3.4 KB Mar 6, 2019, 3:56 PM

sst.R 1.4 KB Mar 6, 2019, 12:02 PM



`readxl` is good at guessing column types

ACTIVITY: SPREADSHEETS

- Go to this week's assignments on the course website.

- Download spreadsheets.Rmd and follow the instructions to complete the assignment.

- Submit the html report.

ACKNOWLEDGEMENTS

- Some ideas, examples, and figures from RStudio webinars, which are licensed CC by SA.