These materials adapted by Amelia McNamara from the RStudio CC BY-SA materials Introduction to R (2014) and Master the Tidyverse (2017).

Introduction to R & RStudio:

deck 06: Importing data

Amelia McNamara

Visiting Assistant Professor of Statistical and Data Sciences

Smith College

January 2018

@AmeliaMN

Katie Leap

Kelly O'Briant

@b23kelly @RLadiesDC

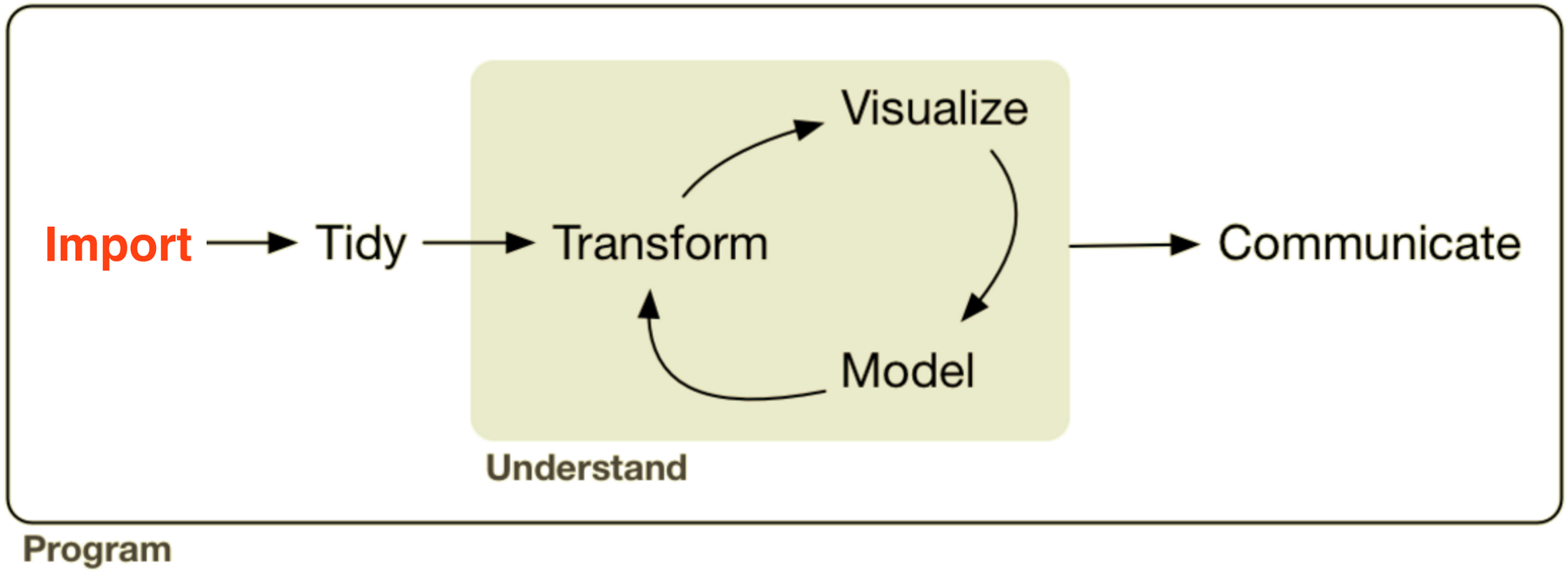
Your turn

Introduce yourself to your nearest 2-4 neighbors.

Schedule

9:00 -10:30	Import
10:30 - 11:00	Morning break (Seaport Foyer)
11:00 - 12:00	Best practices
12:00 - 1:00	Lunch (Grand Ballroom)
1:00 - 3:00	Transform and tidy
3:00 - 3:30	Afternoon break (Seaport Foyer)
3:30 - 5:00	Modeling and going forward

https://rstudio.cloud/project/13632



From *R for Data Science* by Hadley Wickham and Garrett Grolemund.

Importing Data

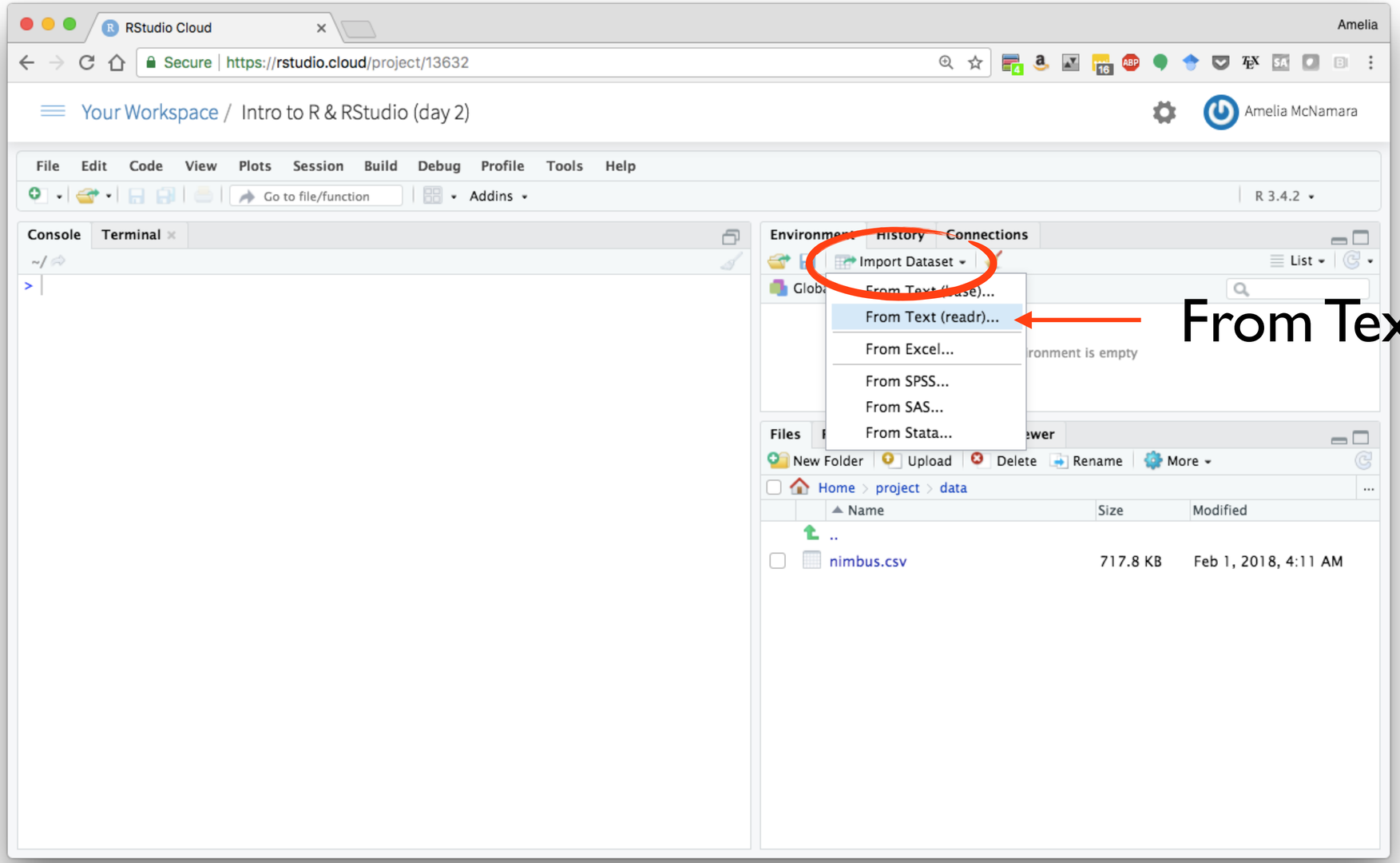
nimbus.csv

```
date,longitude,latitude,ozone
1985-10-01T00:00:00Z,-179.375,-87.5,.
1985-10-01T00:00:00Z,-178.125,-87.5,.
1985-10-01T00:00:00Z,-176.875,-87.5,.
1985-10-01T00:00:00Z,-175.625,-87.5,.
1985-10-01T00:00:00Z,-174.375,-87.5,.
1985-10-01T00:00:00Z,-173.125,-87.5,.
1985-10-01T00:00:00Z,-171.875,-87.5,.
1985-10-01T00:00:00Z,-170.625,-87.5,.
1985-10-01T00:00:00Z,-169.375,-87.5,.
```

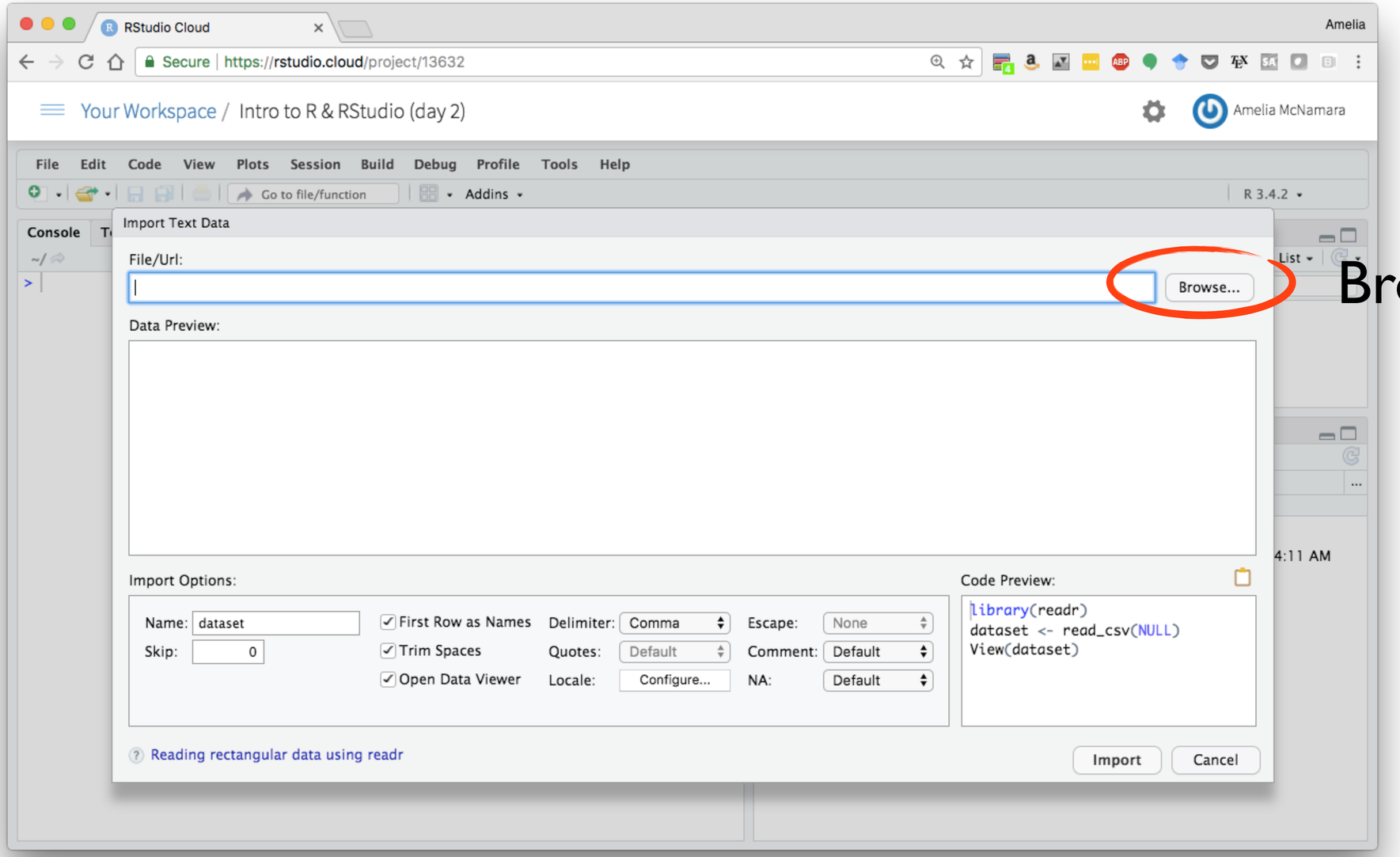
nimbus.csv

```
date,longitude,latitude,ozone
1985-10-01T00:00:00Z,-179.375,-87.5,.
1985-10-01T00:00:00Z,-178.125,-87.5,.
1985-10-01T00:00:00Z,-176.875,-87.5,.
1985-10-01T00:00:00Z,-175.625,-87.5,.
1985-10-01T00:00:00Z,-174.375,-87.5,.
1985-10-01T00:00:00Z,-173.125,-87.5,.
1985-10-01T00:00:00Z,-171.875,-87.5,.
1985-10-01T00:00:00Z,-170.625,-87.5,.
1985-10-01T00:00:00Z,-169.375,-87.5,.
```

Using the RStudio interface

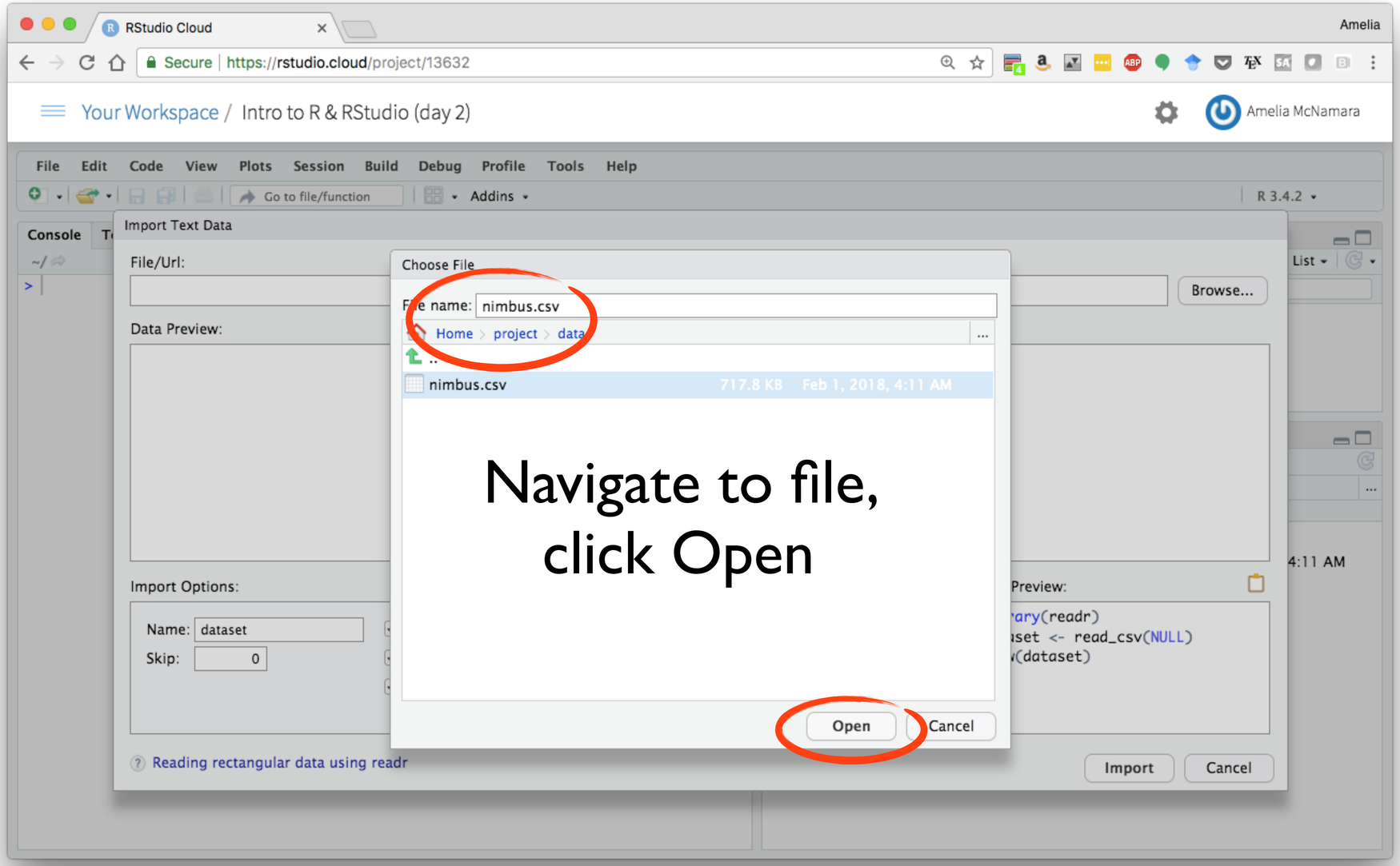


Using the RStudio interface

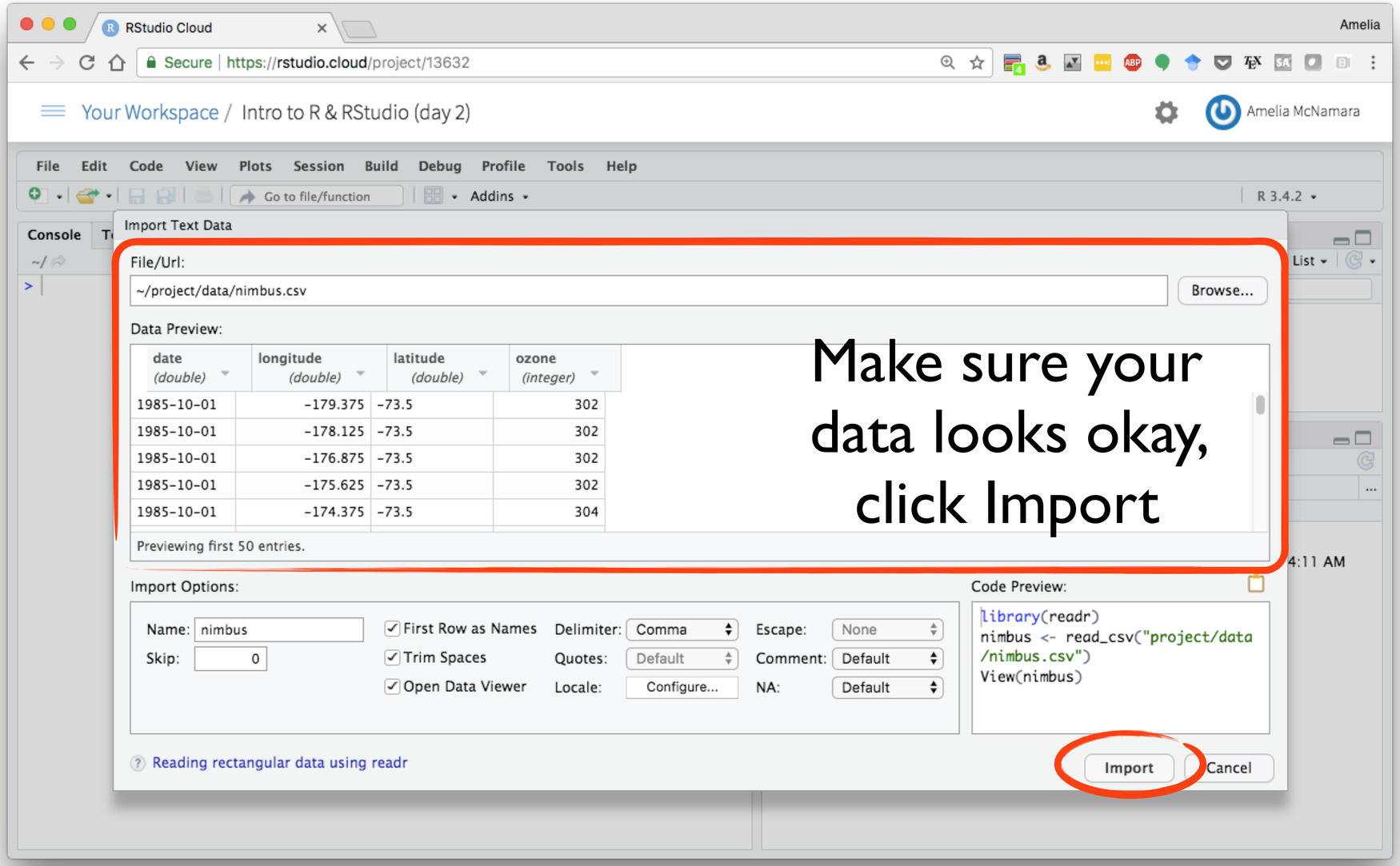


Browse files

Using the RStudio interface

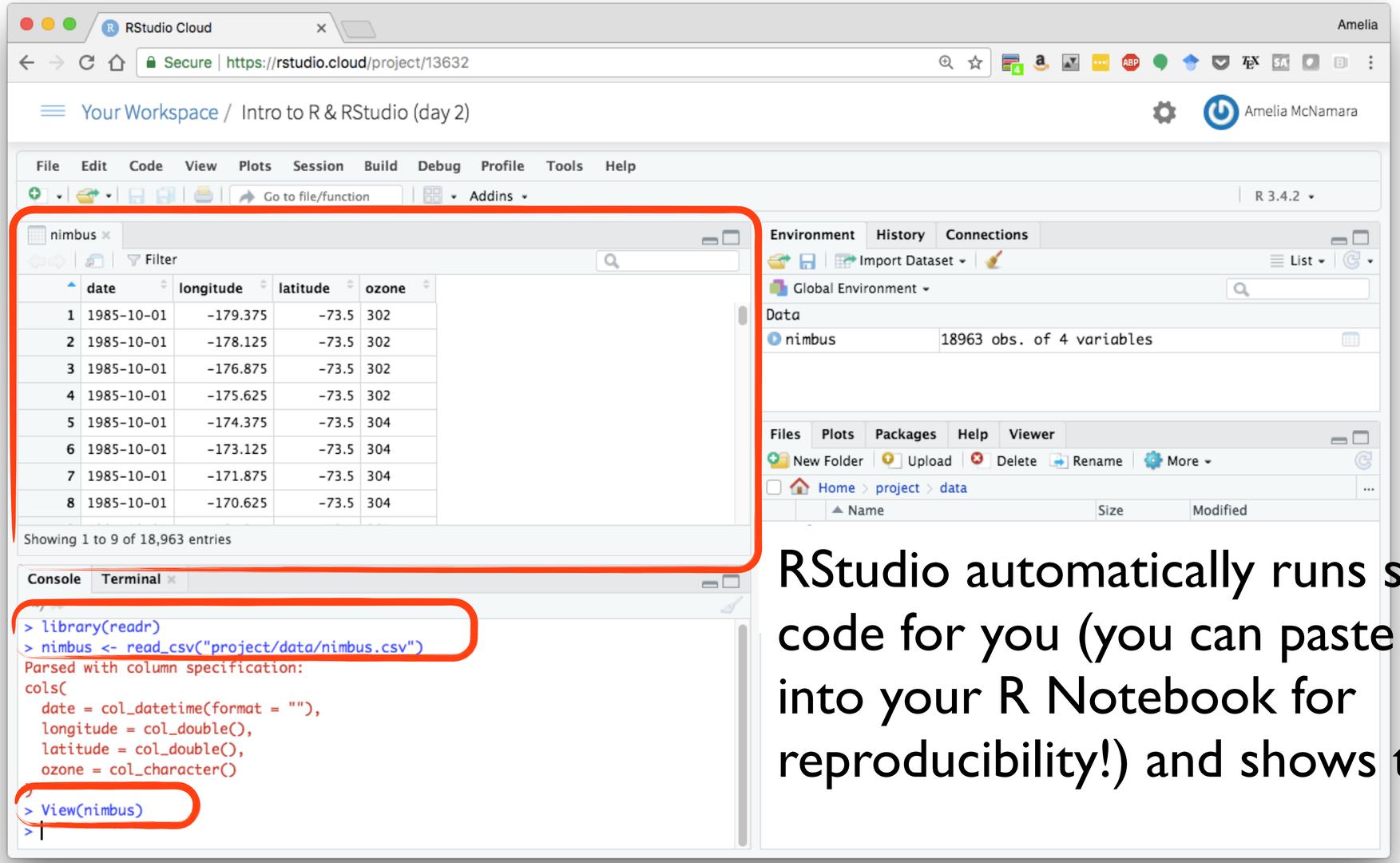


Using the RStudio interface



Using the RStudio interface

Another place where it looks like you got an error, but it's actually just a message



RStudio automatically runs some code for you (you can paste this into your R Notebook for reproducibility!) and shows the View

Your turn

Use the RStudio interface to import the GSS.csv dataset using readr

Your turn

Use the RStudio interface to import the GSS.csv dataset using readr

```
GSS <- read_csv("project/data/GSS.csv")
```

skimr

```
GSS %>%
     skim()
```

```
Skim summary statistics
 n obs: 2540
 n variables: 15

Variable type: character
              variable missing complete    n min max empty n_unique
                   Age       2     2538 2540   9  11     0       73
 ChildhoodFamilyIncome       2     2538 2540   7  17     0        7
           LaborStatus       2     2538 2540   5  16     0        9
         MaritalStatus       2     2538 2540   7  13     0        6
       OpinionOfIncome       2     2538 2540   7  17     0        7
        PoliticalParty       2     2538 2540   9  18     0       10
                  Race       2     2538 2540   5   5     0        3
      RespondentIncome       2     2538 2540   7  14     0       15
                   Sex       2     2538 2540   4   6     0        2
     SexualOrientation       2     2538 2540   8  27     0        6
     TotalFamilyIncome       2     2538 2540   7  14     0       14

Variable type: integer
               variable missing complete    n    mean     sd   p0     p25 median     p75
 HighestSchoolCompleted       3     2537 2540   13.7    3.07    0   12      14     16
                     ID       2     2538 2540 1271.22 734.76    1  635.25 1269.5 1908.75
            NumChildren      25     2515 2540    1.78   1.56    0    0       2      3
                   Year       2     2538 2540 2014      0    2014 2014    2014   2014
 p100     hist
   20 ▁▁▁▁▂▆▆▃▂
 2543 ▇▇▇▇▇▇▇▇
    7 ▆▂▃▆▂▁▁▁▁
 2014 ▁▁▁▁▁▇▁▁▁▁
```

Sometimes, you don't want all the summary statistics skimr gives you. You can use `skim_with()` to remove them

```
skim_with(integer = list(p25 = NULL, p75=NULL))
```

`skim_with()` can also be used to add additional summary statistics (e.g. a trimmed mean) but we won't discuss this

If you change your mind about your changes, use `skim_with_defaults()` to put them back

```
skim_with_defaults()
```

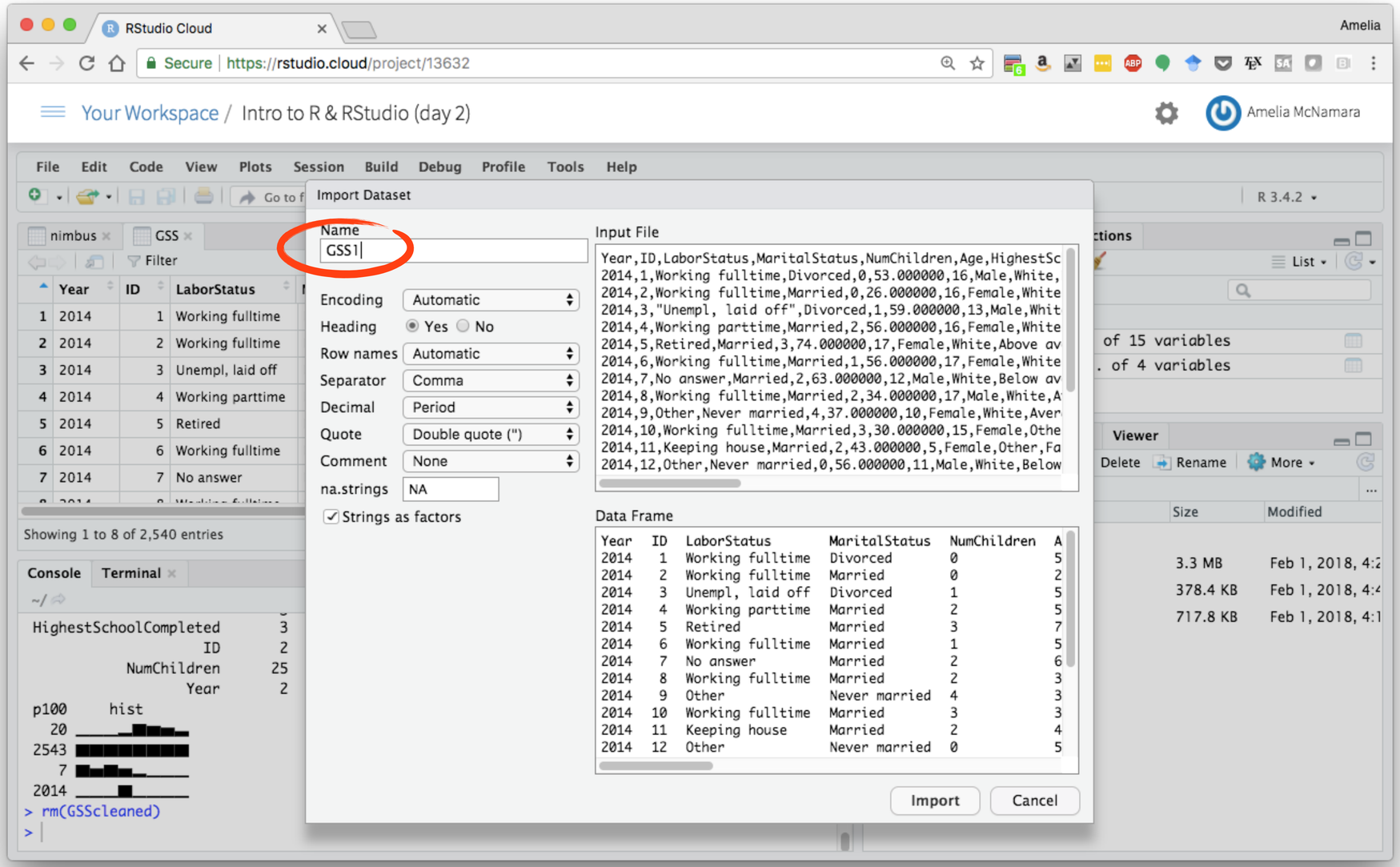
Your turn

Try skimming GSS and see how the output is different

read.csv()

Let's compare what the same data looks like after it's read in with `read.csv()`

Give it a different name to distinguish from our first dataset



Your turn

Try skimming GSS1 and see how the output is different from GSS

read.csv() vs. read_csv()

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 ]
>
```

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
>
```

read_csv()

Compared to read.table and its derivatives, readr functions are:

1. ~ 10 times faster
2. Return tibbles
3. Have more intuitive defaults. No row names, no strings as factors.

readr functions

function	reads
read_csv()	Comma separated values
read_csv2()	Semi-colon separated values
read_delim()	General delimited files
read_fwf()	Fixed width files
read_log()	Apache log files
read_table()	Space separated
read_tsv()	Tab delimited values

readr functions

function	reads
read_csv()	**Comma separated values**
read_csv2()	Semi-colon separated values
read_delim()	General delimited files
read_fwf()	Fixed width files
read_log()	Apache log files
read_table()	Space separated
read_tsv()	Tab delimited values

more programmatic

read_csv()

readr functions share a common syntax

```
df <- read_csv("path/to/file.csv", …)
```

object to save output into

path from working directory to file

Parsing

. = NA

```
nimbus
```

date <S3: POSIXct>	longitude <dbl>	latitude <dbl>	ozone <chr>
1985-10-01	-179.375	-87.5	.
1985-10-01	-178.125	-87.5	.
1985-10-01	-176.875	-87.5	.
1985-10-01	-175.625	-87.5	.
1985-10-01	-174.375	-87.5	.
1985-10-01	-173.125	-87.5	.
1985-10-01	-171.875	-87.5	.
1985-10-01	-170.625	-87.5	.

read_csv()

readr functions share a common syntax

```
nimbus <- read_csv("nimbus.csv", na = ".")
```

object to save output into

path from working directory to file

Value(s) to convert to NA

Your Turn

Reread in **nimbus.csv**. But this time convert the "."'s to NA's. How many NA's are in the ozone column?

Your Turn

Reread in **nimbus.csv**. But this time convert the "."'s to NA's. How many NA's are in the ozone column?

```
nimbus <- read_csv("nimbus.csv", na = ".")
View(nimbus)
nimbus %>%
 skim(ozone)
Skim summary statistics
 n obs: 18963
 n variables: 4

Variable type: integer
 variable missing complete     n   mean    sd  p0 p25 median p75 p100     hist
    ozone     155    18808 18963 299.41 42.61 180 272    296 332  417
```

Quiz

What "type" of column is ozone?

```
nimbus <- read_csv("nimbus.csv", na = ".")
```

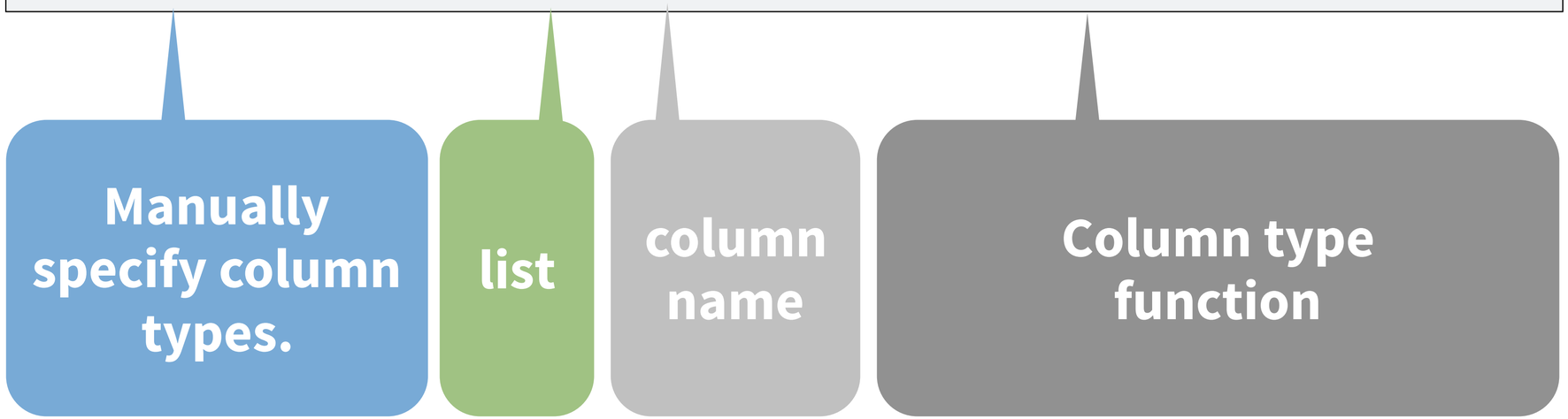
date <S3: POSIXct>	longitude <dbl>	latitude <dbl>	ozone <chr>
1985-10-01	-179.375	-87.5	*NA*
1985-10-01	-178.125	-87.5	*NA*
1985-10-01	-176.875	-87.5	*NA*
1985-10-01	-175.625	-87.5	*NA*
1985-10-01	-174.375	-87.5	*NA*
1985-10-01	-173.125	-87.5	*NA*
1985-10-01	-171.875	-87.5	*NA*
1985-10-01	-170.625	-87.5	*NA*
1985-10-01	-169.375	-87.5	*NA*
1985-10-01	-168.125	-87.5	*NA*

<chr> stands for character string (not a number)

read_csv()

readr functions share a common syntax

```
nimbus <- read_csv("project/data/nimbus.csv", na =
"."),
  col_types = list(ozone = col_double()))
```

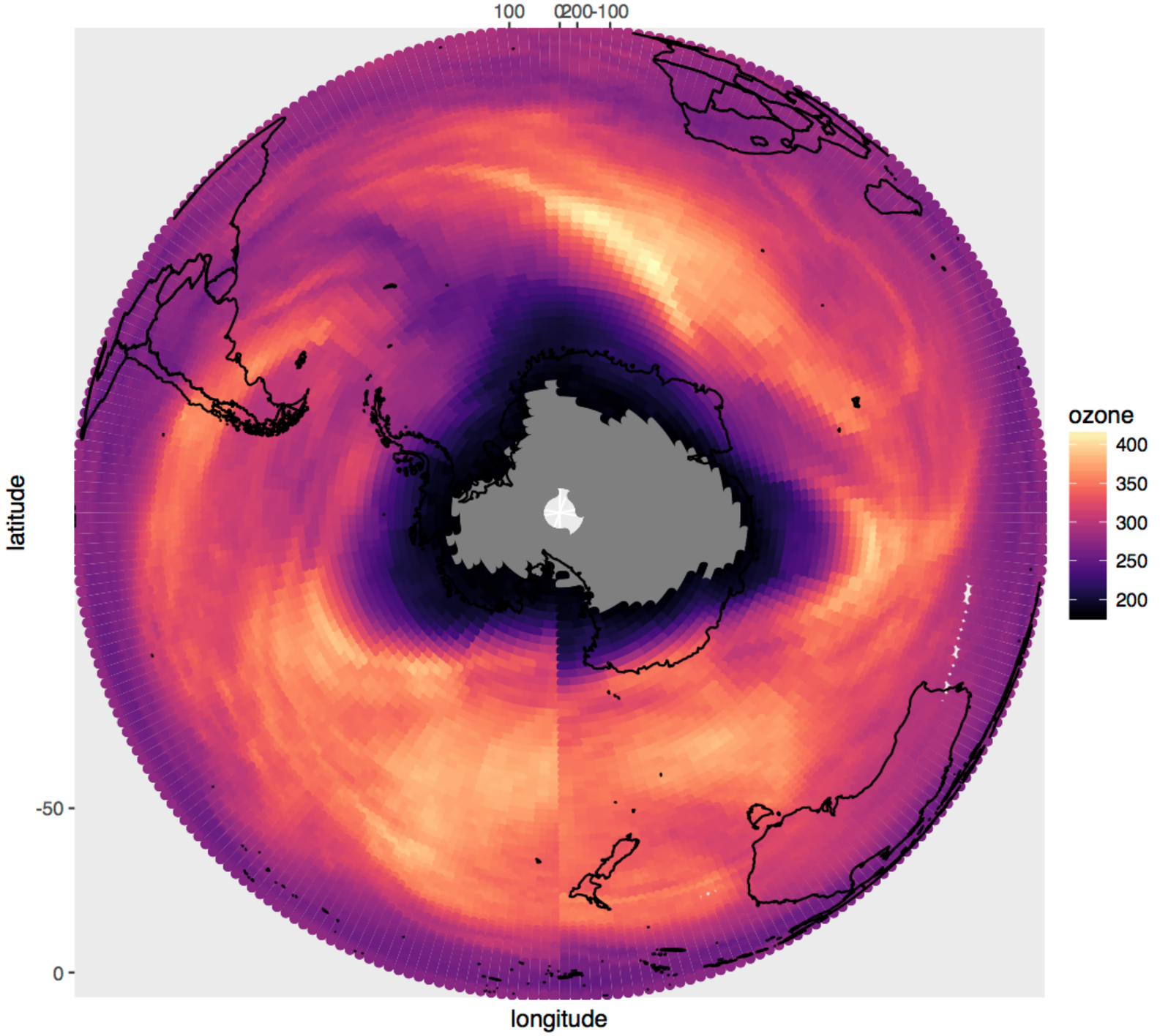


type function	data type
col_character()	character
col_date()	Date
col_datetime()	POSIXct (date-time)
col_double()	double (numeric)
col_factor()	factor
col_guess()	let readr guess (default)
col_integer()	integer
col_logical()	logical
col_number()	numbers mixed with non-number characters
col_numeric()	double or integer
col_skip()	do not read
col_time()	time

type function	data type
col_character()	character
col_date()	Date
col_datetime()	POSIXct (date-time)
col_double()	**double (numeric)**
col_factor()	factor
col_guess()	let readr guess (default)
col_integer()	integer
col_logical()	logical
col_number()	numbers mixed with non-number characters
col_numeric()	double or integer
col_skip()	do not read
col_time()	time

```
nimbus <- read_csv("nimbus.csv", na = ".",
  col_types = list(ozone = col_double()))

library(viridis)
world <- map_data(map = "world")
nimbus %>%
  ggplot() +
    geom_point(aes(longitude, latitude, color = ozone)) +
    geom_path(aes(long, lat, group = group), data = world) +
    coord_map("ortho", orientation=c(-90, 0, 0)) +
    scale_color_viridis(option = "A")
```


100
0
200
-100
latitude
-50
0
longitude
ozone
400
350
300
250
200

Writing

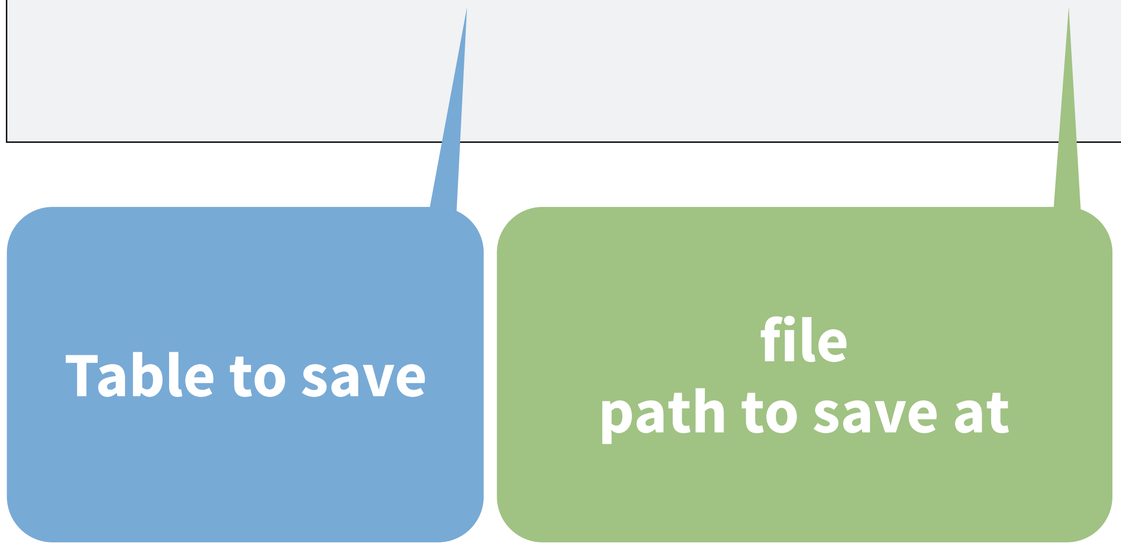
readr functions

function	writes
write_csv()	Comma separated values
write_excel_csv()	CSV intended for opening in Excel
write_delim()	General delimited files
write_file()	Single string, written as is
write_lines()	Vector of strings, one element per line
write_tsv()	Tab delimited values

write_csv()

Saves data set as a csv on your computer.

```
write_csv(nimbus, file = "nimbus2.csv")
```



Other types of data

package	accesses
haven	SPSS, Stata, and SAS files
readxl	excel files (.xls, .xlsx)
jsonlite	json
xml2	xml
httr	web API’s
rvest	web pages (web scraping)
DBI	databases
sparklyr	data loaded into spark

Spreadsheets

‹ All Collections

Collection idea for us?

Practical Data Science for Stats - a PeerJ Collection

Data Science | Statistics | Scientific Computing and Simulation | Computer Education | Computational Science | Social Computing | Software Engineering | Science and Medical Education | Computational Biology | Human-Computer Interaction | Anthropology | Programming Languages | Visual Analytics | Graphics | Data Mining and Machine Learning

Practical Data Science for Stats

The "Practical Data Science for Stats" Collection contains preprints focusing on the practical side of data science workflows and statistical analysis. Curated by Jennifer Bryan and Hadley Wickham.

Karl Broman, Kara Woo. Data organization in spreadsheets. PeerJ preprint and The American Statistician.
https://peerj.com/preprints/3183/

Abstract: Spreadsheets are widely used software tools for data entry, storage, analysis, and visualization. Focusing on the data entry and storage aspects, this paper offers practical recommendations for organizing spreadsheet data to reduce errors and ease later analyses. The basic principles are: be consistent, write dates like YYYY-MM-DD, don't leave any cells empty, put just one thing in a cell, organize the data as a single rectangle (with subjects as rows and variables as columns, and with a single header row), create a data dictionary, don't include calculations in the raw data files, don't use font color or highlighting as data, choose good names for things, make backups, use data validation to avoid data entry errors, and save the data in plain text file.

Your Turn (by request)

Find a dataset on your computer (or the internet) that you would like to import. Use the appropriate function to load it in, and skim() to see if it's being treated appropriately. Ask questions if you get stuck!