Loading & Cleaning Data in R

I know the file exists, why doesn't R?

steffilazerte
@steffilazerte@fosstodon.org
@steffilazerte
steffilazerte.ca

Dr. Steffi LaZerte
Analysis and Data Tools for Science

	River	Site	Ele	Amo	Wea
1	Grasse	Up stream	Al	0.605555555555556	sunny
2	Grasse	Mid stream	Al	0.425	snowy
3	Grase	Down stream	Al	0.194444444444444	wet
4	Oswegatchie	Up stream	Al	1	cloudy
5	Oswegatchie	Mid stream	Al	0.161111111111111	cloudy
6	Oswegatchie	Down stream	Al	0.033333333333333	sunny
7	Raquette	Up stream	Al	0.291666666666667	sunny
8	Raquette	Mid stream	Al	0.0388888888888889	cloudy
9	Raquette	Down stream	Al	0	sunny
10	St. Regis	Up stream	Al	0.680555555555556	sunny
11	St. Regis	Mid stream	Al	0.45	snowy
12	St. Regis	Down stream	Al	0.286111111111111	cloudy
13	Grasse	Up stream	Ba	0.505283381364073	wet
14	Grasse	Mid stream	Ba	0.564841498559078	snowy
15	Grasse	Down stream	Ba	0.523535062439962	cloudy
16	Oswegatchie	Up stream	Ba	0.357348703170029	snowy
17	Oswegatchie	Mid stream	Ba	0.560038424591739	sunny
18	Oswegatchie	Down stream	Ba	1	wet
19	Raquette	Up stream	Ba	0	cloudy
20	Raquette	Mid stream	Ba	0.22478386167147	sunny
21	Raquette	Dow stream	Ba	0.364073006724304	cloudy
22	St. Regis	Up stream	Ba	0.379442843419789	wet
23	St. Regis	Mid stream	Ba	0.296829971181556	snowy
24	St. Regis	Down stream	Ba	0.577329490874159	snowy
25	Grasse	Up stream	Br	0.107142857142857	snowy

Compiled: 2025-01-12

First things first

💾 Save previous script

📂 Open New File

(make sure you're in the RStudio Project)
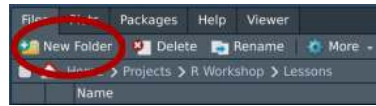
✏️ Write `library(tidyverse)` at the top

💾 Save this new script

(consider names like `cleaning.R` or `3_loading_and_cleaning.R`)

Download the data we'll use in this workshop

1. Create a '**data**' folder in your RStudio project
   - In the "Files" pane click on the folder icon **OR**
   - Navigate to your project folder via your computer's file browser

Click on "New Folder"

2. Right-click "Save Link As.." and download these files to your **data** folder
   - water_cleaned.xlsx
   - water_raw.csv
   - master_moch.txt
   - geolocators.csv
   - grain_size.txt
   - grain_size2.csv
   - grain_meta.csv
   - Sta A Data 2006-11-07.dmp

3

Side Note

R base vs. `tidyverse`

R base vs. `tidyverse`

R base

- Basic R
- Packages are installed and loaded by default
- Base pipe `|>` *

`tidyverse`

- Collection of 'new' packages developed by a team closely affiliated with RStudio
  - e.g., `ggplot2`, `dplyr`, `tidyr`, `readr`
  - Packages designed to work well together
- Use a slightly different syntax
- tidyverse pipe `%>%` or base pipe `|>` *

Useful to know if functions are
`tidyverse` or **R base**

Dealing with data

1. Loading data

- Get your data into R

2. Looking for problems

- Typos
- Incorrectly loaded data

3. Fixing problems

- Corrections
- Renaming
- Dealing with NA's

4. Setting formats

- Text,
- Numbers
- Factors
- Dates

5. Saving your data

Loading Data

Data types: What kind of data do you have?

Specific program files

Type	Extension	R Package	R function
Excel	.xls, .xlsx	readxl*	read_excel()
Open Document	.ods	readODS	read_ods()
SPSS	.sav, .zsav, .por	haven	read_spss()
SAS	.sas7bdat	haven	read_sas()
Stata	.dta	haven	read_dta()
Database Files	.dbf	foreign	read.dbf()

Convenient but…

- Can be unreliable
- Can take longer

For files that don't change, better to save as a `*.csv`

(Comma-separated-variables file)

Data types: What kind of data do you have?

General text files

Type	R base	readr package *
Comma separated	read.csv()	read_csv(), read_csv2()
Tab separated	read.delim()	read_tsv()
Space separated	read.table()	read_table()
Fixed-width	read.fwf()	read_fwf()

- **readr package** especially useful for big data sets (fast!)
- Error/warnings from readr are a bit more helpful

We'll focus on

- readxl package → read_excel()

Where is my data?

Common error

```
1  my_data <- read_csv("weather.csv")
```

```
Error: 'weather.csv' does not exist in current working directory ('/home/steffi/Projects/Workshops/workshop-dealing-with-data').
```

With no folder (just file name) R expects file to be in **Working directory**

Working directory is:

- Where your RStudio project is
- Your home directory (My Documents, etc.) [If not using RStudio Projects]
- Where you've set it (using `setwd()` or RStudio's Session > Set Working Directory)

Don't use `setwd()`

Where is my data?

A note on file paths (file locations)

```
1  /home
```

- folders separated by /
- home is a folder

Where is my data?

A note on file paths (file locations)

```
1   /home/steffi/
```

- folders separated by /
- home and steffi are folders
- steffi is a folder inside of home

Where is my data?

A note on file paths (file locations)

```
1  /home/steffi/Documents/R Projects/mydata.csv
```

- folders separated by /
- home, steffi, Documents, R Projects are folders
- steffi is inside of home, Documents is inside of steffi, etc.
- mydata.csv is a data file inside R Projects folder
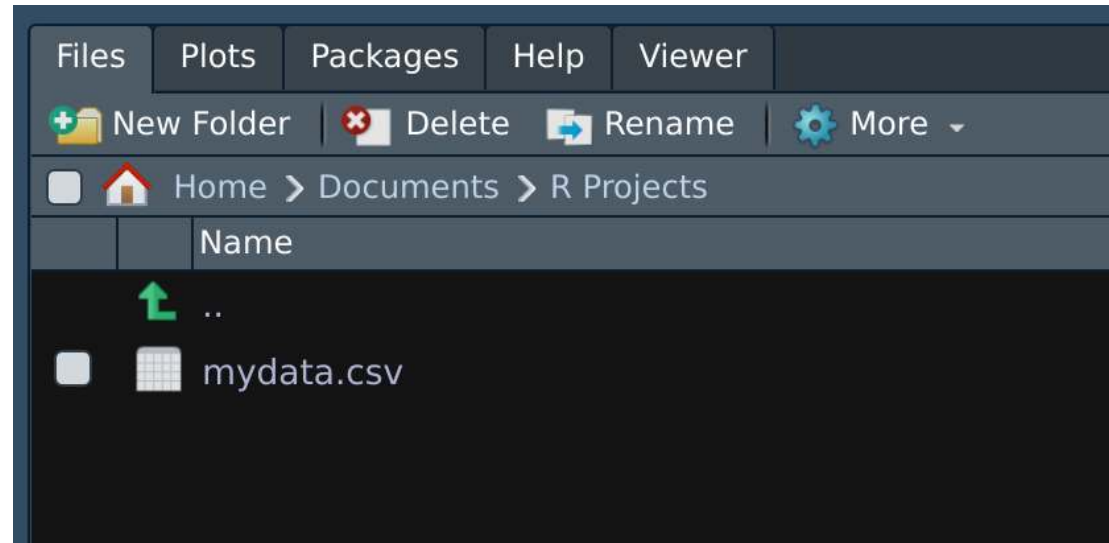
RStudio Files Pane

Where is my data?

Absolute Paths

OS	Path
LINUX	/home/steffi/Documents/R Projects/mydata.csv
WINDOWS	C:/Users/steffi/My Documents/R Projects/mydata.csv
MAC	/users/steffi/Documents/R Projects/mydata.csv

Full location, folders and filename

Relative Paths

Path	Where to look
mydata.csv	Here (current directory)
../mydata.csv	Go up one directory (../)
data/mydata.csv	Stay here, go into "data" folder (data/)
../data/mydata.csv	Go up one directory (../), then into "data" folder (data/)

Only *relative* info
Use relative symbols (e.g., . . /)

With RStudio 'Projects' only need to use **relative** paths

Keep yourself organized

For simple projects

- Create an 'RStudio Project' for each Project

- Create a specific "data" folder within each project (one per project)

```
1  - Prospect Lake Quality                        # Project Folder
2     - prospect_analysis.R
3     - data                                       # Data Folder
4        - prospect_data_2017-01-01.csv
5        - prospect_data_2017-02-01.csv
```

- Use **relative** paths to refer to this folder

```
1  d <- read_csv("data/prospect_data_2017-01-01.csv")
```

Let's Load Some Data!

Your turn: Load some data

Working with `water_cleaned.xlsx`

1. Load the package

```
1  library(readxl)
```

2. Read in the Excel file and assign to object `water`

```
1  water <- read_excel("data/water_cleaned.xlsx")
```

3. Use `head()` and `tail()` functions to look at the data
   e.g., `head(water)` and `tail(water)`

4. Click on the `water` object in your "Environment" pane to look at the whole data set

Use the *'tab'* key in RStudio when typing in the file name for auto-complete

How do I know which function to use?

Program-specific files

- Files which only normally open in a particular program (e.g., Excel)

- **Load with function from specific package (e.g. `read_excel` from readxl package)**

Text files

- Files which open in notepad

- Files which open in RStudio when you click on them in the Files Pane

- **Load with function from readr package (e.g. `read_csv()`, `read_tsv()`, etc.)**

Look at the file extension:

- `water_cleaned.xlsx` ➔ Excel file ➔ `read_excel()`

- `water_raw.csv` ➔ Comma-separated-variables ➔ `read_csv()`

But sometimes not clear…

How do I know which function to use?

Working with: `master_moch.txt`

- In lower right-hand pane, click on **Files**

- Click on **data** folder

- Click on **master_moch.txt**

- Click "View File" (if asked)

```
ID  region  hab freq     freq.sd p.notes
MCB02   kam 0.5266879074    3.9806600009    3.9806600009    0.4592592593
MCB03   kam -0.9707703735   4.1090031783    4.1090031783    0.5
MCB04   kam -0.9707703735   4.2463067674    4.2463067674    0.5151515152
```

This **does not** read the file into R, but only shows you
the contents as text.

How do I know which function to use?

Peak:

- Pick a read function with your best guess (`read_csv()` is a good start)
- Use n_max to read only first few rows

```
1  read_csv("data/master_moch.txt", n_max = 3)
```

```
# A tibble: 3 × 1
  `ID\tregion\thab\tfreq\tfreq.sd\tp.notes`
  <chr>
1 "MCB02\tkam\t0.5266879074\t3.9806600009\t3.9806600009\t0.4592592593"
2 "MCB03\tkam\t-0.9707703735\t4.1090031783\t4.1090031783\t0.5"
3 "MCB04\tkam\t-0.9707703735\t4.2463067674\t4.2463067674\t0.5151515152"
```

\t means tab, so this is tab-separated data

How do I know what to use?

Peak:

- Try again with read_tsv()

```
1  read_tsv("data/master_moch.txt", n_max = 3)   # note change in function!
```

```
# A tibble: 3 × 6
  ID    region     hab  freq freq.sd p.notes
  <chr> <chr>    <dbl> <dbl>   <dbl>   <dbl>
1 MCB02 kam      0.527  3.98    3.98   0.459
2 MCB03 kam     -0.971  4.11    4.11   0.5
3 MCB04 kam     -0.971  4.25    4.25   0.515
```

Excellent!

Specifics of loading functions

`col_names`

Working with Geolocator data

```
1  my_data <- read_csv("data/geolocators.csv")
2  my_data
```

```
# A tibble: 20 × 2
   `02/05/11 22:29:59`  `64`
   <chr>                <dbl>
 1 02/05/11 22:31:59       64
 2 02/05/11 22:33:59       38
 3 02/05/11 22:35:59       38
 4 02/05/11 22:37:59       34
 5 02/05/11 22:39:59       30
 6 02/05/11 22:41:59       34
 7 02/05/11 22:43:59       40
 8 02/05/11 22:45:59       46
 9 02/05/11 22:47:59       48
10 02/05/11 22:49:59       46
# i 10 more rows
```

Oops?

- `read_csv`, `read_tsv`, etc. assume that the first row contains the column names

- This file doesn't have headers

col_names

Working with Geolocator data

Declare no headings

```
1  my_data <- read_csv("data/geolocators.csv",
2                      col_names = FALSE)
3  my_data
```

```
# A tibble: 21 × 2
   X1                    X2
   <chr>              <dbl>
 1 02/05/11 22:29:59     64
 2 02/05/11 22:31:59     64
 3 02/05/11 22:33:59     38
 4 02/05/11 22:35:59     38
 5 02/05/11 22:37:59     34
 6 02/05/11 22:39:59     30
 7 02/05/11 22:41:59     34
 8 02/05/11 22:43:59     40
 9 02/05/11 22:45:59     46
10 02/05/11 22:47:59     48
# i 11 more rows
```

Name headings

```
1  my_data <- read_csv("data/geolocators.csv",
2                      col_names = c("date", "light"))
3  my_data
```

```
# A tibble: 21 × 2
   date                light
   <chr>              <dbl>
 1 02/05/11 22:29:59     64
 2 02/05/11 22:31:59     64
 3 02/05/11 22:33:59     38
 4 02/05/11 22:35:59     38
 5 02/05/11 22:37:59     34
 6 02/05/11 22:39:59     30
 7 02/05/11 22:41:59     34
 8 02/05/11 22:43:59     40
 9 02/05/11 22:45:59     46
10 02/05/11 22:47:59     48
# i 11 more rows
```

skip info rows before data

Working with Grain size data

```r
1  my_data <- read_tsv("data/grain_size.txt")
2  my_data
```

```
# A tibble: 36 × 7
   `DATA DOWNLOAD: 2015-09-23` ...2  ...3       ...4     ...5  ...6  ...7
   <chr>                       <chr> <chr>      <chr>    <chr> <chr> <chr>
 1 SYSTEM 001                  <NA>  <NA>       <NA>     <NA>  <NA>  <NA>
 2 LOGGER X                    <NA>  <NA>       <NA>     <NA>  <NA>  <NA>
 3 lab_num                     CSP   sample_num depth_lb csa   msa   fsa
 4 3177                        CSP01 CSP01-P-1-1 4            13.04 17.37 8.19
 5 3178                        CSP01 CSP01-P-1-2 12           10.74 16.9  7.92
 6 3179                        CSP01 CSP01-P-1-3 35           12.11 17.75 6.99
 7 3180                        CSP01 CSP01-P-1-4 53           17.61 18.16 6.29
 8 3181                        CSP01 CSP01-P-1-5 83           21.05 18.38 6.26
 9 3182                        CSP01 CSP01-P-1-6 105          19.02 18.43 6.28
10 3183                        CSP08 CSP08-P-1-1 10           11.6  17.14 8.18
# i 26 more rows
```

skip info rows before data

Working with Grain size data

```
1  my_data <- read_tsv("data/grain_size.txt")
2  my_data
```

Look at the file:

- Click on **Files** tab
- Click on **data** folder
- Click on **grain_size.txt**
- Click **"View file"** (if asked)

```
DATA DOWNLOAD: 2015-09-23
SYSTEM 001
LOGGER X
lab_num CSP sample_num  depth_lb    csa msa fsa
3177    CSP01    CSP01-P-1-1 4   13.04    17.37   8.19
3178    CSP01    CSP01-P-1-2 12  10.74    16.9    7.92
3179    CSP01    CSP01-P-1-3 35  12.11    17.75   6.99
3180    CSP01    CSP01-P-1-4 53  17.61    18.16   6.29
3181    CSP01    CSP01-P-1-5 83  21.05    18.38   6.26
```

Ah ha!
Metadata was stored at the top of the file

skip info rows before data

Working with Grain size data

- Add `skip = 3` to skip the first three rows

```
1  my_data <- read_tsv("data/grain_size.txt", skip = 3)
2  my_data
```

```
# A tibble: 33 × 7
   lab_num CSP   sample_num  depth_lb   csa   msa   fsa
     <dbl> <chr> <chr>          <dbl> <dbl> <dbl> <dbl>
 1    3177 CSP01 CSP01-P-1-1        4  13.0  17.4  8.19
 2    3178 CSP01 CSP01-P-1-2       12  10.7  16.9  7.92
 3    3179 CSP01 CSP01-P-1-3       35  12.1  17.8  6.99
 4    3180 CSP01 CSP01-P-1-4       53  17.6  18.2  6.29
 5    3181 CSP01 CSP01-P-1-5       83  21.0  18.4  6.26
 6    3182 CSP01 CSP01-P-1-6      105  19.0  18.4  6.28
 7    3183 CSP08 CSP08-P-1-1       10  11.6  17.1  8.18
 8    3184 CSP08 CSP08-P-1-2       27  15.4  16.2  6.76
 9    3185 CSP08 CSP08-P-1-3       90  14.9  15.8  7.12
10    3186 CSP02 CSP02-P-1-1        5   8.75  8.64  3.41
# i 23 more rows
```

Much better!

Your turn: Load this data set

Load Data: `Sta A Data 2006-11-07.dmp`

1. Look at the file

2. Decide which R function to use based on delimiter (comma, space, or tab?)

3. Any other options need to be specified?

It should look like this:

```
# A tibble: 19 × 7
  StartDate Time      Frequency `Rate/Temp`   Pwr Ant      SD
      <dbl> <time>        <dbl>       <dbl> <dbl> <chr> <dbl>
1     39022 17:15:36      150.        34.8    175 M0        0
2     39022 17:19:14      148.        19.2     72 M0        0
3     39022 17:19:25      148.        19.7    194 M1        0
4     39022 17:20:04      149.        33.8    104 M0        0
5     39022 17:20:17      149.        33.7    152 M1        0
6     39022 17:20:57      150.        34.2    188 M0        0
7     39022 17:22:50      148.         9.8    188 M0        0
# i 12 more rows
```

Too Easy?
Load some of your own tricky data

OR
Try to load the second sheet of `water_cleaned.xlsx` and rename

Looking for problems

Look at the data

- Make sure columns as expected (correctly assigned file format)

- Make sure no extra lines above the data (should we have used a skip?)

- Make sure column names look appropriate

```
1  library(palmerpenguins)
2  penguins
```

```
# A tibble: 344 × 8
   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex      year
   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>   <int>
 1 Adelie  Torgersen           39.1          18.7               181        3750 male     2007
 2 Adelie  Torgersen           39.5          17.4               186        3800 female   2007
 3 Adelie  Torgersen           40.3          18                 195        3250 female   2007
 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>     2007
 5 Adelie  Torgersen           36.7          19.3               193        3450 female   2007
 6 Adelie  Torgersen           39.3          20.6               190        3650 male     2007
 7 Adelie  Torgersen           38.9          17.8               181        3625 female   2007
 8 Adelie  Torgersen           39.2          19.6               195        4675 male     2007
 9 Adelie  Torgersen           34.1          18.1               193        3475 <NA>     2007
10 Adelie  Torgersen           42            20.2               190        4250 <NA>     2007
# i 334 more rows
```

Look at the data

- Did the whole data set load?
- Are there extra blank lines at the end of the data?

```
1  tail(penguins)
```

```
# A tibble: 6 × 8
  species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex      year
  <fct>     <fct>           <dbl>         <dbl>             <int>       <int> <fct>   <int>
1 Chinstrap Dream            45.7          17                 195        3650 female   2009
2 Chinstrap Dream            55.8          19.8               207        4000 male     2009
3 Chinstrap Dream            43.5          18.1               202        3400 female   2009
4 Chinstrap Dream            49.6          18.2               193        3775 male     2009
5 Chinstrap Dream            50.8          19                 210        4100 male     2009
6 Chinstrap Dream            50.2          18.7               198        3775 female   2009
```

`skim()` the data

`skim()` is from `skimr`

- Are the formats correct?
  - numbers (`numeric`),
  - text (`character`)
  - date (`date`, `POSIXct`, `datetime`)
  - categories (`factor`)

- Are values appropriate?
  - Should there be NAs?

- Are there any typos?

- Number of rows expected?

```
1  library(skimr)
2  skim(penguins)
```

```
── Data Summary ─────────────────────────
                           Values
Name                       penguins
Number of rows             344
Number of columns          8
_____
Column type frequency:
  factor                   3
  numeric                  5
_____
Group variables            None

── Variable type: factor ───────────────────────────────────────────
  skim_variable n_missing complete_rate ordered n_unique top_counts
1 species               0             1   FALSE        3 Ade: 152, Gen: 124, Chi: 68
2 island                0             1   FALSE        3 Bis: 168, Dre: 124, Tor: 52
3 sex                  11         0.968   FALSE        2 mal: 168, fem: 165

── Variable type: numeric ──────────────────────────────────────────
  skim_variable     n_missing complete_rate   mean      sd    p0    p25    p50    p75   p100 hist
1 bill_length_mm            2         0.994   43.9    5.46  32.1   39.2   44.4   48.5   59.6 ▄▆▅▃
2 bill_depth_mm             2         0.994   17.2    1.97  13.1   15.6   17.3   18.7   21.5 ▆▅▅▂
3 flipper_length_mm         2         0.994   201.    14.1   172    190    197    213    231 ▂▅▅▃
4 body_mass_g               2         0.994  4202.   802.   2700   3550   4050   4750   6300 ▅▅▃▂
5 year                      0             1  2008.   0.818  2007   2007   2008   2009   2009 ▆▁▇▁
```

count() categories

count() is from dplyr*

- Check for sample sizes and potential typos in categorical columns
- Assess missing values

```
1  count(penguins, species)
```
```
# A tibble: 3 × 2
  species       n
  <fct>     <int>
1 Adelie      152
2 Chinstrap    68
3 Gentoo      124
```
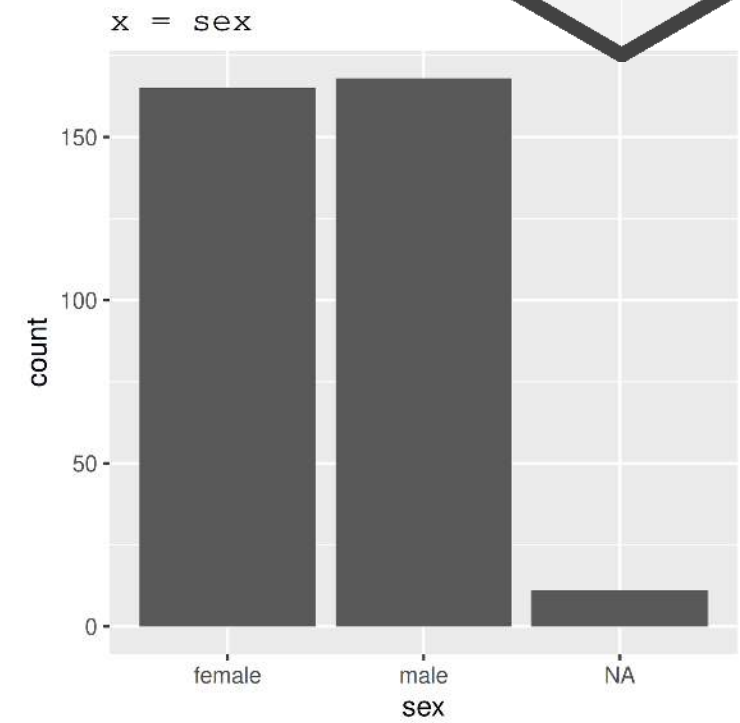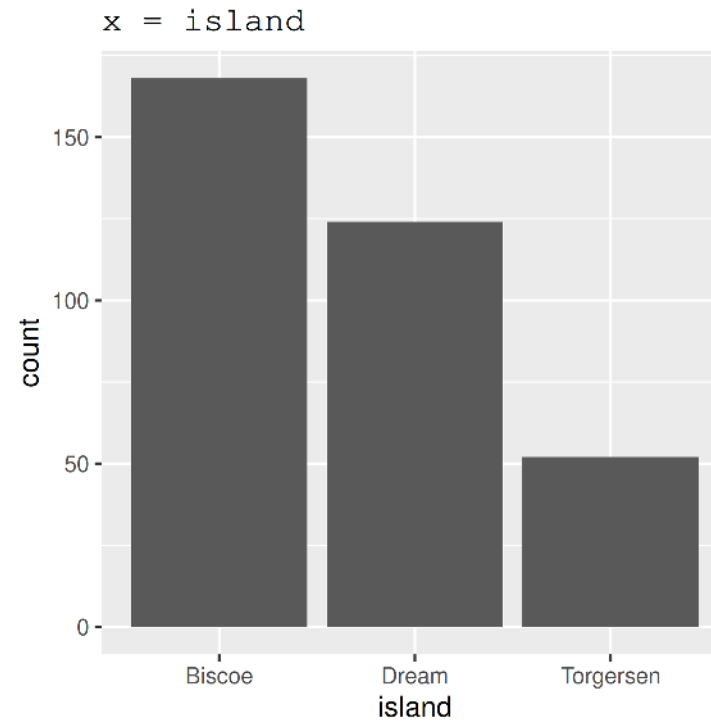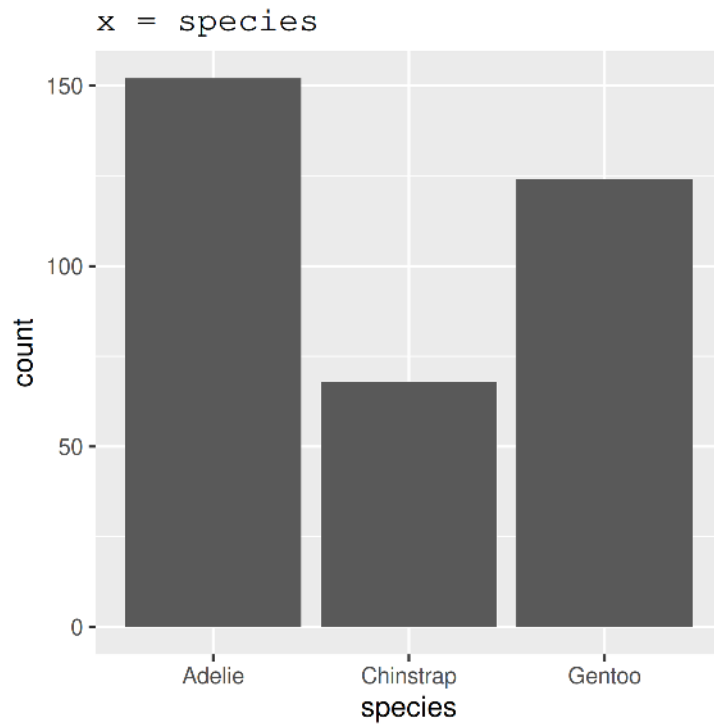
```
1  count(penguins, sex)
```
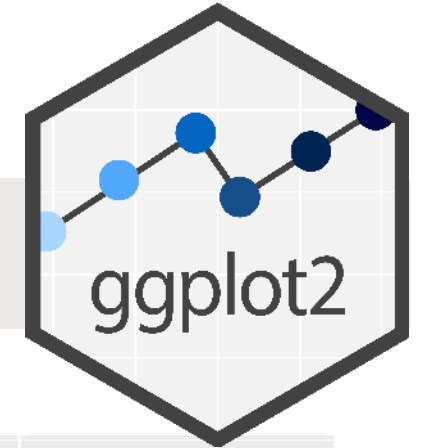```
A tibble: 3 × 2
  sex   n
  <fct>   <int>
1 female   165
2 male   168
3 <NA>   11
```

```
1 count(penguins, island)
```
```
A tibble: 3 × 2
  island   n
  <fct>   <int>
1 Biscoe   168
2 Dream   124
3 Torgersen   52
```
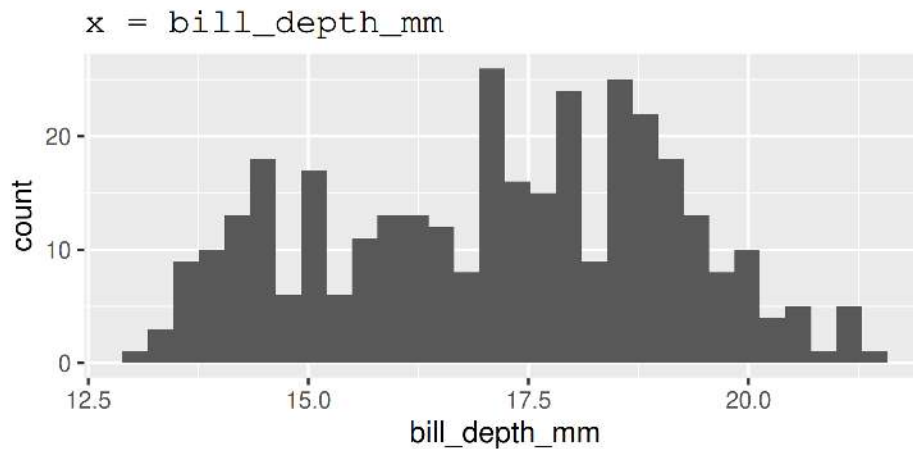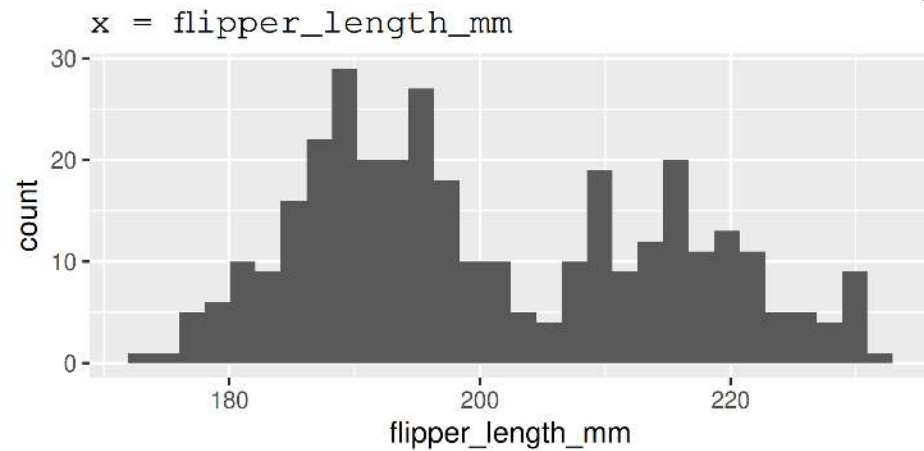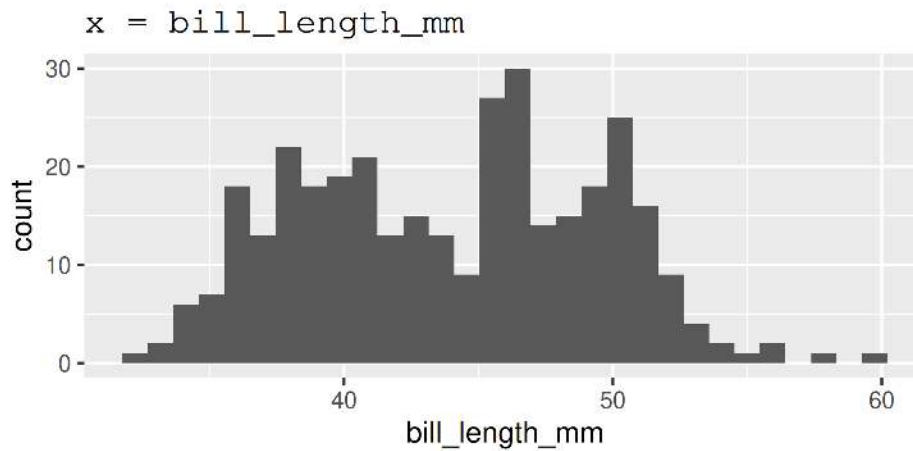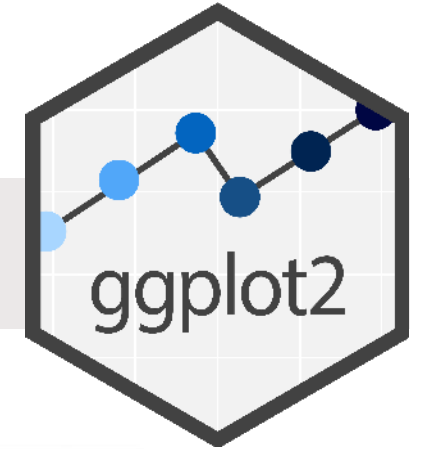
# Plot categories

```
1  # Replace COLUMN with the column name to explore
2  ggplot(data = penguins, aes(x = COLUMN)) +
3    geom_bar()
```

Plot numbers

```
1  # Replace COLUMN with the column name to explore
2  ggplot(data = penguins, aes(x = COLUMN)) +
3    geom_histogram()
```
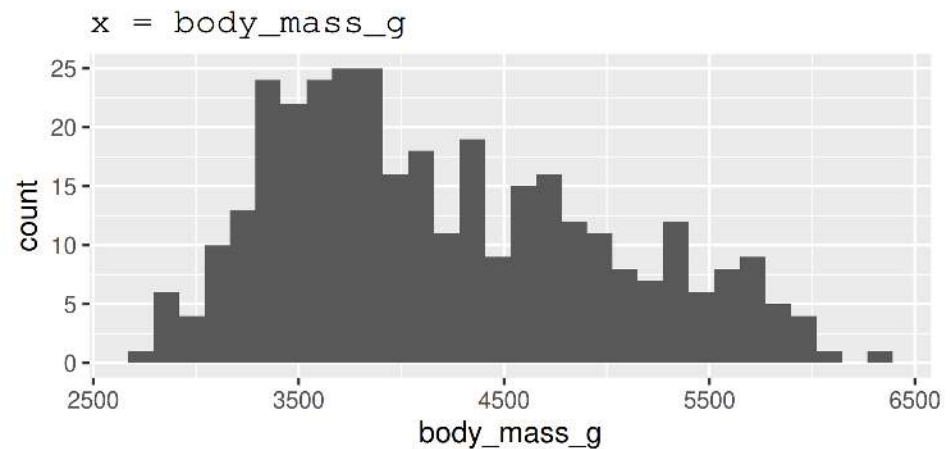
Example of problematic data

Look at the data

Working with: `water_raw.csv`

```r
1  water <- read_csv("data/water_raw.csv")
2  water
```

```
# A tibble: 300 × 7
   `River Name` Site        Ele       Amo `Temperature C°`  Year Wea
   <chr>        <chr>       <chr>    <dbl>            <dbl> <dbl> <chr>
 1 Grasse       Up stream   Al      0.606            10.9   2019 snowy
 2 Grasse       Mid stream  Al      0.425             8.68  2020 cloudy
 3 Grase        Down stream Al      0.194             8.75  2021 cloudy
 4 Oswegatchie  Up stream   Al      1                 0.791 2022 sunny
 5 Oswegatchie  Mid stream  Al      0.161             9.32  2023 snowy
 6 Oswegatchie  Down stream Al      0.0333           10.6   2019 wet
 7 Raquette     Up stream   Al      0.292             4.01  2020 snowy
 8 Raquette     Mid stream  Al      0.0389            5.96  2021 sunny
 9 Raquette     Down stream Al      NA                6.21  2022 cloudy
10 St. Regis    Up stream   Al      0.681             8.02  2023 wet
# i 290 more rows
```

- Column names are not R-friendly (`River Name` and `Temperature C°`) or obvious (what is `Ele`?)
- At least one typo in River (`Grase` should be `Grasse`)

Looking for problems

Your Turn!

```
1  water <- read_csv("data/water_raw.csv")
2  water
```

```
# A tibble: 300 × 7
   `River Name` Site        Ele      Amo `Temperature C°`  Year Wea
   <chr>        <chr>       <chr>   <dbl>           <dbl> <dbl> <chr>
 1 Grasse       Up stream   Al     0.606           10.9   2019 snowy
 2 Grasse       Mid stream  Al     0.425            8.68  2020 cloudy
 3 Grase        Down stream Al     0.194            8.75  2021 cloudy
 4 Oswegatchie  Up stream   Al     1                0.791 2022 sunny
 5 Oswegatchie  Mid stream  Al     0.161            9.32  2023 snowy
 6 Oswegatchie  Down stream Al     0.0333          10.6   2019 wet
 7 Raquette     Up stream   Al     0.292            4.01  2020 snowy
 8 Raquette     Mid stream  Al     0.0389           5.96  2021 sunny
 9 Raquette     Down stream Al     NA               6.21  2022 cloudy
10 St. Regis    Up stream   Al     0.681            8.02  2023 wet
# i 290 more rows
```

- `skim()` the data
- `count()` some columns
- Perhaps make some `ggplot()`s

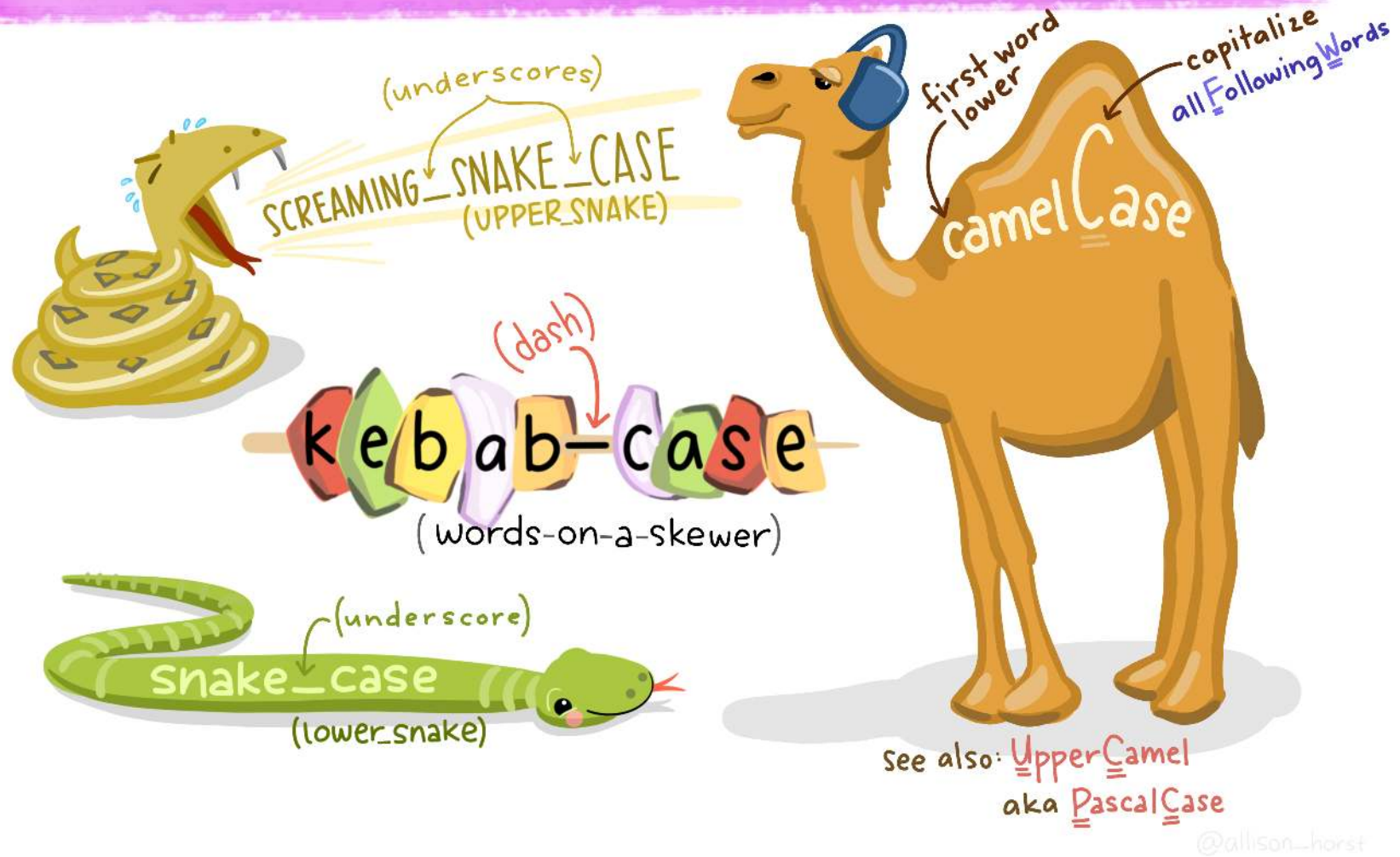Find any problems?

Fixing problems

Cleaning column names

`clean_names()` is from `janitor`*

```
1  library(janitor)
2  water <- clean_names(water)
3  water
```

```
1  # A tibble: 300 × 7
2    river_name  site        ele       amo temperature_c  year wea
3    <chr>       <chr>       <chr>   <dbl>         <dbl> <dbl> <chr>
4   1 Grasse      Up stream   Al      0.606         10.9   2019 snowy
5   2 Grasse      Mid stream  Al      0.425          8.68  2020 cloudy
6   3 Grase       Down stream Al      0.194          8.75  2021 cloudy
7   4 Oswegatchie Up stream   Al      1              0.791 2022 sunny
8   5 Oswegatchie Mid stream  Al      0.161          9.32  2023 snowy
9   6 Oswegatchie Down stream Al      0.0333        10.6   2019 wet
10  7 Raquette    Up stream   Al      0.292          4.01  2020 snowy
11  8 Raquette    Mid stream  Al      0.0389         5.96  2021 sunny
12  9 Raquette    Down stream Al     NA              6.21  2022 cloudy
13 10 St. Regis   Up stream   Al      0.681          8.02  2023 wet
14 # i 290 more rows
```

Side Note: Naming conventions

Side Note: Naming conventions

failed programming cases

ducklingcasE
(flat case, but capitalize the final letter)

BatCase
(upper camel, but upside down)

SH__K_CASE
(screaming snake but with chunks missing)
AR

(snake_case, but backward)
esac_retsbol

@allison_horst

Cleaning column names

rename() is from dplyr*

rename() columns

```
1  water <- rename(water, element = ele, amount = amo, temperature = temperature_c)
2  water
```

```
# A tibble: 300 × 7
   river_name  site        element  amount temperature  year wea
   <chr>       <chr>       <chr>     <dbl>       <dbl> <dbl> <chr>
 1 Grasse      Up stream   Al        0.606       10.9   2019 snowy
 2 Grasse      Mid stream  Al        0.425        8.68  2020 cloudy
 3 Grase       Down stream Al        0.194        8.75  2021 cloudy
 4 Oswegatchie Up stream   Al        1            0.791  2022 sunny
 5 Oswegatchie Mid stream  Al        0.161        9.32  2023 snowy
 6 Oswegatchie Down stream Al        0.0333      10.6   2019 wet
 7 Raquette    Up stream   Al        0.292        4.01  2020 snowy
 8 Raquette    Mid stream  Al        0.0389       5.96  2021 sunny
 9 Raquette    Down stream Al       NA            6.21  2022 cloudy
10 St. Regis   Up stream   Al        0.681        8.02  2023 wet
# i 290 more rows
```
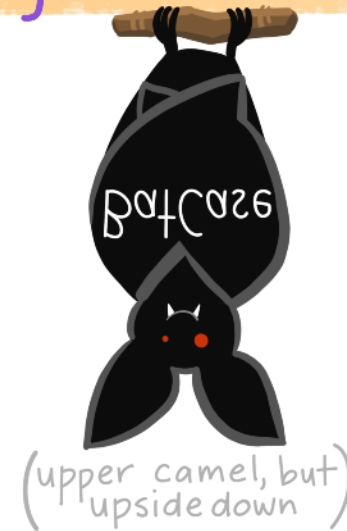
Subsetting columns

select() is from dplyr*

select() columns you want

```
1  water <- select(water, river_name, site, element, amount)
```

OR, unselect() columns you don't want

```
1  water <- select(water, -wea)
2  water
```

```
# A tibble: 300 × 6
   river_name  site        element  amount temperature  year
   <chr>       <chr>       <chr>     <dbl>       <dbl> <dbl>
 1 Grasse      Up stream   Al        0.606       10.9   2019
 2 Grasse      Mid stream  Al        0.425        8.68  2020
 3 Grase       Down stream Al        0.194        8.75  2021
 4 Oswegatchie Up stream   Al        1            0.791  2022
 5 Oswegatchie Mid stream  Al        0.161        9.32  2023
 6 Oswegatchie Down stream Al        0.0333      10.6   2019
 7 Raquette    Up stream   Al        0.292        4.01  2020
 8 Raquette    Mid stream  Al        0.0389       5.96  2021
 9 Raquette    Down stream Al        NA           6.21  2022
10 St. Regis   Up stream   Al        0.681        8.02  2023
# i 290 more rows
```

Cleaning columns

Put it all together

```
1  water <- read_csv("data/water_raw.csv")
2  water <- clean_names(water)
3  water <- rename(water, element = ele, amount = amo, temperature = temperature_c)
4  water <- select(water, -wea)
5  water
```

```
# A tibble: 300 × 6
   river_name  site        element  amount temperature  year
   <chr>       <chr>       <chr>     <dbl>       <dbl> <dbl>
 1 Grasse      Up stream   Al        0.606       10.9   2019
 2 Grasse      Mid stream  Al        0.425        8.68  2020
 3 Grase       Down stream Al        0.194        8.75  2021
 4 Oswegatchie Up stream   Al        1            0.791 2022
 5 Oswegatchie Mid stream  Al        0.161        9.32  2023
 6 Oswegatchie Down stream Al        0.0333      10.6   2019
 7 Raquette    Up stream   Al        0.292        4.01  2020
 8 Raquette    Mid stream  Al        0.0389       5.96  2021
 9 Raquette    Down stream Al        NA           6.21  2022
10 St. Regis   Up stream   Al        0.681        8.02  2023
# i 290 more rows
```

Note how code repeats data frame `water`...

Fixing typos

Remember the typos...

```
1  count(water, river_name)
```

```
# A tibble: 7 × 2
  river_name      n
  <chr>       <int>
1 Grase           1
2 Grasse         73
3 Oswegatchie    75
4 Raquette       74
5 St. Regis      75
6 grasse          1
7 raquette        1
```

Fixing typos

Replace typos

Combine the `if_else` function with the `mutate()` function

```
1  water <- mutate(water, river_name = if_else(river_name == "Grase", "Grasse", river_name))
```

Check that it's gone:

```
1  count(water, river_name)
```

```
# A tibble: 6 × 2
  river_name        n
  <chr>         <int>
1 Grasse           74
2 Oswegatchie      75
3 Raquette         74
4 St. Regis        75
5 grasse            1
6 raquette          1
```

Fixing typos

`if_else()` and `mutate()` from `dplyr` package*

`mutate()` creates or changes columns in a data frame:

```
1  mutate(dataframe, column = new_values)
```

`if_else()` tests for a condition, and returns one value if FALSE and another if TRUE

```
1  if_else(condition, value_if_true, value_if_false)
```

Iterative process

- Make some corrections
- Check the data
- Make some more corrections (either add to or modify existing code)

Many corrections?

Try **case_when()** from **dplyr** package*

case_when() tests for multiple conditions, and returns different values depending

```
1   case_when(condition1 ~ value_if_true1,
2             condition2 ~ value_if_true2,
3             condition3 ~ value_if_true3,
4             TRUE ~ default_value)
```

Your Turn: Fix another "Grasse" typo

1. Check the data with `count()`

2. Use `mutate()` and `if_else()` to fix the typo

```
1  water <- read_csv("data/water_raw.csv")
2  water <- clean_names(water)
3  water <- rename(water, element = ele, amount = amo, temperature = temperature_c)
4  water <- select(water, -wea)
5  water <- mutate(water, river_name = if_else(river_name == "Grase", "Grasse", river_name))
6
7  water <- mutate(???, ??? = ???)
```

Too Easy?
Examine and fix problems in your own data

OR

Use `case_when()` to fix all the river name typos at once...

Tangent: `tidyverse` functions

tidyverse functions

rename(), select(), mutate()

- tidyverse functions always start with the **data**, followed by other arguments
- you can reference any **column** from '**data**'

```
1  water <- read_csv("data/water_raw.csv")
2  water <- clean_names(water)
3  water <- rename(water, element = ele, amount = amo, temperature = temperature_c)
4  water <- select(water, -wea)
5  water <- mutate(water, river_name = if_else(river_name %in% c("Grase", "grasse"), "Grasse",  river_name))
```

- rename() changes column names
- select() chooses columns to keep or to remove (with -)
- mutate() changes column contents

Why use `tidyverse` functions?

Pipes! `|>`* Allow you to string commands together

Instead of:

```
1  water <- read_csv("data/water_raw.csv")
2  water <- clean_names(water)
3  water <- rename(water, element = ele, amount = amo, temperature = temperature_c)
4  water <- select(water, -wea)
5  water <- mutate(water,
6                  river_name = case_when(river_name %in% c("Grase", "grasse") ~ "Grasse",
7                                         river_name == "raquette" ~ "Raquette",
8                                         TRUE ~ river_name))
```

We have:

```
1  water <- read_csv("data/water_raw.csv") |>
2    clean_names() |>
3    rename(element = ele, amount = amo, temperature = temperature_c) |>
4    select(-wea) |>
5    mutate(river_name = case_when(river_name %in% c("Grase", "grasse") ~ "Grasse",
6                                  river_name == "raquette" ~ "Raquette",
7                                  TRUE ~ river_name))
```

Play around

Take a moment to play with this code in your console

Convert this:

```
1  water <- read_csv("data/water_raw.csv")
2  water <- clean_names(water)
3  water <- rename(water, element = ele, amount = amo, temperature = temperature_c)
4  water <- select(water, -wea)
5  water <- mutate(water,
6               river_name = case_when(river_name %in% c("Grase", "grasse") ~ "Grasse",
7                                      river_name == "raquette" ~ "Raquette",
8                                      TRUE ~ river_name))
```

To this:

```
1  water <- read_csv("data/water_raw.csv") |>
2    clean_names() |>
3    rename(element = ele, amount = amo, temperature = temperature_c) |>
4    select(-wea) |>
5    mutate(river_name = case_when(river_name %in% c("Grase", "grasse") ~ "Grasse",
6                                  river_name == "raquette" ~ "Raquette",
7                                  TRUE ~ river_name))
```

Dealing with NAs

Data that *is* missing
Data that *should* be missing

Exploring NAs

- We saw missing values in `amount`
- Use `filter()` to take a closer look

```
1  filter(water, is.na(amount))
```

```
# A tibble: 39 × 6
   river_name  site        element amount temperature  year
   <chr>       <chr>       <chr>    <dbl>       <dbl> <dbl>
 1 Raquette    Down stream Al          NA        6.21  2022
 2 Raquette    Up stream   Ba          NA        5.23  2022
 3 Raquette    Up stream   Br          NA      -99     2019
 4 Oswegatchie Up stream   Ca          NA        4.76  2023
 5 Raquette    Down stream Ce          NA       13.9   2020
 6 Grasse      Up stream   Cu          NA        9.13  2019
 7 Raquette    Down stream Dy          NA        4.98  2019
 8 Raquette    Down stream Er          NA        3.07  2021
 9 Raquette    Down stream Fe          NA        7.20  2023
10 Raquette    Down stream Gd          NA        4.73  2020
# i 29 more rows
```

Omitting NAs

drop_na() is from tidyr*

Omit NAs from the amount column only (drop those rows)

```
1  water_no_na <- drop_na(water, amount)
```

Omit **all** NAs from **all** columns (drop those rows)

```
1  water_no_na <- drop_na(water)
```

Check…

```
1  filter(water_no_na, is.na(amount))
```

```
# A tibble: 0 × 6
# i 6 variables: river_name <chr>, site <chr>, element <chr>, amount <dbl>, temperature <dbl>, year <dbl>
```

```
1  nrow(water_no_na)
```

```
[1] 261
```

Side Note: `filter()` also omits NAs 😱

If we filter by the column with NAs, they are silently dropped

```
1  filter(water, amount < 0.05)
```

```
# A tibble: 15 × 6
   river_name   site        element amount temperature  year
   <chr>        <chr>       <chr>    <dbl>       <dbl> <dbl>
 1 Oswegatchie  Down stream Al      0.0333       10.6   2019
 2 Raquette     Mid stream  Al      0.0389        5.96  2021
 3 Grasse       Mid stream  Br      0.0357       12.4   2019
 4 St. Regis    Up stream   Br      0.0357        3.52  2022
 5 St. Regis    Mid stream  Br      0.0357        0.936 2023
 6 Raquette     Mid stream  Ce      0.0116        6.61  2019
 7 Raquette     Mid stream  Fe      0.00656      10.8   2022
 8 Grasse       Up stream   K       0.0313        3.61  2021
 9 Raquette     Mid stream  La      0.0275        2.50  2020
10 Oswegatchie  Down stream Mn      0.00672       8.89  2019
# i 5 more rows
```

We need to be explicit if we want to keep them

```
1  filter(water, amount < 0.05 | is.na(amount))
```

```
# A tibble: 54 × 6
   river_name   site        element amount temperature  year
   <chr>        <chr>       <chr>    <dbl>       <dbl> <dbl>
 1 Oswegatchie  Down stream Al      0.0333       10.6   2019
 2 Raquette     Mid stream  Al      0.0389        5.96  2021
 3 Raquette     Down stream Al      NA            6.21  2022
 4 Raquette     Up stream   Ba      NA            5.23  2022
 5 Grasse       Mid stream  Br      0.0357       12.4   2019
 6 Raquette     Up stream   Br      NA          -99     2019
 7 St. Regis    Up stream   Br      0.0357        3.52  2022
 8 St. Regis    Mid stream  Br      0.0357        0.936 2023
 9 Oswegatchie  Up stream   Ca      NA            4.76  2023
10 Raquette     Mid stream  Ce      0.0116        6.61  2019
# i 44 more rows
```

Replacing NAs

replace_na() is from tidyr*

```
1  water_no_na <- mutate(water, amount = replace_na(amount, 0))
```

Check…

```
1  filter(water_no_na, is.na(amount))
```

```
# A tibble: 0 × 6
# i 6 variables: river_name <chr>, site <chr>, element <chr>, amount <dbl>, temperature <dbl>, year <dbl>
```

```
1  nrow(water_no_na)
```

```
[1] 300
```

No more NAs!
Same number of rows

(If you want to do a more complex replacement, you'll have to use if_else() or case_when() like we did for typos.)

Converting *to* NA

Remember the problem with `temperature`?

```
1  filter(water, temperature < -10)
```

```
# A tibble: 3 × 6
  river_name  site         element amount temperature  year
  <chr>       <chr>        <chr>    <dbl>       <dbl> <dbl>
1 Raquette    Up stream    Br          NA         -99  2019
2 Oswegatchie Mid stream   K        0.426         -99  2020
3 St. Regis   Mid stream   La       0.367         -99  2023
```

`na_if()` is from `dplyr`*

```
1  water <- mutate(water, temperature = na_if(temperature, -99))
```

Check…

```
1  filter(water, is.na(temperature))
```

```
# A tibble: 3 × 6
  river_name  site         element amount temperature  year
  <chr>       <chr>        <chr>    <dbl>       <dbl> <dbl>
1 Raquette    Up stream    Br          NA          NA  2019
2 Oswegatchie Mid stream   K        0.426          NA  2020
3 St. Regis   Mid stream   La       0.367          NA  2023
```

Fixing formats

Fixing formats

Basic data types

- Text (Characters or Strings)
- Numbers (Doubles, Integers)
- Logical (TRUE or FALSE)
- Factor (Categories)
- Dates
- Date/Times

Look for problems

```
1  water
```

```
# A tibble: 300 × 6
   river_name  site        element  amount temperature  year
   <chr>       <chr>       <chr>     <dbl>       <dbl> <dbl>
 1 Grasse      Up stream   Al        0.606        10.9  2019
 2 Grasse      Mid stream  Al        0.425        8.68  2020
 3 Grasse      Down stream Al        0.194        8.75  2021
 4 Oswegatchie Up stream   Al        1            0.791 2022
 5 Oswegatchie Mid stream  Al        0.161        9.32  2023
 6 Oswegatchie Down stream Al        0.0333       10.6  2019
 7 Raquette    Up stream   Al        0.292        4.01  2020
 8 Raquette    Mid stream  Al        0.0389       5.96  2021
 9 Raquette    Down stream Al        NA           6.21  2022
10 St. Regis   Up stream   Al        0.681        8.02  2023
# i 290 more rows
```

Year could be categorical (factor)
Better for plotting!

Convert to categorical

```
1  water <- mutate(water, year = as.factor(year))
2  water
```

```
# A tibble: 300 × 6
   river_name  site        element  amount temperature year
   <chr>       <chr>       <chr>     <dbl>       <dbl> <fct>
 1 Grasse      Up stream   Al        0.606       10.9  2019
 2 Grasse      Mid stream  Al        0.425        8.68 2020
 3 Grasse      Down stream Al        0.194        8.75 2021
 4 Oswegatchie Up stream   Al        1            0.791 2022
 5 Oswegatchie Mid stream  Al        0.161        9.32 2023
 6 Oswegatchie Down stream Al        0.0333      10.6  2019
 7 Raquette    Up stream   Al        0.292        4.01 2020
 8 Raquette    Mid stream  Al        0.0389       5.96 2021
 9 Raquette    Down stream Al        NA           6.21 2022
10 St. Regis   Up stream   Al        0.681        8.02 2023
# i 290 more rows
```

Changing classes

Function	Input	Output
`as.character()`	Any vector	Text (Characters)
`as.numeric()`	Any vector (but returns NAs if not numbers)	Numbers
`as.logical()`	TRUE, FALSE, T, F, 0 (FALSE), any other number (all TRUE)	TRUE or FALSE
`as.factor()`	Any vector	Categories

We'll deal with dates and times later…

Put it all together...

```r
1  water <- read_csv("data/water_raw.csv") |>
2    clean_names() |>
3    rename(element = ele, amount = amo, temperature = temperature_c) |>
4    select(-wea) |>
5    mutate(river_name = case_when(river_name %in% c("Grase", "grasse") ~ "Grasse",
6                                  river_name == "raquette" ~ "Raquette",
7                                  TRUE ~ river_name),
8          amount = replace_na(amount, 0),
9          temperature = na_if(temperature, -99),
10         year = as.factor(year))
```

And you have a clean, corrected data frame ready to use

- You have not changed the original data
- You have a **reproducible** record of all corrections
- You can alter these corrections at any time
- You have formatted your data for use in R
- Read these steps line by line to remind yourself what you did

Put it all together...

Feel free to annotate within a pipe

```r
 1  water <- read_csv("data/water_raw.csv") |>
 2    # Fix column names
 3    clean_names() |>
 4    rename(element = ele, amount = amo, temperature = temperature_c) |>
 5    select(-wea) |>
 6    mutate(
 7      # Correct typos
 8      river_name = case_when(river_name %in% c("Grase", "grasse") ~ "Grasse",
 9                             river_name == "raquette" ~ "Raquette",
10                             TRUE ~ river_name),
11      # Missing amounts should be 0
12      amount = replace_na(amount, 0),
13      # Problems with temperature logger, -99 is a mistake
14      temperature = na_if(temperature, -99),
15      # Convert for plotting
16      year = as.factor(year))
```

Dates and Times

(Or why does R hate me?)

Dates and Times

Working with Geolocator data

- Date/times aren't always recognized as date/times

```
1  geolocators <- read_csv("data/geolocators.csv", col_names = c("time", "light"))
2  geolocators
```

```
# A tibble: 21 × 2
   time             light
   <chr>            <dbl>
1 02/05/11 22:29:59    64
2 02/05/11 22:31:59    64
3 02/05/11 22:33:59    38
4 02/05/11 22:35:59    38
5 02/05/11 22:37:59    34
6 02/05/11 22:39:59    30
# i 15 more rows
```

Here `time` column is considered `chr` (character/text)

You may know it's a date, but R does not

LUBRIDATE: wrangle times + dates!

YEARS

MONTHS

DAYS

Horst '18

lubridate package*

- Part of `tidyverse`, but needs to be loaded separately
- Great for converting date/times (i.e. telling R this is a date/time)

```
1  library(lubridate)
2  geolocators <- mutate(geolocators, time_fixed = dmy_hms(time))
3  geolocators
```

```
# A tibble: 21 × 3
  time               light time_fixed
  <chr>              <dbl> <dttm>
1 02/05/11 22:29:59     64 2011-05-02 22:29:59
2 02/05/11 22:31:59     64 2011-05-02 22:31:59
3 02/05/11 22:33:59     38 2011-05-02 22:33:59
4 02/05/11 22:35:59     38 2011-05-02 22:35:59
5 02/05/11 22:37:59     34 2011-05-02 22:37:59
6 02/05/11 22:39:59     30 2011-05-02 22:39:59
# i 15 more rows
```

Now `time_fixed` column is considered `dttm` (Date/Time)

So **You** know it's a Date/Time and now **R** knows too

`lubridate` package*

Generally, only the order of the **y**ear, **m**onth, **d**ay, **h**our, **m**inute, or **s**econd matters.

For example

date/time format	function	output class
2018-01-01 13:09:11	`ymd_hms()`	dttm (POSIXct/POSIXt)
12/20/2019 10:00 PM	`mdy_hm()`	dttm (POSIXct/POSIXt)
31/01/2000 10 AM	`dmy_h()`	dttm (POSIXct/POSIXt)
31-01/2000	`dmy()`	Date

`lubridate` is smart enough to detect AMs and PMs

Saving data

(For the love of all that is good don't *lose* that data!!!)*

* but if you've been paying attention, you know that you only need the script 😉

Saving data

Keep yourself organized

- Keep your R-created data in a **different** folder from your 'raw' data*
- If you have a lot going on, split your work into several scripts, and number the both the scripts AND the data sets produced:
- `1_cleaned.csv`
- `2_summarized.csv`
- `3_graphing.csv`

Save your data to file:

- First create the *datasets* folder

```
1  write_csv(water, "datasets/water_cleaned.csv")
```

Dealing with data

1. Loading data

- Get your data into R

2. Looking for problems

- Typos
- Incorrectly loaded data

3. Fixing problems

- Corrections
- Renaming
- Dealing with NA's

4. Setting formats

- Text
- Numbers
- Factors
- Dates

5. Saving your data

Wrapping up: Common mistakes

Assuming your data is in one format when it's not

- Print your data to the console and use `skim()` to explore the format of your data
- Use `skim()`, `count()`, `filter()`, `select()`, `ggplot()` to explore the content of your data

Wrapping up: Common mistakes

Confusing pipes with function arguments

- Pipes (|> or %>%) pass the *output* from one function as *input* to the next function:

```
1  my_data <- my_data |>                 # Pass my_data
2    filter(my_column > 5) |>            # Pass my_data, filtered
3    select(my_column, my_second_column)
```

- Arguments may be on different lines, but all part of *one* function

```
1  my_data <- my_data |>                     # Pass my_data
2    mutate(my_column1 = if_else(...),       # No passing (no pipes!)
3           my_column2 = if_else(...),       # Instead, give 3 arguments to mutate:
4           my_column3 = if_else(...))       # Arguments separated by ",", and surrounded by ( )
```

Wrapping up: Further reading

- R for Data Science
    - Chapter 3: Data transformation
    - Chapter 6: Workflow: scripts and projects
    - Chapter 14: Strings
    - Chapter 16: Factors
    - Chapter 4.3: Workflow: code style > Pipes