
Day 1, May 19: Spreadsheets & OpenRefine

Data needed:

- for the Spreadsheets lesson:
- https://ndownloader.figshare.com/files/2252083
- for the Open Refine lesson:
- https://ndownloader.figshare.com/files/7823341

Course website: https://smithsonianworkshops.github.io/2021-05-19-smithsonian-online/

Attendance Day 1
Please add your name, unit and favorite type of dessert
Jon Barrett NASM CPU, Blueberry pie
-Ashraf Moursi, SAO High Energy Astrophysics Division, sticky bun!
-Henrique V. Figueiró, SCBI, Chocolate cake
- Chad Kamen, CFCH, Tiramisu
- Gemma Cantlow, research fellow with SCLDA and University of Leicester - apple crumble
- Crystal Ferrer, Cooper Hewitt, equal flavor opportunity ice cream fanatic
- Helene Muller-Landau, STRI staff scientist, flan
-Jessica Walthew, Cooper Hewitt conservation, all desserts welcome except flan
-Sarah Alley, Smithsonian Environmental Research Center, apple fritter
-Jeffrey Sosa-Calvo, NMNH, Tiramisu and napoleon
- Jose Diaz, SCBI, cinnamon roll
-Rayna Andrews, Archives of American Art, all desserts welcome, but right now I'm craving key lime pie
Tricia Goulding, NMNH IZ - Crème Brûlée
Barbara Balestra cookies (all of them)
Claudia Zapata-Smithsonian American Art Museum-Pudding
Rachel Collin (STRI) - Durian with sticky rice
Robin O'Hern - NMNH anthro, mint chocolate chip ice cream
Amy Driskell - NMH LAB, chocolate torte

Jen - NMNH, Encyclopedia of Life, ice cream, mint chip or cookies & cream
Ida Hartvig, SERC, generally all types of desserts! not durian though!
Beverly Lamberson Most anything chocolate

<u>Matt: Excel /spreadsheet data management notes</u>
Messy data observations: inconsistent labels for columns, date issues, color coding not good practice, use columns instead of headers; keep single type of information in each column (e.g. measurement separate from unit)
--differentiate between null (missing) data and recorded data that is zero
--pay attention to date formats
best practice is different columns for date/month/year

<u>Jen: OpenRefine</u>
Facets: use facets from any column header to open up options.

- you can directly edit duplicate/misspelled names to condense but there's a better automated way

-sort by... count gives most common first
Clustering: use cluster tool to [somehow, algorithmically] collapse
Parsing: use column header toggle to open option for split into columns; select _ (space) as delimiter, execute
Undo/Redo: brings up your history, click on the history to revert to previous
* don't hit back in your browser [to go back]! But if you do it's ok, it's autosaving and tracking your changes in this field
Common transforms, In a column header >edit>common transforms> trim leading... will remove leading spaces
Edit column names in the same toggle menu for column headers (for split columns) but you can't reuse a name
Filtering (does not edit data; check this in undo/redo if needed)

- Text filter: type a substring, case sensitive/insensitive as you prefer
- To use facet + filter, and select more than one option use "include" while hovering over the name (or "exclude")
- Use invert to reverse your selection to everything else, reset

Sort: use toggle menu in column header; shortcut to current sort is next to rows at the top of your sheet

- Reorder order permanently will change the data instead of just the view. (under sort short cut)

To reformat columns: under header - edit cells- common transform-- to number (e.g.)

- now you can have Numeric facets (not just text)- will give you a graphic representation of data; there's a handle on each end to change the selection. (+ you can have more than one)

Scripts *just Google it, scripting is in GREL, a language derived from Google Refine Language (now generalized)*; e.g. lots of if, then statements for new columns, etc. easy to do

- Do save the scripts as method documentation.

Exporting: you can export sections or your full data set- use the menu in custom to set which columns etc. you want for a smaller export.

Day 2, May 20: SQL

Data needed: https://doi.org/10.6084/m9.figshare.1314459
Click on download all
Direct download: https://ndownloader.figshare.com/articles/1314459/versions/10

Course website: https://smithsonianworkshops.github.io/2021-05-19-smithsonian-online/

Helpful figures to understand joins can be found on the python lesson: https://datacarpentry.org/python-ecology-lesson/05-merging-data/index.html

Attendance Day 2

Please add your name and your favorite thing you learned about spreadsheets or OpenRefine from yesterday!
-Rachel Collin - it's a great way to find typos.
-Henrique V. Figueiró - How to split columns
Ida Hartvig - excellent for combining information and for extracting subsets of data. will be useful for our collection data files!
-Jessica Walthew- OR good for splitting columns
-Rayna Andrews - it makes extracting data/filtering so much easier
-Chad Kamen - how to make scatterplots!
-Tricia Goulding- Using OR to filter data and clean up cells with different spellings or formats
-Jose Diaz- how to use openrefine to sort and clean data, super useful for my internship
-Ashraf Moursi - I learned I don't need to spend hours on a 200k+ row spreadsheet...
-Irene Holst- sorting groups
-Beverly Lamberson - Open refine is cool tool
-Sarah Alley - really helpful to learn how to navigate large datasets
-Gemma Cantlow - I found it very useful to learn how to filter data and split columns
-Robin O'Hern - how to use text filter to find words in long descriptions, also how to export and apply the things you did
-Jeffrey Sosa-Calvo- OpenRefine. cool tool
-Amy Driskell - faceting
-Helene Muller-Landau - OpenRefine looks like it could be really useful for the ForestGEO community
-Jon Barrett - OpenRefine looks like a great tool
- Amanda Devine - saving operations in a JSON file and applying them to new projects

-Full lesson: https://datacarpentry.org/sql-ecology-lesson/00-sql-introduction/index.html

Q: When I import a spreadsheet into OpenRefine, the colors disappear. The different colors contain information that we would like to keep. Is there a way to import colors into OpenRefine?
A: We can't import spreadsheet formatting into OpenRefine. According to the principles of Tidy Data, it is not ideal to store important information in colors, because many programs can't import formatting. It would be safer to create new columns in your spreadsheet to put in those values.
A: this is really interesting on color (is/is not) data or metadata, and how to (maybe, seems to be version dependent!) extract the info using macros in excel: https://stackoverflow.com/questions/24382561/excel-

formula-to-get-cell-color

SQLCipher seems to be an extension for SQLite that adds the ability to encrypt SQLite databases. For the purposes of our lesson, it should function the same way as DB Browser for SQLite
https://www.zetetic.net/sqlcipher/design/

Challenge question

Open each of these csv files and explore them. What information is contained in each file? Specifically, if I had the following research questions:

- How has the hindfoot length and weight of *Dipodomys* species changed over time?
- What is the average weight of each species, per year?
- What information can I learn about *Dipodomys* species in the 2000s, over time?

What would I need to answer these questions? Which files have the data I need? What operations would I need to perform if I were doing these analyses by hand?
--manually sorting, copying and pasting would probably be the prior approach; Open refine would make this a little neater/quicker using facets/filters
--"Index matching" in excel will allow you to query and extract results to another file (tip from Amanda)
--filter survey.csv file for species DM, DS, and DO (from species.csv) and then plot hindfoot length and weight for these three species

Index-Match, if you need a better way of data matching Excel than just copying and pasting:
https://exceljet.net/index-and-match

Vlookup is another useful Excel command for merging data from another spreadsheet. +1

After importing table data as csv, toggle down to see how table data has been classed (integer, text...); right click on top for Modify table

execute sql tab: here is where we make queries (autocomplete will suggest for you)
ex 1) select * (all) from surveys ; (semicolon is the end of syntax)
press play (triangle) to execute;
other example commands: select year, month, day from surveys; select species from species; (all rows)
+ LIMIT 5 (limit # rows)
+ for distinct records: select distinct species from surveys

Q: Is case important when writing SQL code?
A: the best practice is to use all caps for SQL keywords
but technically, lowercase will work as well
using all caps helps with readability in any SQL code you'll write

Table names and field names should use the same capitalization that they use in the database. For example, you should use SELECT * FROM surveys, not SELECT * FROM sUrVeYs

Q: Can we write our SQL commands in one line?
A: Yes, that will work. However, for the purposes of clarity and readability, it is best to use multiple lines. SQL commands can get really complex, so writing them on one line would be confusing for us

(humans) to read and understand.

Return 2:10 ET

Q: How do I get the results of a SQL query out of DB Browser?
A: There is an icon above the SQL query window with a floppy disk. That will allow you to export your results as a CSV file.

Q: I had some code after the semi-colon in my SQL query and I got an error.
A: A semi-colon is used to indicate the end of a SQL query/command. If you have some code afterwards, the SQL software does not know what you want to do with it and throws an error.

You can write comment lines in your SQL code. These will not be run by the program. (use --)
Commenting is a great idea if you want to save your queries in a file. It makes them easier to understand for collaborators, including your future self!

```
-- This selects only species_id column
SELECT DISTINCT species_id
-- This tells to use surveys table
FROM surveys;
```

Also Windows Notepad in the latest versions can understand UNIX linefeeds as an end of line character.

other commands (functions): ROUND (round to # digits) e.g. select year, ROUND (weight/1000,2) from surveys
[or 1000**.0**] which will force integer into float if your weight field was not identified as real/float but integer instead

Challenge question

- Write a query that returns the year, month, day, species_id and weight in mg.
- Note: weight is currently in grams

select year, month, day, species_id, weight*1000.0 +1
from surveys;

SELECT year, month, day, species_ID, ROUND(weight*1000, 2)
FROM surveys;
SELECT year, month, day, species_id, weight*1000 FROM surveys;

SELECT year, month, day, species_id, weight/1000
FROM surveys
have no idea how to request milligrams- i know it's not there but math isn't working in my brain today
SELECT year, month, day, species_id, weight*1000
FROM surveys;

SELECT year, month, day, species_id, weight * 1000

FROM surveys

SELECT year, month, day, species_id, weight*1000
FROM surveys;

Filtering using WHERE, OR, IN:
WHERE sets conditions (year >=2000); OR adds up conditions, IN creates a list, WHERE (species_id IN ('DM', 'DO', 'DS'); use parens to keep conditions separate

Challenge question

- Produce a table listing the data for all individuals in Plot 1 that weighed more than 75 grams, telling us the date, species id code, and weight (in kg).

Mice only? or all critters?

```
Select year, month, day, species_id, weight/1000
FROM surveys
where (weight >=75) AND (plot_id='1')


SELECT day, month,year , species_id, ROUND(weight/1000, 2)
FROM surveys
WHERE (plot_id='1') AND (weight>75);



SELECT year, month, day, species_id, weight/1000   -- select means choose those columns
FROM surveys  -- from this data file
WHERE (weight>75) and (plot_id=='1');

select year, month, day, species_id, weight
from surveys
Where weight > 75 and plot_id=1;
;
SELECT year, month, day, species_id, weight/1000
FROM surveys
WHERE (weight> 75) AND (plot_id ='1');

select year, month, day, species_id, weight/1000
from surveys
where (plot_id = 1) and (weight > 75)

SELECT year, species_id, weight
FROM surveys
WHERE weight > 75
```

```
SELECT year, month, day, species_id, weight/1000
FROM surveys
WHERE (plot_id = 1) AND (weight > 75);
```

```
select month, day, year, species_id, weight/1000
 from surveys
where (plot_id=1)and (weight>75);
```

```
SELECT year, month, day, species_id, weight/1000, plot_id
FROM surveys
WHERE (plot_id = 1) AND (weight*1000 > 75); << won't this return too many results bc weight is already in g?
```

yeah, it did

sounds like better data management would be to include unit in the header "weight (g)" so this doesn't have to been by implication

Paula´s note: this is a great point and many people do include the units in the column name (I vote for this practice). Alternatively, a good dataset will have its metadata associated. Metadata is the description of what is each column, units, what type of data is included. And with that handy, one could always go and check.

Aggregate functions:

- count (COUNT)
- average (AVG)
- sum (SUM)
- minimum (MIN)
- maximum (MAX)
- , etc.

Challenge question

Write a query that returns: the total weight, average weight, minimum and maximum weights for all animals caught over the duration of the survey. Can you modify it so that it outputs these values only for weights between 5 and 10?

```
SELECT SUM (weight), AVG (weight), MIN (weight), MAX (weight)
FROM surveys
WHERE (weight>5) AND (weight<10);
```

```
Select COUNT (*), count (weight), sum (weight), avg (weight), min (weight), max (weight)
FROM surveys
where (10> weight ) AND (weight >5) -- (but couldn't combine this into 10>weight>5)
```

```
SELECT SUM(weight), MIN(weight), MAX(weight), ROUND(AVG(weight),3)
FROM surveys
WHERE (weight>5) AND (weight<10);
```

```
select sum (weight), avg (weight) min (weight), max (weight)
from surveys
where (weight =>5) and (weight=<10)
```

```
SELECT SUM(weight), AVG(weight), MIN(weight), MAX(weight)
FROM surveys
WHERE 5 <= weight <= 10; --note that this runs but gives incorrect output
```

```
select sum(weight), round(avg(weight),2), min(weight), max(weight)
from surveys where (weight >= 5 and weight <=10);
```

The order that SQL clauses is written matters. We always have to do SELECT __ FROM ___ WHERE ___ etc.

These two queries do not produce the same results:

```
SELECT *
FROM surveys
WHERE 5 <= weight <= 10;
```

```
SELECT *
FROM surveys
WHERE (weight >= 5) AND (weight <= 10);
```

The first query is incorrect. This is because it is not comparing weight to 5 and to 10; it is comparing weight to 5, and storing the result as an integer (0 = false, 1 = true). Then it is comparing this stored integer to 10, and 0 or 1 will always be less than 10.

An interesting corrolary of that is that you can test for non-null values by just listing the column value:

```
select species_id, year, avg(weight), avg(hindfoot_length)
from surveys
where weight or hindfoot_length
group by species_id, year;
```

↑ I would suggest doing an explicit comparison just for clarity's sake, but I guess this could work!
https://www.w3schools.com/sql/sql_null_values.asp

Ordering/grouping and aliases:

```
SELECT species_id, max (year) AS last_surveyed_year-- define a new variable
FROM surveys
GROUP BY species_id
ORDER BY last_surveyed_year --call back this variable
```

Programming maxim - it takes two people to debug any program; only one of them needs to know what the program does.
Trying to explain how something works is the best way to learn how something works as well!

Q: Having vs where?: having is on top of a filter
A: WHERE is used to filter by an existing field. HAVING is used to filter by an aggregated field.
e.g.

```
SELECT species_id, COUNT(species_id) AS species_count
FROM surveys
WHERE species_id = 'DM'
GROUP BY species_id
HAVING species_count > 10;
```

Challenge question

Write a query that returns, from the species table, the number of species in each taxa, only for the taxa with more than 10 species.

```
SELECT taxa, COUNT(taxa) AS occurences
FROM species
GROUP BY taxa
HAVING occurences > 10;
```

~~select taxa, count(taxa) as occurrences~~
~~from species~~
~~group by taxa~~
~~having occurrences > 10;~~

```
SELECT taxa, COUNT(species_id)
FROM species
GROUP BY taxa
HAVING COUNT(species_id) >10
```

```
SELECT taxa , count (taxa)
FROM species
group by (taxa) --- up to here you get a count of how many sp in each taxa (13 birds, 1 rabbit, 9 reptiles, 31 rodents)
having count(taxa) >10 -- this step limits the selection to counts over 10
```

Q: Is it a problem to use the same name for a table and a field in that table? (e.g. the species field in the species table)
A: In general, it's better not to use the same name for a table and for a field in that table. The computer will not be confused, because only table names work in the FROM clause and only field names work in the SELECT clause. But it can be confusing for a human reading the query!

Lesson on joins: https://datacarpentry.org/sql-ecology-lesson/03-sql-joins/index.html

Q: If two tables have multiple overlapping columns (e.g. species_id and year)
A: In general, you probably wouldn't want to do this. This would indicate that you have redundant

information in both tables, and there may be something messy or off about the way your database is set up.
A: If you do want to join on multiple columns, you can use AND
e.g.

```
SELECT *
FROM surveys
JOIN species
ON surveys.species_id = species.species_id
   AND surveys.year = species.year;
```

Q: Can you give the new table an alias too?
A: Yes, you can give the resulting table a new alias. This can be useful when you do subqueries inside other queries :-)

Q: What if you want the average weight in two columns by gender?

```
SELECT plots.plot_type, surveys.sex, AVG(surveys.weight)
FROM surveys
JOIN plots
ON surveys.plot_id = plots.plot_id
GROUP BY plots.plot_type, surveys.sex;
```

Q: How would you spread a column's values (e.g. sex) into individual columns (e.g. male, female, unspecified)?
A: It's slightly complicated. Here's one way you can accomplish this:
https://stackoverflow.com/questions/22177982/sql-select-to-spread-row-data-to-columns
A: In general, having a separate column for sex and for the corresponding count is a more tidy way of storing the data. You can then pivot it with SQL or with another program (Excel, R, etc.) but for some applications (aggregating, plotting, filtering) it will be easier to work with if the data are not spread into multiple columns.

Challenge question

- Write a query that returns the number of animals caught of each genus in each plot. Order the results by plot number (ascending) and by descending number of individuals in each plot.

```
--to start: joining surveys x species
select *
from surveys
join species
on surveys.species_id= species.species_id -- same effect as "USING (species_id)" --[this was not necessary]

--selection can call variables from both tables, AND a count of one of those variables, so replace:
select surveys.plot_id, species.genus, Count (species.genus)
--add grouping
group by surveys.plot_id, species.genus
```

```
--add ordering
order by plot_id ASC, count (species.genus) DESC

SELECT surveys.plot_id,species.genus, COUNT(*) as nanimals
FROM surveys
JOIN species
ON surveys.species_id=species.species_id
GROUP BY species.genus, surveys.plot_id
ORDER BY surveys.plot_id ASC, nanimals DESC
;
```

Feedback survey: https://docs.google.com/forms/d/e/1FAIpQLSf4KDz3udu-B4b4U_2lQf87o8txvtN_isCrABkkPqRpdf_PvA/viewform?usp=sf_link

----

R Day 1

Wednesday, May 25, 2021

Workshop Links

Installation instructions for R and RStudio: https://datacarpentry.org/ecology-workshop/setup-r-workshop.html

RStudio Binder: https://mybinder.org/v2/gh/SmithsonianWorkshops/binders/rstudio?urlpath=rstudio
(Use this if you are having trouble installing R and RStudio)

Full Portal Project dataset: https://ndownloader.figshare.com/articles/1314459/versions/10

Read surveys data directly into R:
```
surveys <- read_csv("https://ndownloader.figshare.com/files/10717186")
```

Lessons:
R, RStudio, and R Markdown: https://datacarpentry.org/R-ecology-lesson/00-before-we-start.html
Intro to R: https://datacarpentry.org/R-ecology-lesson/01-intro-to-r.html
Starting with data: https://datacarpentry.org/R-ecology-lesson/02-starting-with-data.html

Additional Resources

Paula's R Markdown tutorial: https://github.com/paulapappalardo/BrownBag-rmarkdown/tree/master

RStudio cheat sheets: https://www.rstudio.com/resources/cheatsheets/

RStudio keyboard shortcuts (Windows & Linux)
Insert R code chunk: Ctrl + Alt + I
Run current code chunk: Ctrl + Shift + Enter
Insert assignment operator (<-): Alt + -
Insert pipe (%>%): Ctrl + Shift + M

RStudio keyboard shortcuts (macOS)
Insert R code chunk: Command + Option + I
Run current code chunk: Command + Shift + Enter
Insert assignment operator (<-): Option + -
Insert pipe (%>%): Command + Shift + M

Attendance

Please write your name and an example of a project relevant to your work or your interests where you could use both OpenRefine and SQL.

Amanda Devine: I could download NMNH collection records and use OpenRefine filtering and faceting to clean up and standardize the country names where each specimen was collected. I could also download a dataset containing the size of each country in the world (land area). Then I could import both files into a SQL database, join the two tables, group by country, and calculate the density of collecting (total specimens/land area) to represent which countries had been most thoroughly sampled.

Jon Barrett - Use SQL and OpenRefine to extract useful information from TMS for phot oprocessing (GUIDs for instance).

Henrique Vieira Figueiró - Use SQL to create a master table where I can have all the information necessary for several analyses.

Ida Hartvig - Use both tools to organize and clean up large collection database of DNA samples and fungal cultures, which is currenlty managed just as an excel sheet
Beverly Lamberson - data cleansing of a system to identify dups, missing data, etc. In particular for our system that has events for our DC and NY locations in separate segment of the database; need to identify dups within each side as well as across both.
Robin O'Hern - i could use open refine to clean up data from the collecitons databases and SQL to draw connections between that data and information gathered through a survey.

Rachel Collin- I could combine environmental monitoring data with counts of organisms.

Tricia Goulding - Use OpenRefine to clean up species names in a spreadsheet before exploring and summarizing data on different groups of taxa in SQL.

Sarah Alley - Use both SQL and OpenRefine to filter and extract data from specific plots and sampling dates for extracellular enzyme activity data

Irene Holst Use both to extract and filter data relating to our reference collections of Phytoliths and Starch grains

Jeff Sosa

Helene Muller-Landau. OpenRefine could be useful for cleaning up tree plot data. To be honest, I don't see anything that SQL can do that I wouldn't rather do in R, considering that I know how to use R already.

Rayna Andrews - Use both to collect and filter collection metadata

Gemma Cantlow - I could use SQL and OpenRefine to extract and filter data relating to Learning Lab collections, e.g. subject areas, types of collections, age ranges

Jess Walthew- review data from a past plastics survey to identify items listed in poor condition to resurvey. will need open refine to manage non-controlled vocabularies for search terms

Amy Driskell - use SQLite to combine spreadsheets. Use OpenRefine to tidy up spreadsheets.

Ashraf Moursi - manipulating large sets of orbital events data

Chad Kamen - editing lengthy metadata sheets!

Questions for today:

Q: What are the differences between workspace and R projects
A: The workspace is what holds your objects, functions, all the things you are working with for a particular task. You can choose to save it so that it reloads everytime you want to use it. It is not recommended though, in terms of reproducibility. Ideally your script (set of instructions) will load the data you need, have all the instructions, and generate what you need and save it. It does help with organization to not save the workspace and "nudge" you to be careful that all you need is on the R script. An R project, is meant to organize all the files you need for a specific project, and the huge advantage is that it automatically sets up a "we are here", in a way that you can refer directly to subfolders there without having to write a very long path. That is why it is so useful to share with collaborators. When you share a folder that is an R project, the person opening it will have access to the same file organization you see in your computer, without having to use a personal looking path. Here you can see more info: https://r4ds.had.co.nz/workflow-projects.html

Q: Is there a difference between using "AND" or "&"?
A:I think it is a matter of language. SQL and Python uses "and" while R uses "&".
R operators https://stat.ethz.ch/R-manual/R-devel/library/base/html/Logic.html
More info here: https://stackoverflow.com/questions/16027840/whats-the-differences-between-and-and-in-r#:~:text=The%20docs%20says%3A,first%20element%20of%20each%20vector.

Q: is it the same to use single quote than double quote
A: more info here
https://stat.ethz.ch/R-manual/R-devel/library/base/html/Quotes.html#:~:text=Descriptions%20of%20the%20various%20uses%20of%20quoting%20in,constants.%20Single%20and%20double%20quotes%20delimit%20character%20constants

Markdown syntax: https://rmarkdown.rstudio.com/authoring_basics.html

Challenge:

A. Create two variables: mass (with a value of 47.5) and age (with a value of 122).
B. Run the following code. What are the end values of mass, age, and mass_index?
C. What does this tell you about how R assigns values to variables?

```
mass <- mass * 2.0
age  <- age - 20
mass_index <- mass/age
```

Challenge:

Using the animals vector (animals <- c("mouse", "rat", "dog", "cat")), what do the following commands return?
A. animals[c(1, 1, 1, 4, 4, 4)]
B. animals[2:4]
C. animals[-1]
D. animals[c(TRUE, FALSE, TRUE, FALSE)]

```
[1] "mouse" "mouse" "mouse" "cat"   "cat"   "cat"
[1] "rat" "dog" "cat"
[1] "rat" "dog" "cat"
[1] "mouse" "dog"   #https://www.youtube.com/watch?v=54Afdxd6sUQ
# maybe next time we will play the catdog song for this challenge :)
```

Feedback form: https://docs.google.com/forms/d/e/1FAIpQLSf4KDz3udu-B4b4U_2lQf87o8txvtN_isCrABkkPqRpdf_PvA/viewform

----

R Day 2

Thursday, May 26, 2021

Workshop Links

Installation instructions for R and RStudio: https://datacarpentry.org/ecology-workshop/setup-r-workshop.html

RStudio Binder: https://mybinder.org/v2/gh/SmithsonianWorkshops/binders/rstudio?urlpath=rstudio
(Use this if you are having trouble installing R and RStudio)

Read surveys data directly into R:
```
surveys <- read_csv("https://ndownloader.figshare.com/files/10717186")
```

Lessons:
Starting with data: https://datacarpentry.org/R-ecology-lesson/02-starting-with-data.html
Data manipulation with dplyr and tidyr: https://datacarpentry.org/R-ecology-lesson/03-dplyr.html
Data visualization: https://datacarpentry.org/R-ecology-lesson/04-visualization-ggplot2.html

Additional Resources

Paula's R Markdown tutorial: https://github.com/paulapappalardo/BrownBag-rmarkdown/tree/master

RStudio cheat sheets: https://www.rstudio.com/resources/cheatsheets/

RStudio keyboard shortcuts (Windows & Linux)
Insert R code chunk: Ctrl + Alt + I
Run current code chunk: Ctrl + Shift + Enter
Insert assignment operator (<-): Alt + -
Insert pipe (%>%): Ctrl + Shift + M

RStudio keyboard shortcuts (macOS)
Insert R code chunk: Command + Option + I
Run current code chunk: Command + Shift + Enter
Insert assignment operator (<-): Option + -
Insert pipe (%>%): Control + Shift + M or Command + Shift + M

Attendance Day 2
Please add your name and which software you are likely to want to use and learn more 1) OpenRefine, 2) SQL, 3) R. You can mark more than one :)

-Paula: 1) for checking species names, cool clustering algorithms; 2) to store a large database; 3) for everything else :)
-Jose: R and SQL
-Amy: all three
-Rayna: definitely likely to use OpenRefine. Want to learn more SQL and R!
-
-Rachel - Definitely R, since you can apparently do pretty much anything in R.
-Henrique V. Figueiró - I'll be using R and SQL

-Beverly Lamberson, plan to look into all of them and decide which will be best for different needs
-Sarah Alley: R since I am already working on several projects using R and I would really like to learn more
-Tricia Goulding - R since there are so many different tools to use
Jon Barrett 1) R/RStudio 2) SQL 3) Whatever works for the particular need
iIrene Holst all three
-Chad Kamen - 1) OpenRefine has already been so helpful for my work!
Ashraf Moursi - all
- Helene Muller-Landau. OpenRefine and R
Jess Walthew- OpenRefine and R
Gemma Cantlow - R
Robin O'hern - all three!

Challenge
Using pipes, subset the surveys data to include animals collected before 1995 and retain only the columns year, sex, and weight.

```
surveys %>%
    filter(year<1995) %>%
    select(year, sex, weight)

surveys%>%
  filter(year < 1995)%>%
  select(year, sex, weight)

surveys %>%
  filter(year <1995) %>%
  select(year, sex, weight)

surveys %>%
  filter(year < 1995) %>%
  select(year, sex, weight)

surveys  %>%
  filter(year < 1995)  %>%
  select(year, sex, weight)
```

Create a new data frame from the surveys data that meets the following criteria: contains only the species_id column and a new column called hindfoot_cm containing the hindfoot_length values converted to centimeters. In this hindfoot_cm column, there are no NAs and all values are less than 3.

Hint: think about how the commands should be ordered to produce this data frame!

```
surveys %>%
  mutate(hindfoot_cm=hindfoot_length/10) %>%
  select(species_id, hindfoot_cm) %>%
  filter(!is.na(hindfoot_cm)) %>%
  filter(hindfoot_cm < 3)

surveys %>%
  mutate(hindfoot_cm = hindfoot_length /10) %>%
  select(species_id, hindfoot_cm) %>%
  filter(!is.na(hindfoot_cm), hindfoot_cm < 3)

surveys %>%
  filter (!is.na(hindfoot_length)) %>%
  mutate(hindfoot_cm =hindfoot_length/10) %>%
  filter(hindfoot_cm <3) %>%
  select(species_id, hindfoot_cm)

surveys %>%
  select(species_id, hindfoot_length) %>%
  mutate(hindfoot_length_cm = hindfoot_length / 10) %>%
  filter(!is.na(hindfoot_length_cm), hindfoot_length_cm < 3)

surveys %>%
  mutate(hindfoot_cm = hindfoot_length/10)%>%
  filter(hindfoot_cm <3 & !is.na(hindfoot_cm)) %>%
select(species_id, hindfoot_cm)
surveys %>%
filter(! is.na(hindfoot_length)) %>%
  filter(hindfoot_length <30) %>%
mutate( hindfoot_cm = hindfoot_length/10) %>%
  select(species_id,hindfoot_cm)

  surveys %>% mutate(hindfoot_cm =hindfoot_length/10) %>% filter(!is.na(hindfoot_cm) &
hindfoot_cm<3) %>% select(species_id,hindfoot_cm)
```

It's possibly more efficient, especially with large data sets, to filter everything and then do the additional processing.

Challenge
How many animals were caught in each plot_type surveyed?

```
surveys %>%
  count(plot_type)
```

Questions for today:

Q:How do you know when to put %>% at the end of a line?
A:The %>% connects different select, filter and mutates. When you’re done combining those, you don’t

put a %>%
You need to put always the %>% end of the line that way R understand that there is more code coming

Q:short cut to %>%
A:Pipe shortcut : control + shift + m (Mac, Windows)
                         cmd+shift+m (Mac)

Q:Could you clarify the difference between na.rm and !is.na?
A:na.rm is an argument (an input to a function) for functions that tells those functions to ignore NA values
is.na() is a standalone function that returns a logical vector of Boolean (TRUE/FALSE) values
!is.na() returns a logical vector where every non-NA value is TRUE, so it can be used to filter and get just the non-NA values
e.g. These two lines of code should produce the same results

```
weight_g <- c(2, 4, 4, NA, 6)
mean(weight_g, na.rm = TRUE)
mean(weight_g[!is.na(weight_g)])
```

Q: Is there any convention that it's "nicer" coding to use fewer lines?
A: I think in general people strive to balance efficiency and clarity. Sometimes if you write many lines of code, you might be doing a lot of extra work that could be done by calling one function. But sometimes, if you end up having multiple nested functions inside one another, it would actually be better to write multiple lines of code instead.
Q: but so for example, in our earlier challenge I did a separate filter step to remove the null values before performing any other work. this makes sense to me and it how i'd do it "manually", vs. filtering them out later after performing other tasks. So technically for a very big data set this pre-filtering would be more "efficient" even if the code itself doesn't look tidier, right?
A: Right - filter first and there'll be fewer rows to do the calculations and add columns on. If you filter last, you'll multiply out and add the nee column on every row, then throw a lot of them away.
A: I think that's a good consideration. Code that runs efficiently is important, esp. if you plan on scaling the data set up.

Q:
A:

```
surveys %>%
  count(plot_type) #this seemed too easy :)

surveys %>%
  group_by(taxa) %>%
  count(plot_type)
```

Data Visualization (ggplot2)

Lesson website: https://datacarpentry.org/R-ecology-lesson/

Resource:
Thomas Lin Pedersen
"ggplot2 workshop part 1"
https://www.youtube.com/watch?v=h29g21z0a68

Starting dataset:

surveys_complete <- read_csv("https://smithsonianworkshops.github.io/2021-05-19-smithsonian-online/surveys_complete.csv")

Colors in R:
http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Q&A

Q: The variables should be inside the aesthetics (aes) function?
A: Yes. When we create a ggplot, we use aesthetic mapping to tell ggplot which variables we would like to use to create the plot.
ggplot(data = surveys_complete, mapping = aes(x = weight, y = hindfoot_length))

Q: how do we add a filter to look only at some species types? do we first use pipes to set a new filtered survey_complete file and call that instead or can you do it within the ggplot itself?
A: Three ways:

1. you can create a new filtered version of surveys_complete and use that to plot

```
surveys_complete_filtered <- surveys_df %>% filter(species != 'NL')
ggplot(data = surveys_complete_filtered, mapping = aes(...))
```

2. you can use pipes to filter the data and pipe it right into ggplot. If you do this, you don't need to specify the data argument in your ggplot function

```
surveys_df %>%
filter(species != 'NL') %>%
ggplot(mapping = aes(...))
```

3. you can use the subset() function to filter the data in the ggplot function

```
ggplot(data = subset(surveys_df, species != 'NL'), mapping = aes(...))
```

https://www.datanovia.com/en/blog/how-to-subset-a-dataset-when-plotting-with-ggplot2/

Challenge
Use what you just learned to create a scatter plot of weight (for y) over species_id (for x) with the plot types showing in different colors. Is this a good way to show this type of data?

```
ggplot(data=surveys_complete,mapping=aes(x=species_id,y=weight)) +
geom_point(alpha=0.1,aes(color=plot_type))
```

No, it's not a good way to show these data. Would be better to have separate panels for species (or plots), and to use violin plots.

```
ggplot(data = surveys_complete, mapping = aes(x = species_id, y = weight)) +
  geom_point(alpha = 0.1, aes(color = plot_type))

ggplot(data = surveys_complete, mapping = aes(x = species_id, y = weight)) + geom_point(alpha = 0.1,
aes(color = plot_type))

ggplot(data = surveys_complete, mapping=aes(x=species_id, y=weight))+ geom_point(aes(color =
plot_type))

ggplot(data= surveys_complete, mapping =aes (x=species_id, y=weight)) +
  geom_point(alpha =0.6, aes(color=plot_type)) #ugly graph
```

~~~~~
```{r}
ggplot(data =surveys_complete, mapping = aes(y=weight, x=species_id)) +
  geom_point(aes(color = plot_type))
```


```{r}
ggplot(data=surveys_complete, mapping=aes(x=species_id, y=weight))+
  geom_point(alpha=0.1, aes(color=plot_id))
````
~~~~~

```
ggplot(data = surveys_complete, mapping = aes(y= weight, x=species_id)) +
  geom_point(alpha = 0.1, mapping = aes(color=plot_type))
```

Starting with this plot:

```
ggplot(data = surveys_complete, mapping = aes(x = species_id, y = weight)) +
  geom_jitter(alpha = 0.3, color = "purple") +
  geom_boxplot(alpha = 0)
```

Set the point color in the jitter plot based on plot_type

```
  ggplot(data = surveys_complete, mapping = aes(y= weight, x=species_id)) +
  geom_jitter(alpha = 0.3, aes(color = plot_type)) +
  geom_boxplot(alpha = 0)

  ggplot(data= surveys_complete, mapping =aes (x=species_id, y=weight)) +
  geom_jitter(alpha = 0.5, aes(color=plot_type)) +
  geom_boxplot(alpha =0)

 # this is a more interesting graph: using color for species ID and using plot_type as the x
 ggplot(data= surveys_complete, mapping =aes (x=plot_type, y=weight)) +
```

~~~
  geom_jitter(alpha = 0.5, aes(color=species_id)) +
  geom_boxplot(alpha =0)

```{r}
ggplot(data =surveys_complete, mapping = aes(y=weight, x=species_id)) +
  geom_jitter(alpha=0.3, aes (color =plot_type))+
  geom_boxplot(alpha=0)

```
ggplot(data = surveys_complete, mapping = aes(x = species_id, y = weight)) +
  geom_jitter(alpha = 0.3, aes(color = plot_type)) +
  geom_boxplot(alpha = 0)
~~~

Feedback form: **https://docs.google.com/forms/d/e/1FAIpQLSf4KDz3udu-B4b4U_2lQf87o8txvtN_isCrABkkPqRpdf_PvA/viewform?usp=sf_link**