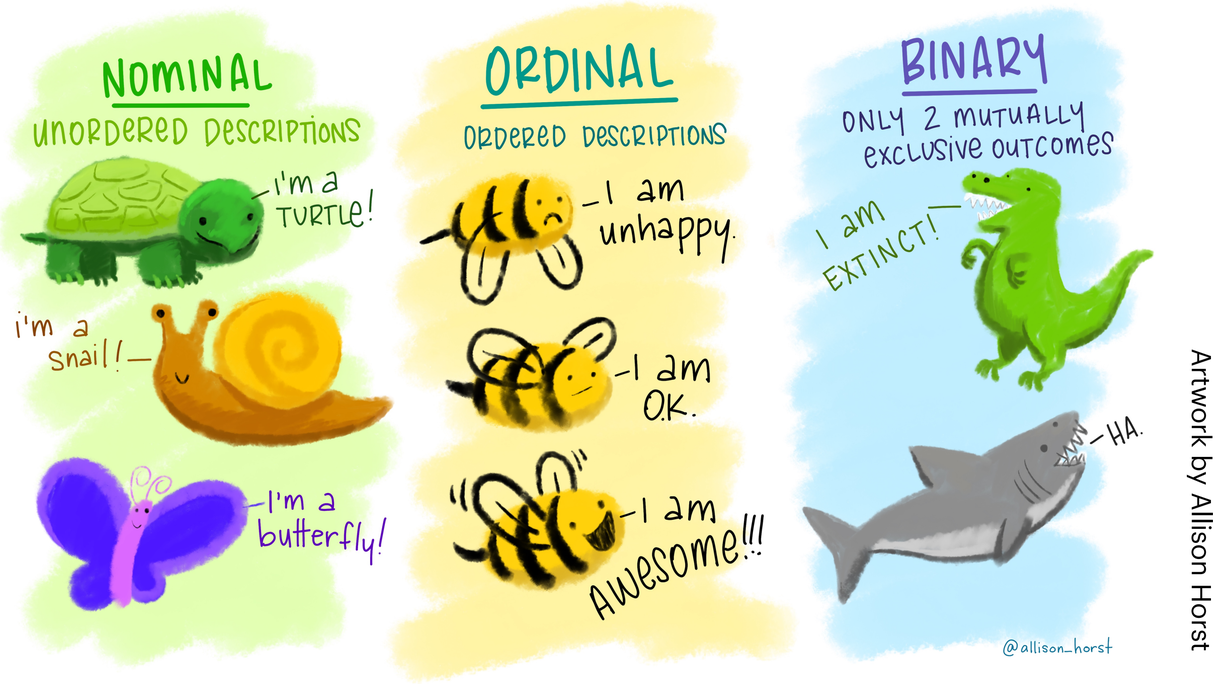
Data types and structures

Week 3 (10/15/25)



Artwork by Allison Horst

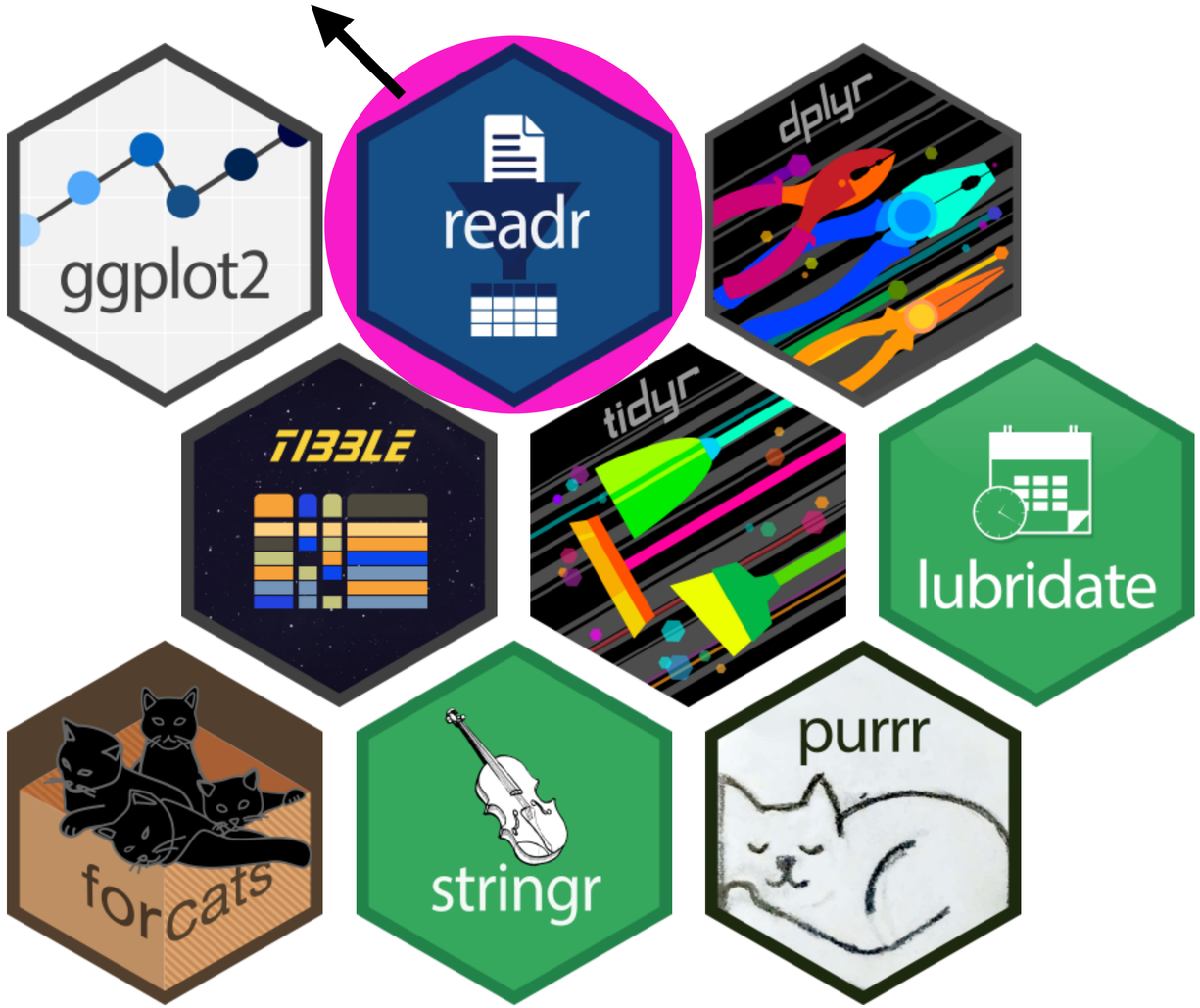
Stepfanie M. Aguillon

Outline of today's class

- Introduction to loading data into R
- Data types and data structures
- Continue practicing with RStudio/GitHub and Quarto

Loading data into R

`tidyverse` package
for loading data into R



Loading data with `readr`

1. download the `ratdat_combined.csv` file from GitHub (in the `datasets` folder) and put it in your R Project for the course

2. pull from GitHub, then start a new script or Quarto document and load the `tidyverse`

```
library(tidyverse)
```

Loading data with `readr`

3 decide which `readr` command to use for the dataset

`read_tsv()`	TSV files (“tab-separated values” file)
`read_csv()`	CSV files (“comma-separated values” file)
`read_delim()`	any delimited files (need to specify the delimitation)

Loading data with readr

Data import with the tidyverse : : CHEATSHEET

Read Tabular Data with readr

read_*(file, col_names = TRUE, col_types = NULL, col_select = NULL, id = NULL, locale, n_max = Inf, skip = 0, na = c("", "NA"), guess_max = min(1000, n_max), show_col_types = TRUE) See **?read_delim**

A	B	C
1	2	3
4	5	NA

read_delim("file.txt", delim = "|"**)** Read files with any delimiter. If no delimiter is specified, it will automatically guess.
To make file.txt, run: write_file("A|B|C\n1|2|3\n4|5|NA", file = "file.txt")

A	B	C
1	2	3
4	5	NA

read_csv("file.csv"**)** Read a comma delimited file with period decimal marks.
write_file("A,B,C\n1,2,3\n4,5,NA", file = "file.csv")

A	B	C
1.5	2	3
4.5	5	NA

read_csv2("file2.csv"**)** Read semicolon delimited files with comma decimal marks.
write_file("A;B;C\n1,5;2;3\n4,5;5;NA", file = "file2.csv")

A	B	C
1	2	3
4	5	NA

read_tsv("file.tsv"**)** Read a tab delimited file. Also **read_table()**.
read_fwf("file.tsv", fwf_widths(c(2, 2, NA))**)** Read a fixed width file.
write_file("A\tB\tC\n1\t2\t3\n4\t5\tNA\n", file = "file.tsv")

USEFUL READ ARGUMENTS

A	B	C
1	2	3
4	5	NA

No header
read_csv("file.csv", col_names = FALSE)

x	y	z
A	B	C
1	2	3
4	5	NA

Provide header
read_csv("file.csv",
col_names = c("x", "y", "z"))

Read multiple files into a single table
read_csv(c("f1.csv", "f2.csv", "f3.csv"),
id = "origin_file")

1	2	3
4	5	NA

Skip lines
read_csv("file.csv", skip = 1)

A	B	C
1	2	3

Read a subset of lines
read_csv("file.csv", n_max = 1)

A	B	C
NA	2	3
4	5	NA

Read values as missing
read_csv("file.csv", na = c("1"))

Specify decimal marks
read_delim("file2.csv", locale =
locale(decimal_mark = ","))

Save Data with readr

write_*(x, file, na = "NA", append, col_names, quote, escape, eol, num_threads, progress)

A	B	C
1	2	3
4	5	NA

write_delim(x, file, delim = " "**)** Write files with any delimiter.

write_csv(x, file**)** Write a comma delimited file.

write_csv2(x, file**)** Write a semicolon delimited file.

write_tsv(x, file**)** Write a tab delimited file.

One of the first steps of a project is to import outside data into R. Data is often stored in tabular formats, like csv files or spreadsheets.

The front page of this sheet shows how to import and save text files into R using **readr**.

The back page shows how to import spreadsheet data from Excel files using **readxl** or Google Sheets using **googlesheets4**.

OTHER TYPES OF DATA

Try one of the following packages to import other types of files:

- **haven** - SPSS, Stata, and SAS files
- **DBI** - databases
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)
- **readr::read_lines()** - text data

Column Specification with readr

Column specifications define what data type each column of a file will be imported as. By default readr will generate a column spec when a file is read and output a summary.

spec(x**)** Extract the full column specification for the given imported data frame.

```
spec(x)
# cols(
#   age = col_integer(),
#   edu = col_character(),
#   earn = col_double()
# )
```

age is an integer
edu is a character
earn is a double (numeric)

COLUMN TYPES

Each column type has a function and corresponding string abbreviation.

- **col_logical()** - "l"
- **col_integer()** - "i"
- **col_double()** - "d"
- **col_number()** - "n"
- **col_character()** - "c"
- **col_factor(**levels, ordered = FALSE**)** - "f"
- **col_datetime(**format = ""**)** - "T"
- **col_date(**format = ""**)** - "D"
- **col_time(**format = ""**)** - "t"
- **col_skip()** - "-", "_"
- **col_guess()** - "?"

USEFUL COLUMN ARGUMENTS

Hide col spec message
read_*(file, show_col_types = FALSE)

Select columns to import
Use names, position, or selection helpers.
read_*(file, col_select = c(age, earn))

Guess column types
To guess a column type, read_ *() looks at the first 1000 rows of data. Increase with **guess_max**.
read_*(file, guess_max = Inf)

DEFINE COLUMN SPECIFICATION

Set a default type

```
read_csv(
  file,
  col_type = list(.default = col_double())
)
```

Use column type or string abbreviation

```
read_csv(
  file,
  col_type = list(x = col_double(), y = "l", z = "_")
)
```

Use a single string of abbreviations

```
# col types: skip, guess, integer, logical, character
read_csv(
  file,
  col_type = "_?ilc"
)
```

CC BY SA Posit Software, PBC • info@posit.co • posit.co • **readr.tidyverse.org** • **readxl.tidyverse.org** • **googlesheets4.tidyverse.org** • HTML cheatsheets at **pos.it/cheatsheets** • readxl 1.4.5 • googlesheets4 1.1.1 • Updated: 2025-08

https://rstudio.github.io/cheatsheets/data-import.pdf

Loading data with `readr`

3 decide which `readr` command to use for the dataset

`read_tsv()`	TSV files (“tab-separated values” file)
`read_csv()`	CSV files (“comma-separated values” file)
`read_delim()`	any delimited files (need to specify the delimitation)

Loading data with `readr`

3 decide which `readr` command to use for the dataset

`read_tsv()` ← TSV files ("tab-separated values" file)

`read_csv()` ← CSV files ("comma-separated values" file)

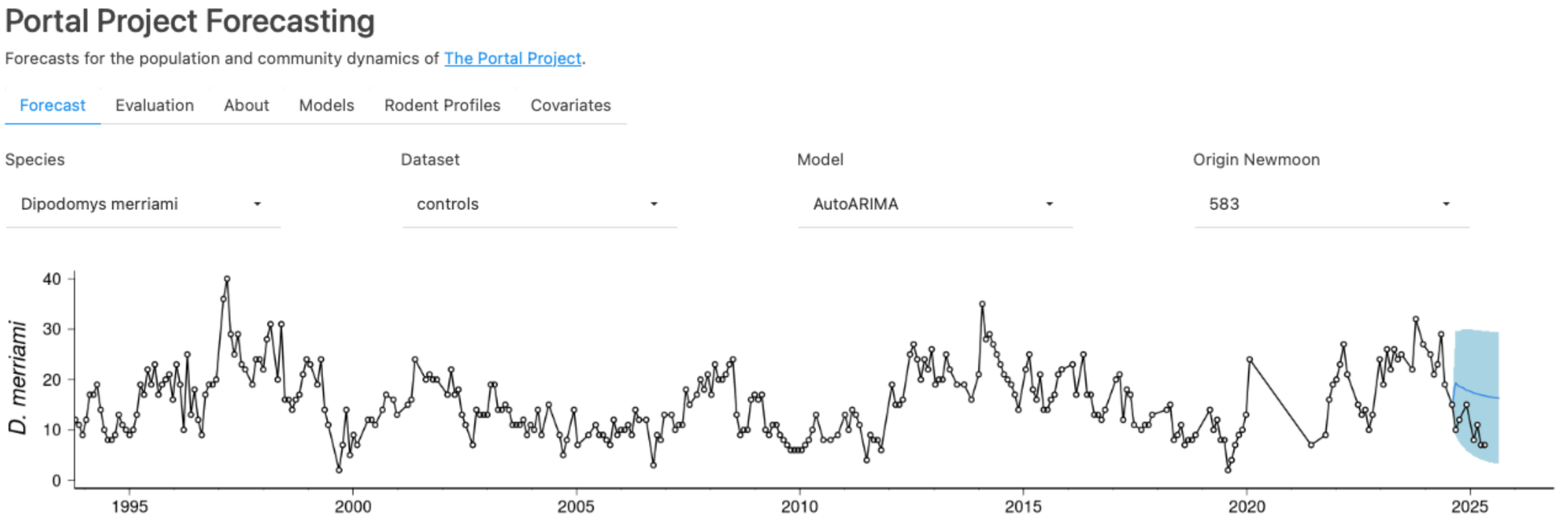
`read_delim()` ← any delimited files (need to specify the delimitation)

4

```
ratdat <- read_csv("ratdat_combined.csv")
```

The `ratdat` dataset

The Portal Project is a long-term ecological study being conducted near Portal, AZ. Since 1977, the site has been used to study the interactions among rodents, ants and plants and their respective responses to climate. To study the interactions among organisms, we experimentally manipulate access to 24 study plots. This study has produced over 100 scientific papers and is one of the longest running ecological studies in the U.S.



https://portal.weecology.org/

Many other ways to load data into R!

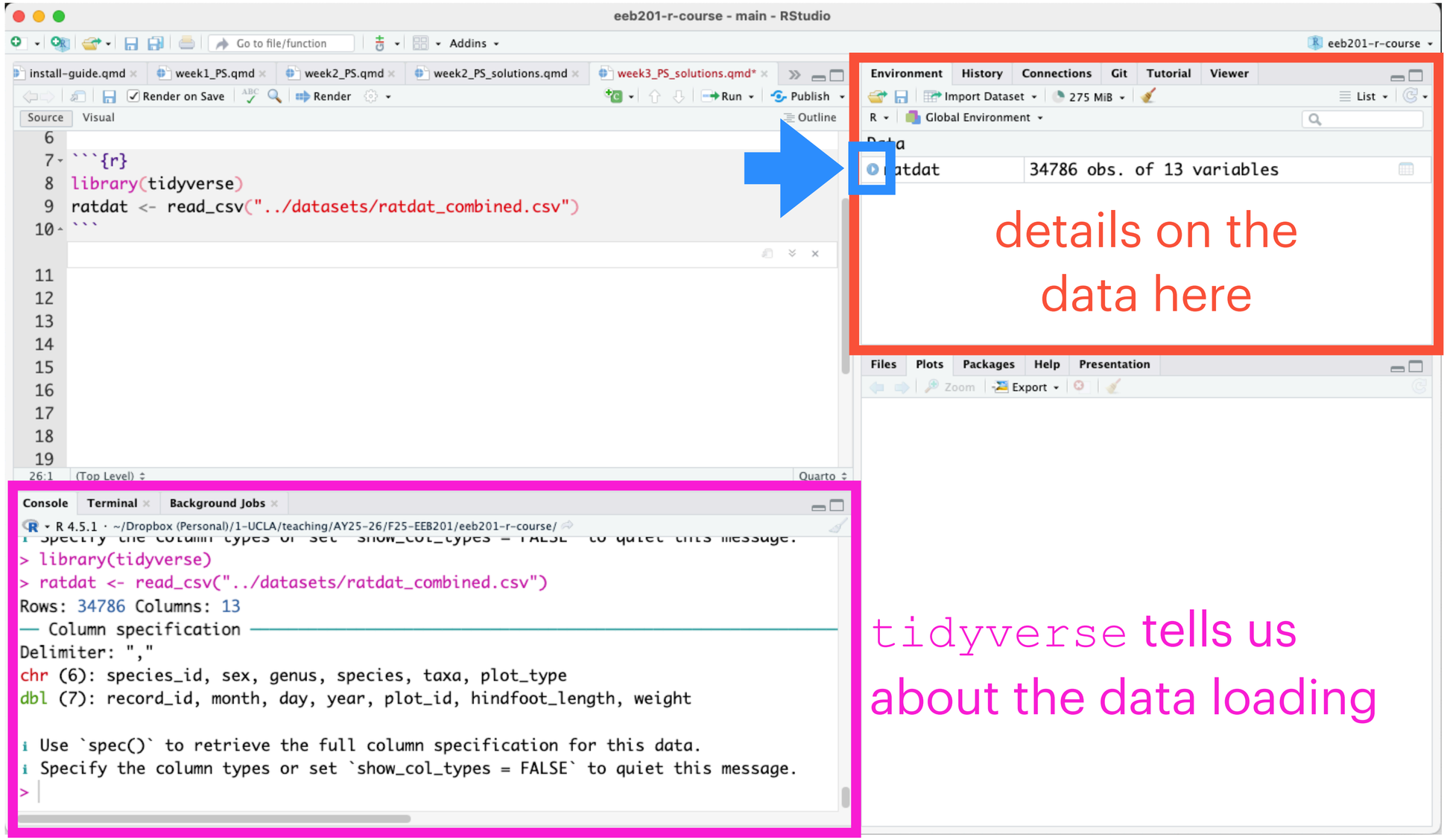
Excel files

Google sheets

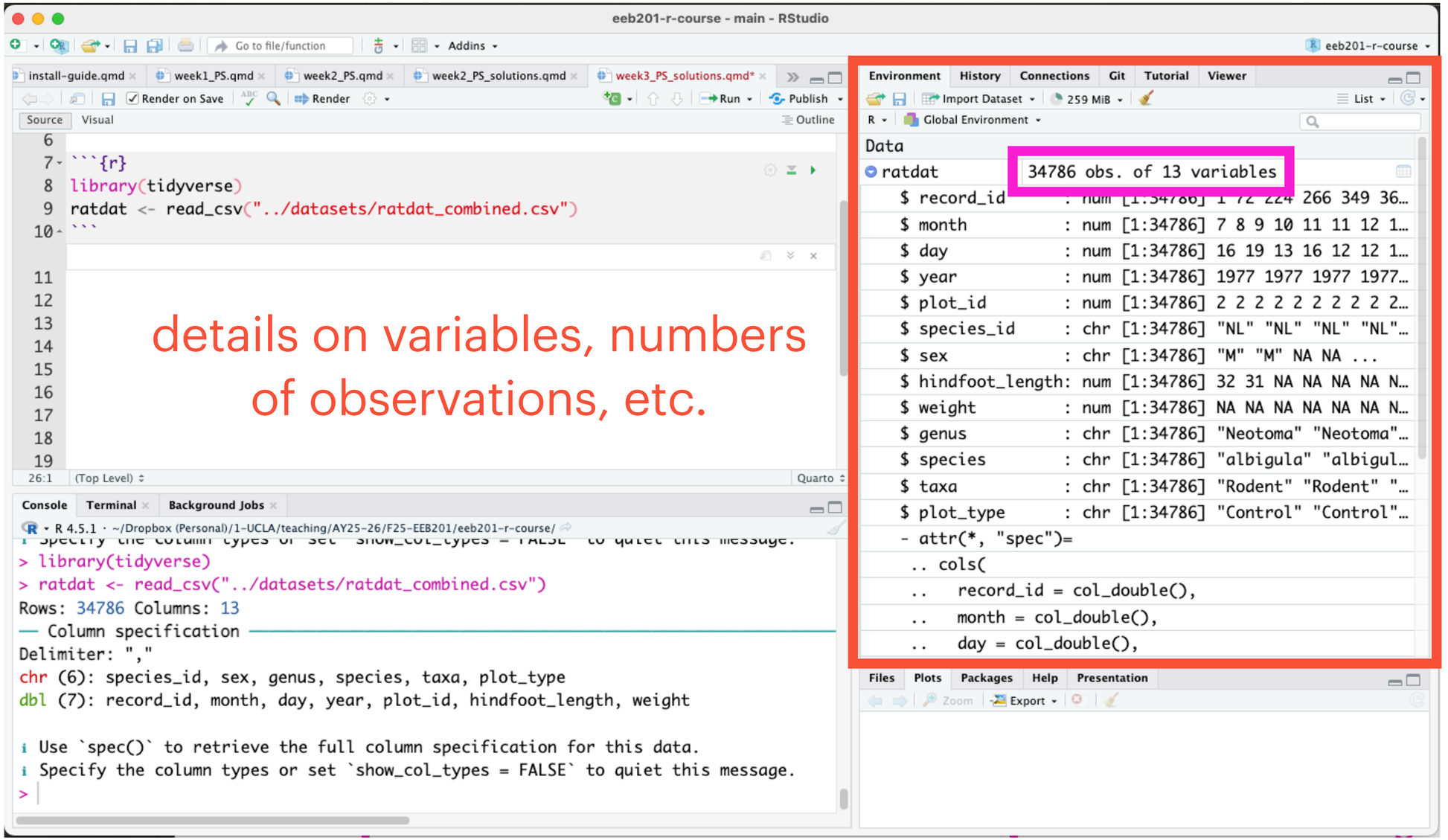
Google Drive

Website scraping

Use the environment panel to see details

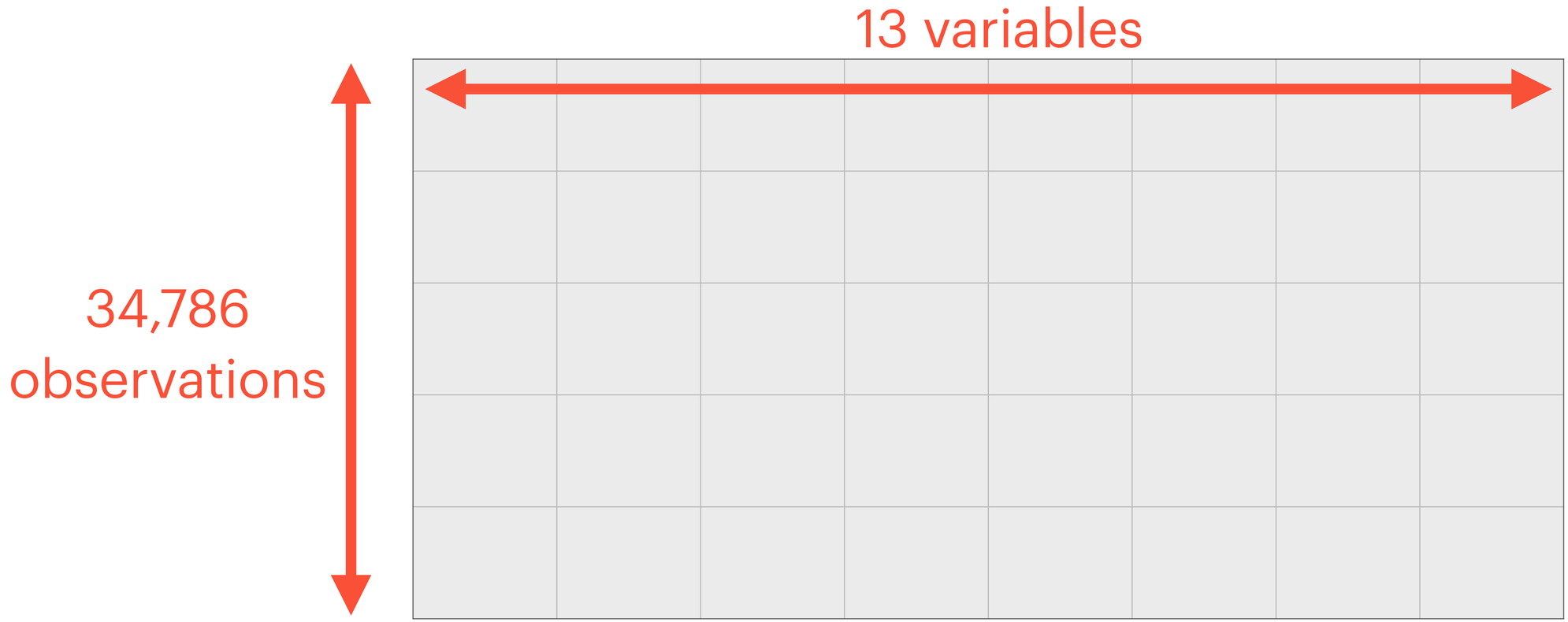


Use the environment panel to see details

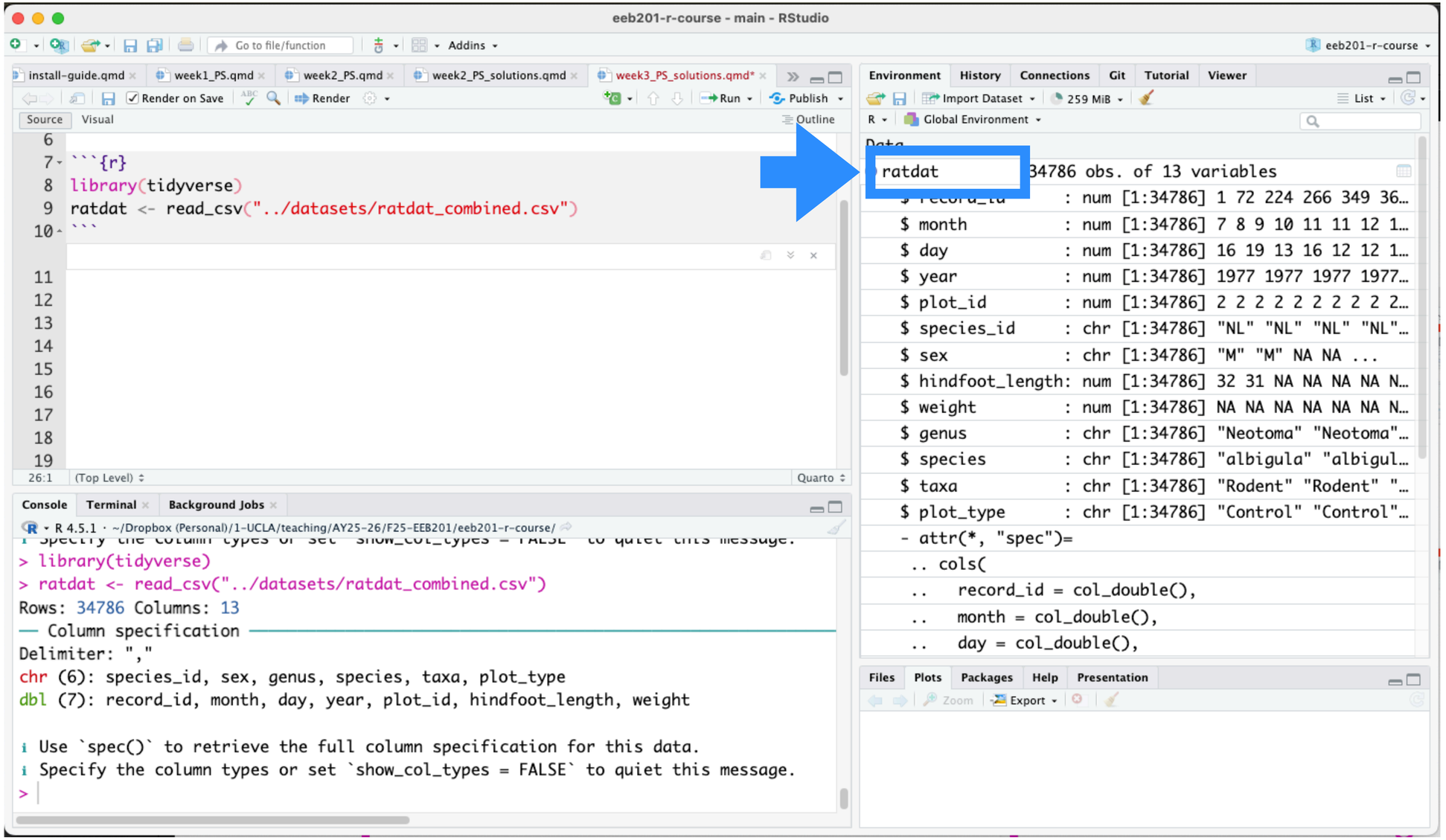


Use the environment panel to see details

`ratdat` dataset contains 34786 obs. of 13 variables



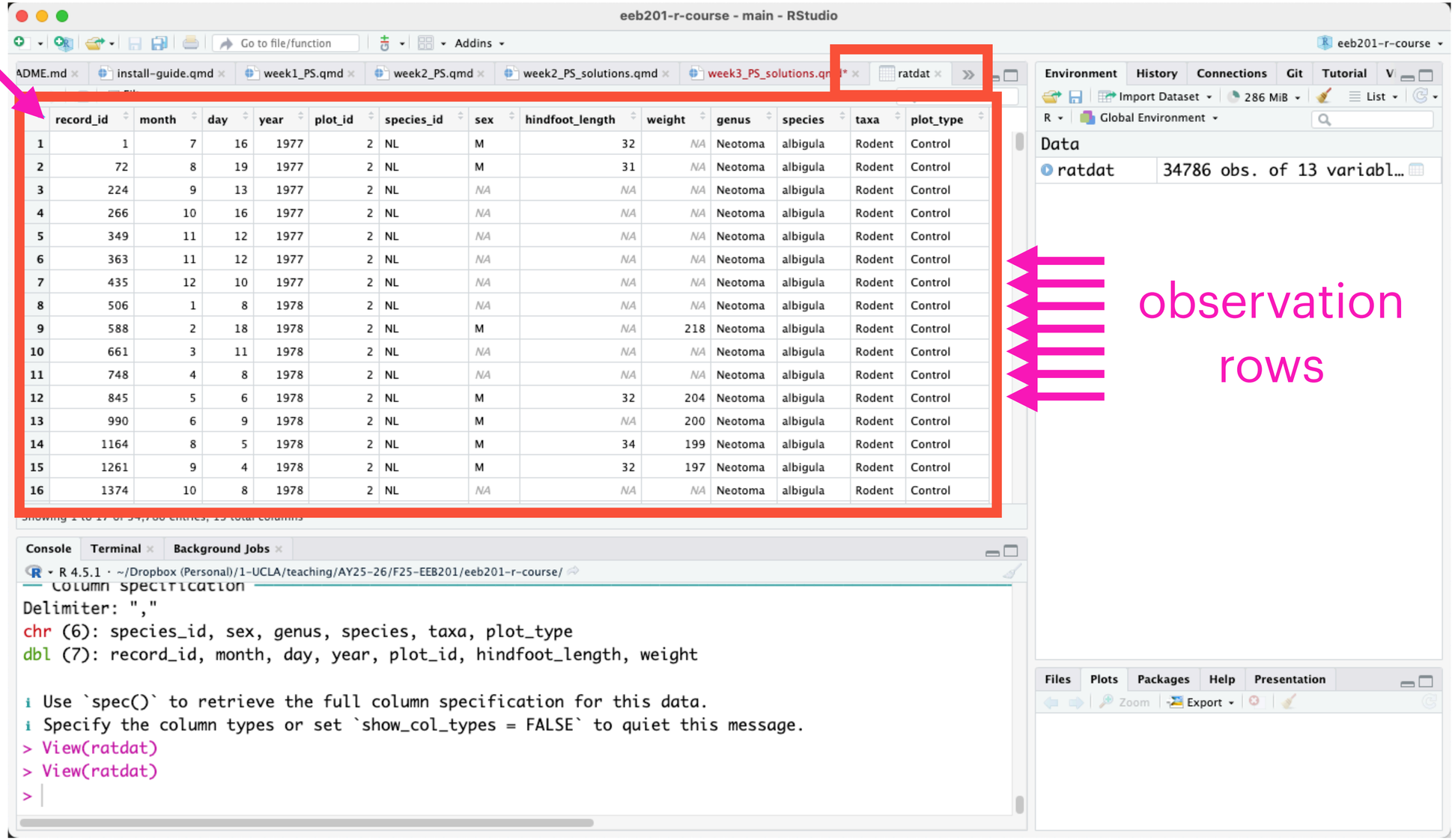
Use the environment panel to see details



live coding

Use the environment panel to see details

variable columns



	record_id	month	day	year	plot_id	species_id	sex	hindfoot_length	weight	genus	species	taxa	plot_type
1	1	7	16	1977	2	NL	M	32	NA	Neotoma	albigula	Rodent	Control
2	72	8	19	1977	2	NL	M	31	NA	Neotoma	albigula	Rodent	Control
3	224	9	13	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
4	266	10	16	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
5	349	11	12	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
6	363	11	12	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
7	435	12	10	1977	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
8	506	1	8	1978	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
9	588	2	18	1978	2	NL	M	NA	218	Neotoma	albigula	Rodent	Control
10	661	3	11	1978	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
11	748	4	8	1978	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control
12	845	5	6	1978	2	NL	M	32	204	Neotoma	albigula	Rodent	Control
13	990	6	9	1978	2	NL	M	NA	200	Neotoma	albigula	Rodent	Control
14	1164	8	5	1978	2	NL	M	34	199	Neotoma	albigula	Rodent	Control
15	1261	9	4	1978	2	NL	M	32	197	Neotoma	albigula	Rodent	Control
16	1374	10	8	1978	2	NL	NA	NA	NA	Neotoma	albigula	Rodent	Control

observation rows

Data types

Types of data in R: what are they?

Numeric	`1.25, 7, 10.5, 3.14159`
Integer	`1, 2, 3, 4, 5`
Character	`"Jan", "Feb", "Mar"`
Logical	`TRUE, FALSE`

live coding

Checking the data type

You can use the “concatenate” function `c()` to create these vectors

```
x <- c(1, 2, 3, 3.5)
```

You can use the `class()` to check the type of data you have

```
class(x)
```

live coding

Checking the data type

You can use the "concatenate" function `c()` to create these vectors

```
x <- c(1, 2, 3, 3.5)
```

```
y <- c("plot1", "plot2", "plot3")
```

You can use the `class()` to check the type of data you have

```
class(x)
```

```
class(y)
```

Changing between types of data

Two useful functions are `as.numeric()` and `as.character()`

```
x <- c(1, 2, 3, 3.5)

class(x)
```

```
[1] "numeric"
```

```
x_new <- as.character(x)

class(x_new)
```

```
[1] "character"
```

You can also do all of this within a dataset!

Let's try using `class()` within `ratdat`

```
class(ratdat$year)
```

the `$` operator lets us select particular variables from a dataset!

`dataset$variable`

You can also do all of this within a dataset!

Let's try using `class()` within `ratdat`

```
class(ratdat$year)
```

```
[1] "numeric"
```

Now try changing this variable with `as.character()`

```
ratdat$year <- as.character(ratdat$year)

class(ratdat$year)
```

```
[1] "character"
```

Types of data in R: what are they?

Numeric	`1.25, 7, 10.5, 3.14159`
Integer	`1, 2, 3, 4, 5`
Character	`"Jan", "Feb", "Mar"`
Logical	`TRUE, FALSE`

Characters vs. Factors

Character	`"Jan", "Feb", "Mar"`	Categorical variables
Factor	`"Jan", "Feb", "Mar"`	Categorical variables with a known set of possible values ("Jan" ... "Dec")

Change a variable to a factor with `as.factor()`

```
ratdat$month <- as.factor(ratdat$month)

class(ratdat$month)
```

```
[1] "factor"
```

Characters vs. Factors

View the possible values of your factor with `levels()`

```
levels(ratdat$month)

[1] "1"  "2"  "3"  "4"  "5"  "6"
"7"  "8"  "9"  "10" "11" "12"
```

** Having issues with a function that *should* work with a categorical variable, but isn't? The type of categorical variable is a place to start troubleshooting! Sometimes you need a factor but have a character (or vice versa). **

Data structures

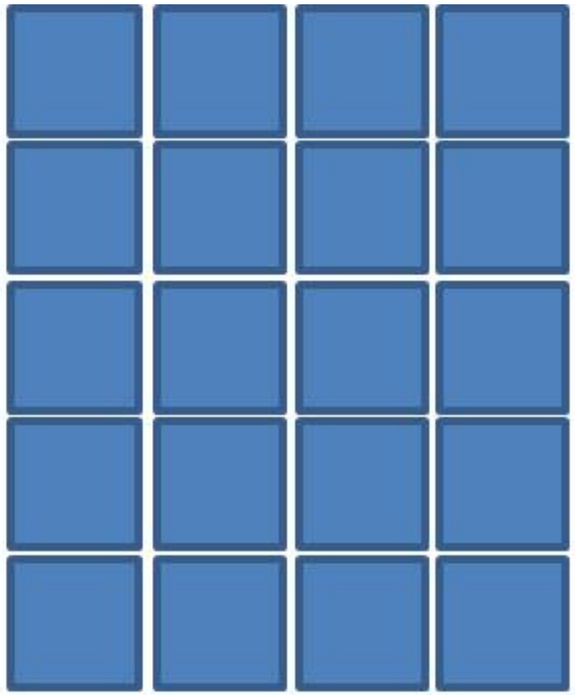
Data structures in R: what are they?

Vectors

A set of numbers or characters (1D)

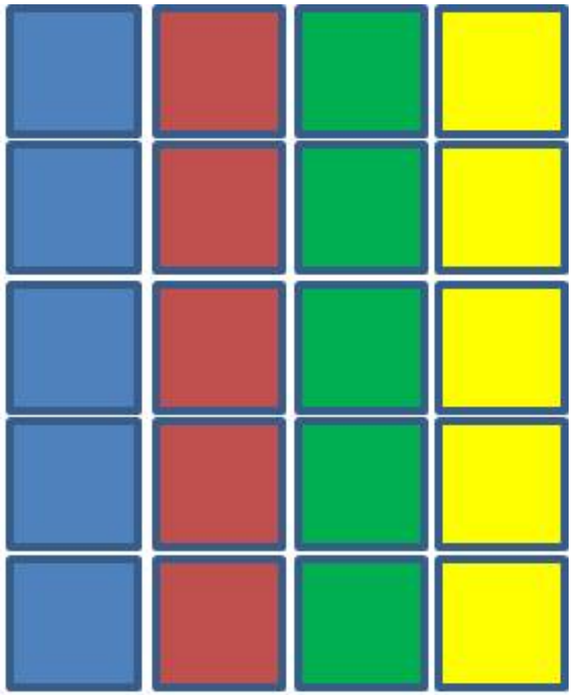
Matrix

Multiple vectors of the same type (2D)

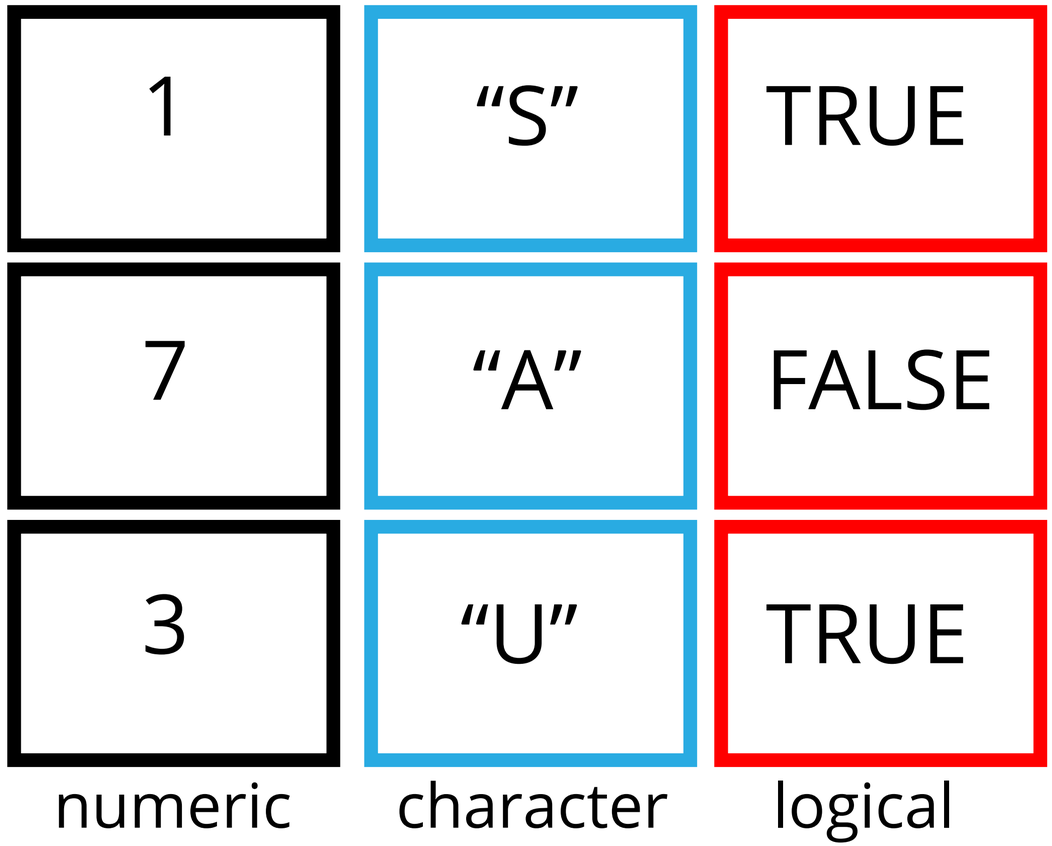


Data Frames

Multiple vectors of different types (2D)



Data structures in R: what are they?



Checking the data structure

There are a few functions to check the data structure

```
is.vector(ratdat)
```

```
[1] FALSE
```

```
is.matrix(ratdat)
```

```
[1] FALSE
```

```
is.data.frame(ratdat)
```

```
[1] TRUE
```

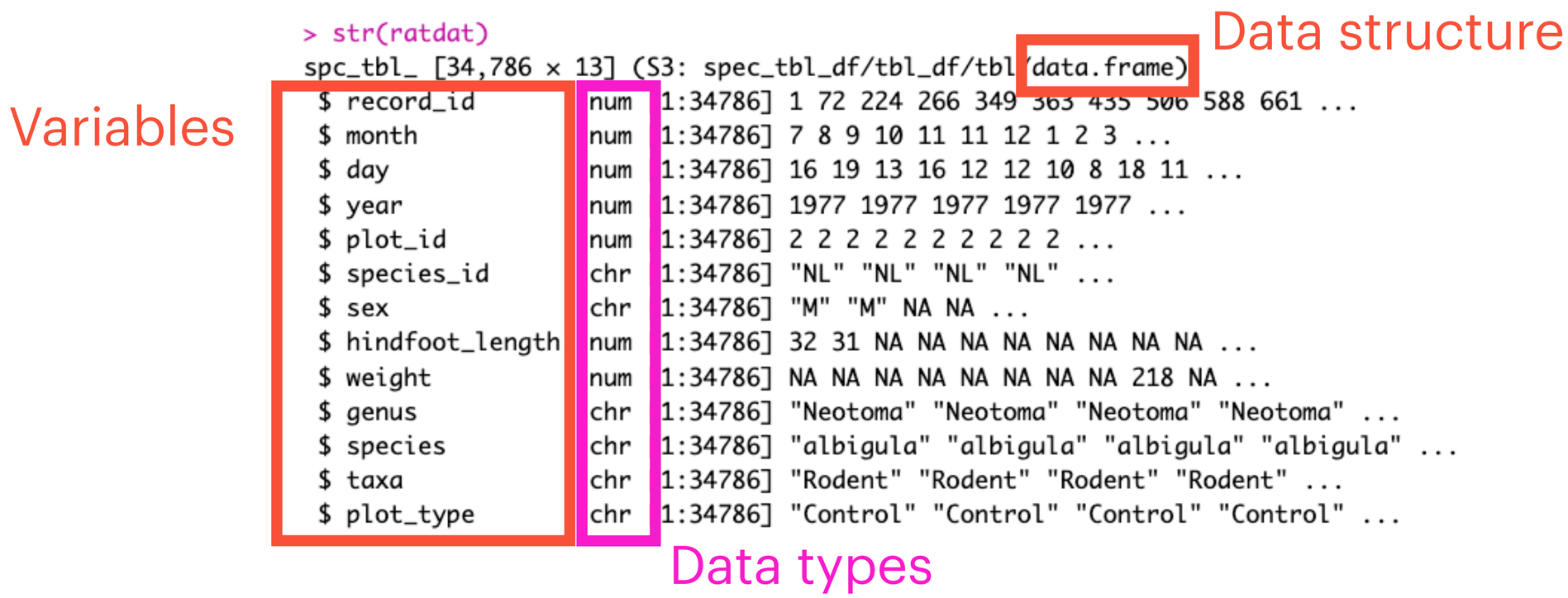
*You can move between data frames and matrices just like with different data types using `as.data.frame()` or `as.matrix()`

Data types AND data structures

Checking your data using `str()`

`str()` is a useful function to check data types and "structures" all at once

```
str(ratdat)
```



Checking your data with `tidyverse`

When importing a dataset with `readr`, you'll get info about the data

"chr" = character

"dbl" = double (numeric)

```
Rows: 34786 Columns: 13
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (6): species_id, sex, genus, species, taxa, plot_type
dbl (7): record_id, month, day, year, plot_id, hindfoot_length, weight
```

`spec()` will also give you similar "specification" details

```
spec(ratdat)
```

```
cols(
  record_id = col_double(),
  month = col_double(),
  day = col_double(),
  year = col_double(),
  plot_id = col_double(),
  species_id = col_character(),
```

Practice, practice, practice...

Some new Quarto concepts to try out!

1. Add links to a Quarto document

```
<link_url>
```

```
[display text](link_url)
```

2. Code chunk options in Quarto documents

````
```{r}
#| eval: false
a <- 1 + 1
a
```
````

````
```{r}
#| echo: false
b <- 1 + 1
b
```
````

````
```{r}
#| include: false
c <- 1 + 1
c
```
````

````
```{r}
#| error: true
d < 1 + 1
d
```
````

````
```{r}
#| message: false
library(tidyverse)
```
````

````
```{r}
#| warning: false
# try this with
# code from PS2!
```
````

https://r4ds.hadley.nz/quarto.html#chunk-options

Some new Quarto concepts to try out!

`#| eval: true/false`	should code be evaluated?
`#| echo: true/false`	should code be included?
`#| include: true/false`	should code or output be included?
`#| error: true/false`	should errors be included? (& not halt the doc?)
`#| message: true/false`	should messages be included?
`#| warning: true/false`	should warnings be included?

https://r4ds.hadley.nz/quarto.html#chunk-options

Further reading

- Introduction to readr
- Data Import with the `tidyverse` Cheatsheet