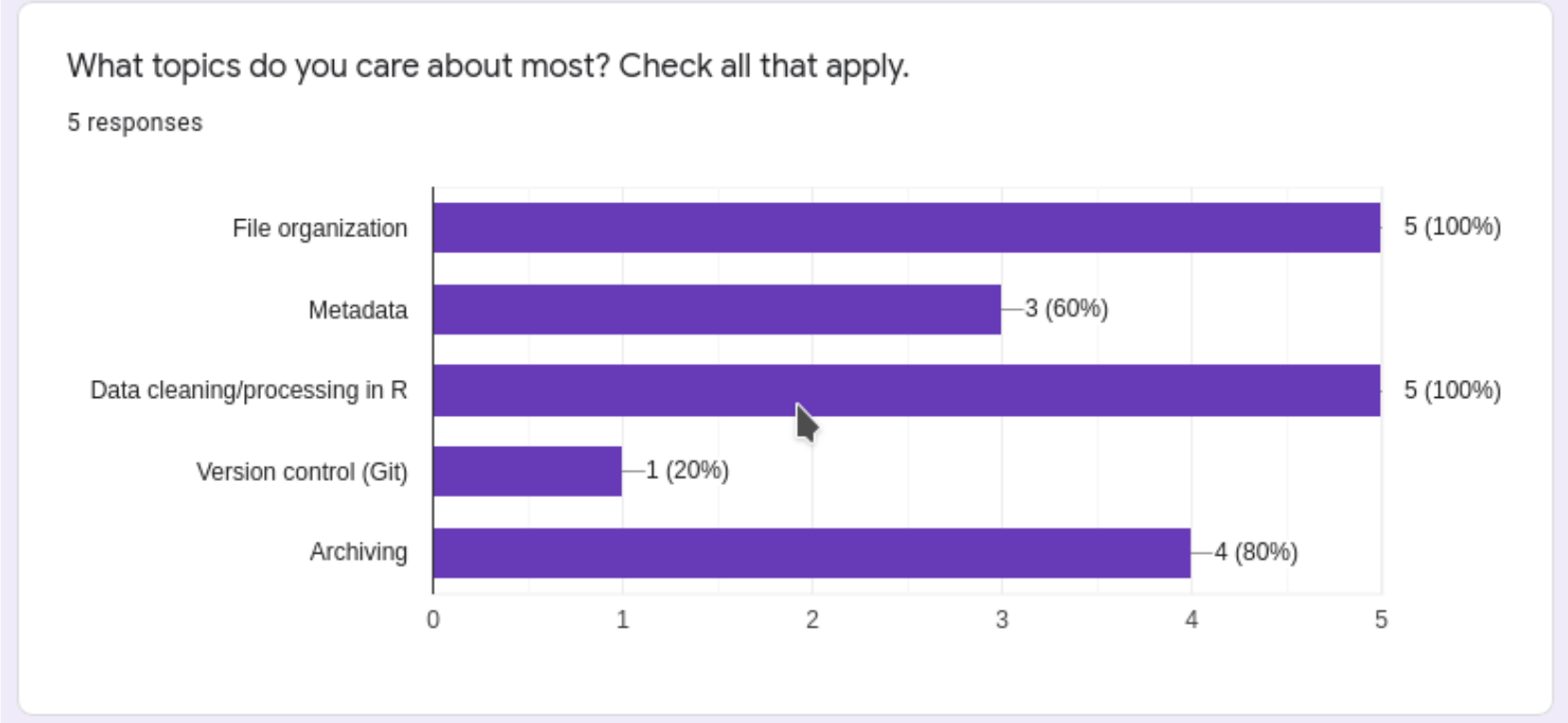
File Organization & Data Cleaning in R

Isabella Richmond
October 12, 2021 [updated: October 11 2021]

What we are learning today & why

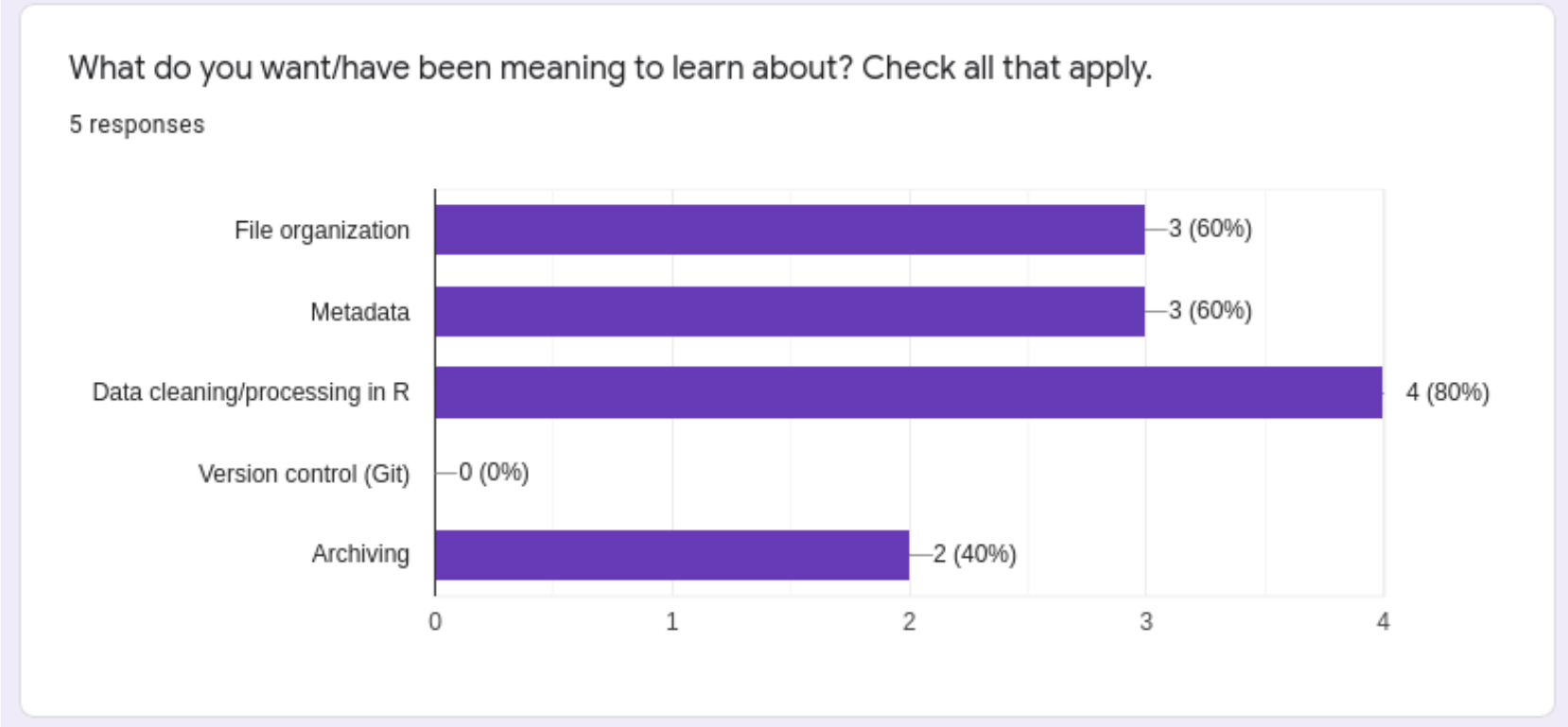
- Everyone wants to learn about file organization and data cleaning/processing! So here we go!



`

What we are learning today & why

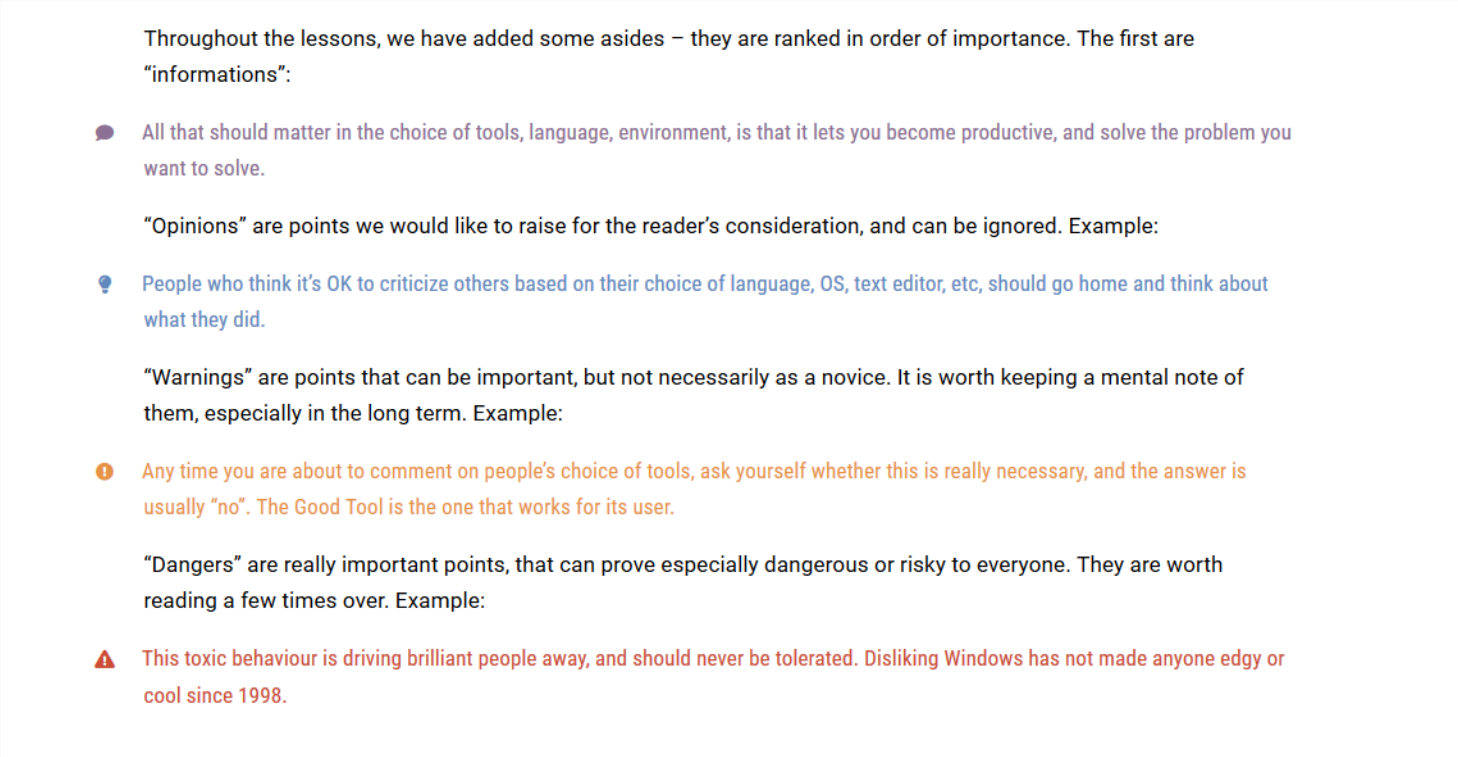
- Everyone wants to learn about file organization and data cleaning/processing! So here we go!



`

A couple notes

- Data archiving in the Ziter lab
  - We do have a place and and specific formatting requirements for data archiving - we can discuss this more later but this exists (reach out if this is something you want to do in the next couple of weeks)



Throughout the lessons, we have added some asides – they are ranked in order of importance. The first are "informations":

All that should matter in the choice of tools, language, environment, is that it lets you become productive, and solve the problem you want to solve.

"Opinions" are points we would like to raise for the reader's consideration, and can be ignored. Example:

People who think it's OK to criticize others based on their choice of language, OS, text editor, etc, should go home and think about what they did.

"Warnings" are points that can be important, but not necessarily as a novice. It is worth keeping a mental note of them, especially in the long term. Example:

Any time you are about to comment on people's choice of tools, ask yourself whether this is really necessary, and the answer is usually "no". The Good Tool is the one that works for its user.

"Dangers" are really important points, that can prove especially dangerous or risky to everyone. They are worth reading a few times over. Example:

This toxic behaviour is driving brilliant people away, and should never be tolerated. Disliking Windows has not made anyone edgy or cool since 1998.

[1]: Scientific Computing

Part 1: Data & File Organization

Tidy Data

- Tidy data is a framework for how data should be formatted for easy and efficient data cleaning created by Hadley Wickham
  - These principles are the underpinnings of `tidyverse` packages (e.g. ggplot2)

Principles

1. Each variable forms a column
2. Each observation forms a row
3. Each type of observational unit forms a table

`

Part 1: Data & File Organization

Tidy Data

- The best way to start data/file organization is to use best practices in data collection/spreadsheet formatting
- There are some really common spreadsheet errors that ecologists often use when collecting data
- We are not going to format a spreadsheet today, but I'd encourage you to take a look at that link and incorporate suggestions next time you're collecting/inputting data

`

Part 1: Data & File Organization

Good file structure is important because it:

- Ensures the integrity of your data
- Makes it easier to share your code with people
- Makes it easier to upload your code/data with manuscript submission
- Makes it easier to come back after a break

Best practices include (but are not limited to):

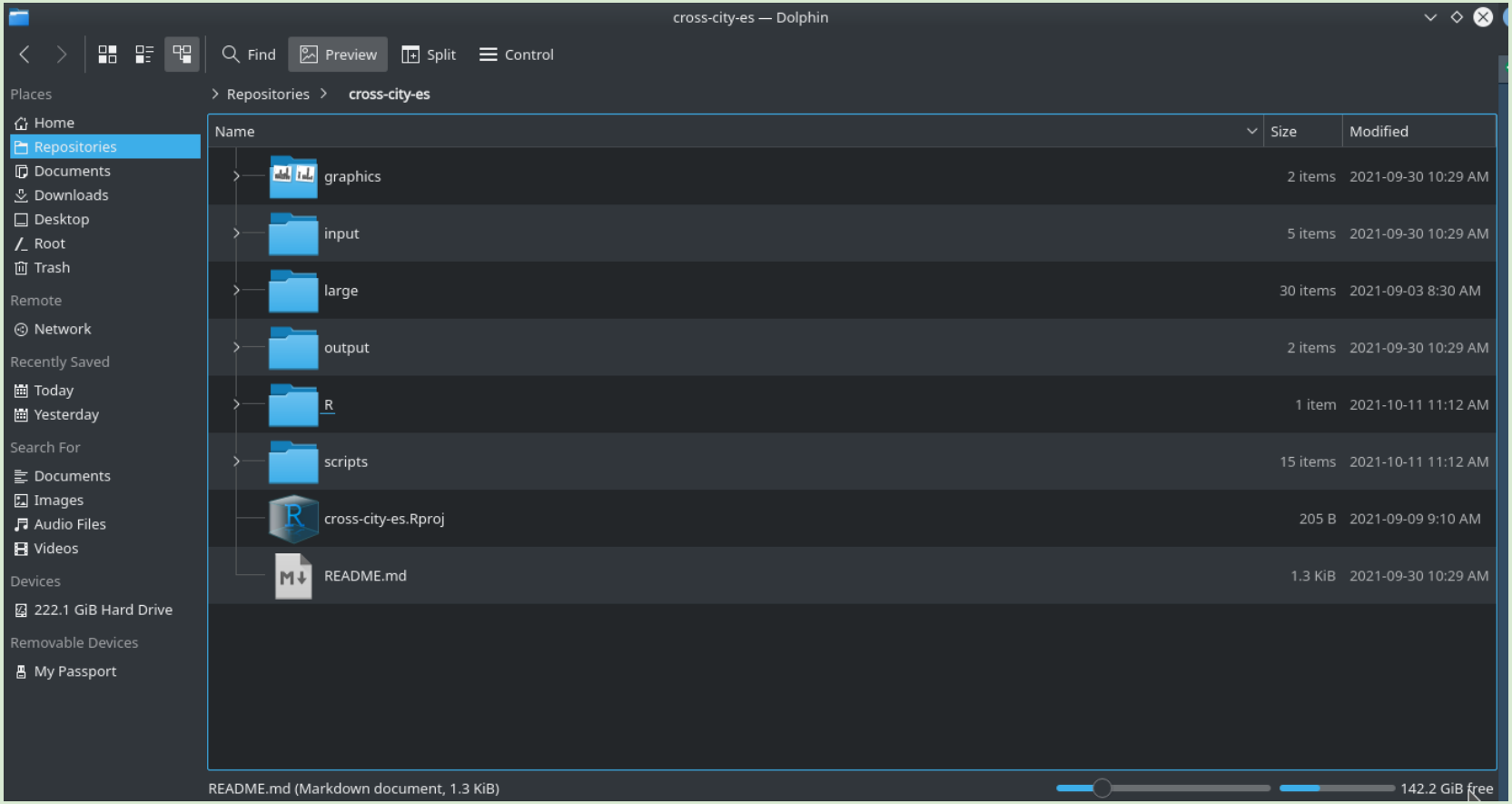
- Use an R Project file so that your project is easily shareable
- Always treat raw data as read-only
- Store cleaned data in a separate folder (or distinguish clearly)
- Treat output as disposable - you should always be able to re-generate with script
- Have separate function and figure scripts

`

[1]: Software Carpentry Project Management

Part 1: Data & File Organization

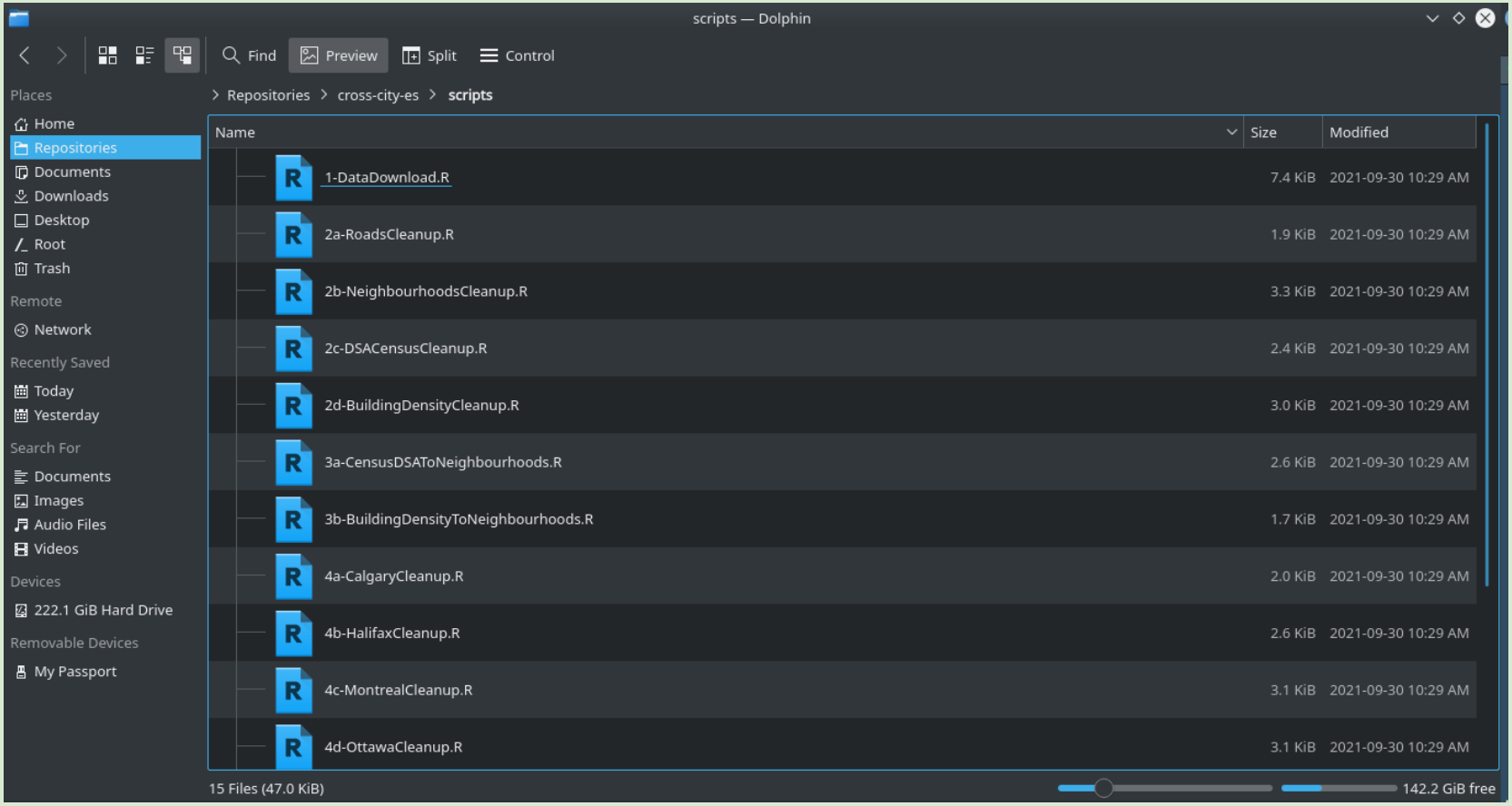
This is how I set up my file structure with my R Projects. If you want more examples, most of my repositories on GitHub are set up in in the same (or very similar manner):



`

Part 1: Data & File Organization

This is how I set up my file structure with my R Projects. If you want more examples, most of my repositories on GitHub are set up in in the same (or very similar manner):



`

Part 2: File Organization Practice

All together now!

- We are going to download messy and unstructured GitHub folder from here & unzip
- Then we are going to set up an R project & the file structure we just talked about

Ok... Moving on to cleaning data in R!

`

Part 3: Cleaning Data in R

Some reasons why I clean my data in R (use the tool that works for you)

1. Reproducible
2. Open-source and cross-platform
3. Reliable & clear
4. High-quality graphics
5. Great community & resources
6. Scales with datasets
7. Steep learning curve with a high payoff

`

[1]: Data Carpentry Data Analysis & Visualiation in R for Ecologists

Part 3: Cleaning Data in R

Two big "ecosystems" for data cleaning in R

`tidyverse`

- makes functions corresponding to the most common operations
- "do one thing at a time approach"
- chain together multiple things to make a easily understood group of operations
- many dependencies

`data.table`

- provides a high performance version of base R
- no dependencies
- complex syntax (I think) but it is very powerful once understood and can make your code extremely efficient by combining many operations and minimizing copying data, freeing up memory

`

[1]: tidyverse [2]: data.table [3]: Comparison of tidyverse and data.table

Part 4: Practice cleaning messy Data in R

I have created a spreadsheet that integrates MANY of the common problems in datasets... We are going to go through step by step and clean this dataset so it is ready for analysis.

NOTE: code is provided using both tidyverse and data.table philosophies in this workshop. Feel free to choose whichever one makes sense to you in the future (or mix & match!). Providing multiple ways to do everything was meant to give you options, not confuse you

NOTE 2: this code assumes you set up your files in an R project with the datasheet in an `input/` folder

`

Part 4: Practice cleaning messy Data in R

Problem: Importing Data

`tidyverse`

```
# initial import
tdf ← read_csv("input/example-data.csv")
# read_csv doesn't recognize that the file is delimited using ";" instead of ",".
tdf ← read_delim("input/example-data.csv", delim = ";")
# we still have the problem that the first 8 lines need to be removed for this to
tdf ← read_delim("input/example-data.csv", delim = ";", skip = 8)
```

`data.table`

```
# initial import using data.table
dtdf ← fread("input/example-data.csv")
# fread doesn't recognize that the file is delimited using ";" instead of ",". Let
dtdf ← fread("input/example-data.csv", sep = ";")
# we still have the problem that the first 8 lines in data.table need to be remove
dtdf ← fread("input/example-data.csv", sep = ";", skip = 8)
```

`

Part 4: Practice cleaning messy Data in R

Problem: Special Characters & UTF-8 Encoding

What would life in Québec be without special characters! When we investigate our tidyverse dataframe, we see an issue with "Montréal". This issue often arises with accents and other special characters, when the dataframe is not properly read as UTF-8

`tidyverse`

```
# tidyverse can be re-encoded to UTF-8 after import
tdf$City ← enc2utf8(as(tdf$City, "character"))
```

`data.table`

```
# data.table incorporates it into the import line
dtdf ← fread("input/example-data.csv", sep = ";", skip = 8, encoding = "UTF-8")
```

`

Part 4: Practice cleaning messy Data in R

Problem: NAs & Associated Values

There are often NAs in our datasets, which we may or may not want to keep often, there are also other values that count, biologically, as NAs but are not read into R automatically as NAs. For example, blanks, "NA" (instead of a true NA), zeroes (for some variables), etc.

`tidyverse`

```
# Strategy A: When importing (for tidyverse only)
tdf ← read_delim("input/example-data.csv", delim = ";", skip = 8, na = c("","  ",
# Strategy B: After importing
tdf$DBH ← na_if(tdf$DBH, 0)
# Drop the NAs
tdf ← drop_na(tdf)
```

`data.table`

```
# Strategy B: After importing
dtdf$DBH[dtdf$DBH == 0] ← NA
# Drop the NAs
dtdf ← na.omit(dtdf)
```

Part 4: Practice cleaning messy Data in R

Problem: Duplicates

Sometimes, there are duplicate observations/rows in datasets. These can really mess up your analysis if you don't catch them. Testing your dataset for duplicates and then removing what you find is an important step in data cleaning!

`tidyverse`

```
# investigate which lines are duplicates
tdup ← tdf[duplicated(tdf), c("TreeID", "City", "DBH", "Species,")]
# remove duplicates
tdf ← distinct(tdf)
```

`data.table`

```
# investigate which lines are duplicates
dtdup ← dtdf[duplicated(dtdf), c("TreeID", "City", "DBH", "Species,")]
# remove duplicates
dtdf ← unique(dtdf)
```

`

Part 4: Practice cleaning messy Data in R

Problem: Column Names

Bad column names can make your life a living hell. Before moving into analysis make sure your column names follow best practices, you'll thank me later! You don't want numbers, special characters, or spaces in your column names. Let's change our really bad column names.

`tidyverse`

```
tdf ← rename(tdf,
             Date = "1st Date Measured",
             Time = "Time  Measured",
             Species = "Species,")
```

`data.table`

```
dtdf ← setnames(dtdf,
                c("1st Date Measured", "Time  Measured", "Species,"),
                c("Date", "Time", "Species"))
```

`

Part 4: Practice cleaning messy Data in R

Problem: Value Formatting - Spelling Mistakes

In our dataset, Toronto is spelled wrong. If you have a huge dataset with many spelling errors, OpenRefine is a great tool to use. For us, our dataset is small and we can visually inspect it for spelling errors so we will replace the error in R.

`tidyverse`

```
tdf$City ← recode(tdf$City, Toronno = "Toronto")
```

`data.table`

```
dtdf$City[dtdf$City %in% c("Toronno", "TO", "The birthplace of Drake")] ← "Toront
```

`

Part 4: Practice cleaning messy Data in R

Problem: Value Formatting - Formatting Column Types

City and species are currently formatted as character types, but for our study we want them to act as factors - we need to reclassify them. Its always important to check the classes of your data, R makes assumptions when importing your data and can be wrong!

`tidyverse`

```
tdf$City ← as_factor(tdf$City)
tdf$Species ← as_factor(tdf$Species)
```

`data.table`

```
factor_cols ← c("City", "Species")
dtdf[ ,(factor_cols) := lapply(.SD, as.factor),
      .SDcols = factor_cols]
```

`

Part 4: Practice cleaning messy Data in R

Problem: Value Formatting - Text Parsing

We have a comma at the end of our species names, that we don't really want. We can use text parsing to remove those unwanted commas from the Species column.

`tidyverse`

```
tdf$Species ← str_remove(tdf$Species, "[,]") # this will remove a comma anywhere
```

`data.table`

```
dtdf$Species ← gsub(",","",dtdf$Species)
```

`

Part 4: Practice cleaning messy Data in R

Problem: Date/Time Formatting

Date and time formatting in R can be really tricky and frustrating sometimes but is often really necessary for field data (and other data). `anytime` is a cool package that makes the process a little bit easier. First we want a date-time column, not separate entities

`tidyverse`

```
tdf ← unite(tdf, "DateTime", Date:Time, sep = " ")
```

`data.table`

```
dtdf[,DateTime:=paste0(Date," ",Time)]
dtdf[, c("Date","Time"):=NULL]
```

Now we use anytime to format the columns. NOTE: Be careful of time zones, anytime will automatically set to where you are. If you are using data from other time zones and need to indicate that, use the `parsedate` package.

```
tdf$DateTime ← anytime(tdf$DateTime)
dtdf$DateTime ← anytime(dtdf$DateTime)
```

Part 4: Practice cleaning messy Data in R

Problem: Reshaping

Often the way datasets are initially set up are not ideal for things like plotting and modelling (they don't follow tidy data practices). So we need to reshape the dataframe - make it longer or wider - to do what we need to do.

`tidyverse`

```
# let's make the dataframe wider - we only want one entry per city
tdf_wide ← pivot_wider(tdf, names_from = Species, values_from = DBH)
# Hmmmmm that is not the most useful format - let's make it longer again (or melt
tdf_long ← pivot_longer(tdf_wide, cols = 4:14, names_to = "Species", values_to =
```

`data.table`

```
# dcast and melt for data.table
dtdf_wide ← dcast(dtdf, formula = City + TreeID + DateTime ~ Species, value.var =
dtdf_long ← melt(dtdf_wide, id.vars = c("TreeID","City", "DateTime"), variable.na
```

`

Part 4: Practice cleaning messy Data in R

Problem: Saving/Exporting Data

Don't forget to save your beautiful, cleaned data! Maybe you have a "cleaned" folder in your input directory, save it there.
If this is a final product, save it to output.
If this is an intermediate item, the best way to save it is as an .rds file
If this is a final product or something you will be sharing, save it as a .csv

`tidyverse`

```
saveRDS(tdf, "output/TidyData.rds")
write_csv(tdf, "output/TidyData.csv")
```

`data.table`

```
saveRDS(dtdf, "output/DataTableData.rds")
fwrite(dtdf, "output/DataTableData.csv")
```

`

Resources

- data.table in R Guide
- R for Reproducible Scientific Analysis
- Data Analysis & Visualization in R for Ecologists
- Data Organization in Spreadsheets for Ecologists
- Tidy Data by Hadley Wickham
- Dr. Christie Bahlai's Reproducible Quantitative Methods Course
- Wildlife Ecology & Evolution Lab's Guide by Alec Robitaille
- data.table Website
- tidyverse Website
- xaringan Website: package I used to make these slides
- OpenRefine

`