
Additional workshops to attend

November Software Carpentry with Python (https://software-carpentry.org/lessons/)

- Eventbrite link: https://www.eventbrite.com/e/accelerate-precision-health-czif-software-carpentry-workshop-tickets-896511298427?aff=oddtdtcreator

Please feel free to email workshops@carpentries.org with questions as you work through the material, or questions about the upcoming events.

Data Carpentry Workshop

Chan Zuckerberg Initiative Foundation Accelerate Precision Health

14-17 October October 11:00 am-3:00 pm Eastern Time (UTC -4)

Check the time in your region at https://www.timeanddate.com/worldclock/fixedtime.html?msg=Data+Carpentry+%40+Chan+Zuckerberg+Initiative+Foundation+Accelerate+Precision+Health&iso=20241014T11&p1=3918&ah=4

Zoom Link: https://carpentries.zoom.us/my/carpentriesroom3 password 202020
Workshop Website Link: https://klgallagher.github.io/2024-10-14-carpentriesCZI-online/

Workshop Host: Danielle Sieh, The Carpentries
Instructors: Katherine Gallagher, Yuanxi Fu, Gracielle Higino, Jennifer (Jay) Gordon

Helpers: Vitor Pavinato, Xochitl Ortiz Ross, Ann James

Day One

Minute Card: https://docs.google.com/forms/d/e/1FAIpQLSeeorEiTXrUEcxK-yWm4VnuRET2EJZ3Q9-dMEuwWa-OaVCXbA/viewform?usp=sf_link
Attendance:

- Harriet Blankson
- Kenya Bynes
- Ann Myatt James, she/they, Helper, Mac user ajames31@gwu.edu
- Kenyaita Hodge
- Gracielle Higino, she/her, Instructor graciellehigino@gmail.com
- Victor Eno
- Anzo Panem
- Cheryl
- Fernando Marques Romero
- Rema Gesaka
- Cheryl Cropp

Data Organisation in Spreadsheets: Find the full lesson to follow along and learn on your own at https://datacarpentry.org/spreadsheet-ecology-lesson/

Monday, 14 October 2024: Spreadsheets

Complete lesson: https://datacarpentry.org/spreadsheet-ecology-lesson/
Messy file for download: **Download** this data file to your computer: https://ndownloader.figshare.com/files/2252083

Survey sorting exercise dataset: https://github.com/datacarpentry/spreadsheet-ecology-lesson/blob/gh-pages/data/survey_sorting_exercise.xlsx?raw=true

Questions:

- What should you do with blank data
  - Sometimes missing values from datasets you download may use NA, blanks, -9999, or special characters (like . —). These data may need cleaning prior to use
  - the important part is making sure that blanks are missing data and not zeros, which would be data
-
-
-

Notes

- Never edit your original file. Always save a new copy.
- Every variable goes in its own column. Each observation gets its own row.
- Each cell should only contain 1 piece of information. For example, don't include units in the cell with the number, keep it in the header.
- A common format to save files = .csv (comma-separated values). Many softwares will be able to read this format.
- When cleaning and tidying your datasets, create a "clean data" spreadsheet, copy the raw data over, and reshape it as needed. **Avoid doing anything with the raw data**, as it is the original source of information that you can come back to in case something goes wrong.
- It's good practice to remove any formating (and they may not be kept when saving as .CSV); instead, if you have "color coding", for example, transform that into text or numeric information in a dedicated column.
- It's good practice to split dates into as many units as possible
    - 2013-07-16 should become three columns: one for the year, one for the month and one for the day.
- If you have codes or acronyms, you can create a READ ME text file or separate spreadsheet that defines what each variable is and how the data was collected etc.
    - a separate tab also ensures the current spreadsheet can be seamlessly read into programs like R
    - data “legends” may also be referred to as “metadata”, which means data that explains the data
- Important to note: .csv files will not save different “tabs”, but you can select 1 tab to save
- This lesson takes you through formatting problems: https://datacarpentry.org/spreadsheet-ecology-lesson/instructor/02-common-mistakes.html
- You can split dates by using the functions:
    - =YEAR(date_cell); e.g. =YEAR(A3) if the date is in cell A3
    - = MONTH(date_cell)
    - = DAY(date_call)
    - You can change the top row and then drag down to auto-fill the rest of the cells below.
- The CSV file extension is also considered to be the recommended extension for data sharing and preservation in data repositories
- Quality Assurance information: https://datacarpentry.org/spreadsheet-ecology-lesson/instructor/04-quality-control.html

Data Cleaning with OpenRefine for Ecologists

Complete lesson: http://datacarpentry.org/OpenRefine-ecology-lesson/index.html

Setup

- Detailed instructions: http://datacarpentry.org/OpenRefine-ecology-lesson/index.html#setup
- Download and install OpenRefine: https://openrefine.org/download.html
    - Windows users should download the “Windows (with Java)” version from the Download

    page to avoid having to install Java
    - Mac users can use the kit available in the Download page, Java does not have to be installed separately.
    - Linux users must install Java in their system.
- Download this data to play with: http://datacarpentry.org/OpenRefine-ecology-lesson/data/Portal_rodents_19772002_simplified.csv

Notes

- Use Facets and Filters to select subsets of your data to act on
    - can identify errors or outliers in data
- Undo / Redo tab has a list of steps you have completed during the project
    - You can go back to any state of your data at any time
- You can split columns that contain multiple information by defining what should be used to separate the information (e.g., " " indicates a blank space). Use Edit column > Split into several columns...
- Rename columns using Edit column > Rename this column
- You can join multiple columns into a single column, like to join day, month, and year into a single Date column. Edit column > Join columns, select which columns you want to combine, define how you want to separate the information from each columns (e.g., "-"), > Write result in new column named: name your new column.
- Data colustering allows you to detect data points that are written differently (typos, abbreviations) but are meant to be the same. Edit cells > cluster and edit
    - It uses different methods and algorythms, you can play around with these and see if some of the combinations find any data clusters. More info: https://openrefine.org/docs/technical-reference/clustering-in-depth
- Faceting is an exploration of the data, and overview, filtering reduces the number of rows in your dataset
- Employ *text filter* or *include/exclude* to filter to a subset of rows.
    - if you facet scientificName, you can see "edit" or "include" next to each species name. You can use this to choose which species to include in your filter. Once you click on include, you can see that the button will change to exclude if you change your mind.
    - you can also filter via text search (text filter).
- Sort tables by a column.
    - drop-down menu of a column > sort > choose how you want to sort
- Sort tables by multiple columns.
    - you can also sort by multiple columns. Sort one column, then another. The order in which things are sorted will match the order in which you sorted the columns.
    - When you already have a column sorted, it also gives you the option to "sort by this column alone". You can use this to re-start sorting or change the order of the sorting.
- Add external reconciliation services.
    - Reconcile > Start reconciling
    - Add standard service...: https://eol.org/api/reconciliation to add the reconciliation service for the encyclopedia of life
- Cleanup scientific names by matching them to a taxonomy.
    - Reconcile > Start reconciling > Encyclopedia of life > Start reconciling
- Identify and correct misspelled or incorrect names for a taxon.

Questions

-
-
-
-
-

Day Two

Minute Card:
https://docs.google.com/forms/d/e/1FAIpQLSfybTq9anPQO4SRlhpG1ZNWRJh0B4MF65QSzP9SmZroIn3ljw/viewform

Attendance:

- Gracielle Higino
- Harriet Blankson
- Kenya Bynes
- Fernando Marquez Romero
- Enzo Panem
- Victor Eno
- Kenyaita Hodge

Open Refine

Notes

- All changes are being tracked in OpenRefine, and this information can be used for scripts for future analyses or reproducing an analysis (json file).
    - You can also apply the same cleaning steps on a new dataset
- OpenRefine can save the clean data to a number of formats.
- Cleaned data or entire projects can be exported from OpenRefine.
- Projects can be shared with collaborators, enabling them to see, reproduce and check all data cleaning steps you performed.

Questions

- How can you find the URLs to use when fetching information from URL?
    - You need to look for databases that have API access and look for instructions in their documentation.
    - http://datacarpentry.org/OpenRefine-ecology-lesson/07-looking-up-data.html

SQL

You can follow directions to download the data and install the software from the set up page:
https://datacarpentry.org/sql-ecology-lesson/index.html
Follow along with the lesson: https://datacarpentry.org/sql-ecology-lesson/00-sql-introduction.html

Notes

- DB browser SQLite: https://sqlitebrowser.org/dl/
- where to download data: https://figshare.com/articles/dataset/Portal_Project_Teaching_Database/1314459
- Please write your name: anything about your experience with database
- Danielle Sieh - I work with a database every day. Excited to learn more!
- Kenyaita Hodge - I have very limited experience. I have used Access to create databases but that was a while ago.
- Enzo Panem: limited experience to matlab hw in class during undergrad
- Victor Eno. very limited experience with databases.
- Relational databases help get rid of redundancies in your data and keep everything updated
- SQL stands for Structured Query Language. SQL allows us to interact with relational databases through queries. These queries can allow you to perform a number of actions such as: insert, select, update and delete information in a database.
- Goal: How do I select/find the exact data that I need to answer my research questions?
    - select subsets of the data (rows and columns)
    - group subsets of data
    - do math and other calculations
    - combine data across spreadsheets
- A relational database is made up of tables which are related to each other by shared keys.
- In "Execute SQL" you can insert comments/notes that are not code by starting with " -- "
- Different queries on the same page can be differentiated by adding a " ; " at the end of each query.
- Use the " * " to indicate " all " (e.g., select every column)
- To save your SQL file click on the 3rd icon from the left on the bar which says "save SQL file" and that will save your script.
- Click on "Save the results view" to save the output of your query. You can save it as .CSV.

EXAMPLE QUERIES:

- -- **select** the year column from surveys

- SELECT year
- FROM surveys;

- -- you can also select multiple columns

- SELECT year, month, day
- FROM surveys;

- -- Sometimes you don’t want to see all the results, you just want to get a sense of what’s being returned. In that case, you can use a **LIMIT** clause. In particular, you would want to do this if you were working with large databases. It is most useful when combined with **ORDER.**

- SELECT *
- FROM surveys

- LIMIT 5;
- -- use **DISTINCT** to find unique/specific values
- SELECT DISTINCT species_id
- FROM surveys;
- -- you can *filter* your data using **WHERE** to match certain specifications. Use **AND** to include multiple requirements/specifications.
- SELECT *
- FROM surveys
- WHERE(plot_id=1)AND(weight>75)
- -- this one returns the survey records that were recorded on plot #1 and had a weight >75.

Questions

- Is the semicolon necessary at the end? Not in SQLite, but it might be in other databases.

Day Three

Workshop Host: Danielle Sieh, The Carpentries
Instructors: Yuanxi Fu, Jennifer (Jay) Gordon
Helpers: Xochitl Ortiz Ross, Ann James

Minute Card:
https://docs.google.com/forms/d/e/1FAIpQLSdepdq33yjMMJT9JrOTJLqpychghlcvLpYInBi_CtW3QX6niw/viewform?usp=sf_link

Attendance:

- Kenya Bynes
- Harriet Blankson
- Ann Myatt James (Helper)
- Kenyaita Hodge
- Fernando Marquez Romero
- Victor Eno
- Enzo Panem

Data Analysis and Visualization in R for Ecologists Part 1

Follow along at: https://datacarpentry.org/R-ecology-lesson/
Keyboard shortcuts in RStudio: https://support.posit.co/hc/en-us/articles/200711853-Keyboard-Shortcuts-in-the-RStudio-IDE
ggplot2 color codes: https://sape.inf.usi.ch/quick-reference/ggplot2/colour
Data Visualization CheatSheet: https://rstudio.github.io/cheatsheets/data-visualization.pdf
Resource related to the use of quotations (single quote vs. double quote) in RStudio: https://stat.ethz.ch/R-manual/R-devel/library/base/html/Quotes.html

Wrapping code in RStudio Quick Tip: https://stackoverflow.com/questions/39901028/wrap-code-in-r-studio-text-editor

Introduction to R and RStudio - Notes

- R is a programming language and software used to run commands in that language
- RStudio is software to make it easier to write and run code in R
- Use the help tab, bottom right, to look for helpful documentation
- Console vs. Script areas: you can type code in the console, this is also where the output of your code will show up, however you generally don't save this part. To save your code, write it in the script document above.
- Write your code in scripts for reproducibility and portability
- Make comments in your script by using " # " in front of the text (like " -- " for SQL)
- The Run button will perform the current line of the script. You can also use "Command+enter" for Mac or "Control+enter" for Windows. See more keyboard shortcuts in the link above.
- install.package("package_name") --> to install a package
- Once you install a package, you will then need to load it using: library(package_name)
- You can check whether your package was loaded properly by checking the “package” tab, finding the package and noticing whether there is a checkmark next to it

Data visualization with ggplot2

- View(data) --> opens up a table with your data so you can look at it
- str(data) --> shows what the structure of the data looks like
- Plot data using the package: ggplot2
- First tell ggplot what data you want to use:
- ggplot(data = complete_old)
- Then you want to add some aesthetics, like what the x and y axes are
- ggplot(data = complete_old, mapping = aes (x = weight, y = hindfoot_length))
- You then need to tell it what kind of graph you want by adding the following at the end of the previous code:
- + geom_point()
- ggplot uses "+" to add *layers* to your graph
- To add some transparency to the dots use "alpha = "
- geom_point(alpha = 0.2)
- Add color by using "color = "
- geom_point(color = "blue")
- Find what color names ggplot recognizes by using this link: https://sape.inf.usi.ch/quick-reference/ggplot2/colour
- To make a box plot instead of a scatterplot use:

- geom_boxplot()
- You can color your boxes based on a specific variable by specifying that in color = variable (for the outline) or fill = variable (to fill in the color)
- ggplot(data = complete_old, mapping = aes (x = weight, y = hindfoot_length, **color = plot_type**)) + geom_boxplot()
- Notice that we are specifying the variable to color by in the aes argument of ggplot
- To wrap the legend labels add a new layer (+) of
- scale_x_discrete(labels = label_wrap_gen(width = 10))
- If you want to see the data points on top of your boxes you add a new layer (+)
- geom_jitter(alpha = 0.2)
- But now the points are covering up the boxes and it doesn't look pretty. We can change the order of the layers so that the boxplots are on top. Move the boxplot() after geom_jitter(). That works... but it looks ugly and it now covers the dots! Let's switch the colors so that the points are what is colored. To do this, we are going to get rid of the color = plot_type from the overall aes() to the geom_jitter.
- ggplot(data = complete_old, aes(x = plot_type, y = hidfoot_length)) +
- geom_jitter(alpha = 0.2, aes(color = plot_type))
- We also don't need to represent the outliers in the boxplot if we are showing the data points
- geom_boxplot(outlier.shape = NA)
- Final code:
- ggplot(data = complete_old, mapping = aes(x = plot_type, y = hindfoot_length)) + geom_jitter(aes(color = plot_type), alpha = 0.2) + geom_boxplot(outlier.shape = NA, fill = NA)

Questions

-
-
-

Day Four

Workshop Host: Danielle Sieh, The Carpentries
Instructors: Katherine Gallagher
Helpers: Xochitl Ortiz Ross

Minute Card:
https://docs.google.com/forms/d/e/1FAIpQLSewjelRB5Z6ufYWq43rR2fpjhjfEaXRLmY1E4ucEHKs7L

H_Hw/viewform

Post-Workshop Survey:
https://carpentries.typeform.com/to/UgVdRQ?slug=2024-10-14-carpentriesCZI-online

Attendance:

- Katherine Gallagher, Instructor, she/hers
- Kenya Bynes
- Victor Eno
- Harriet Blankson
- Kenyaita Hodge
- Enzo Panem

Data Analysis and Visualization in R for Ecologists Part 2

https://datacarpentry.org/R-ecology-lesson/how-r-thinks-about-data.html#vectors-the-building-block-of-data

Notes

Vectors

- Vectors are lists of numbers or characters. Dataframes are made up of several vectors (each column is a vector).
- Integer or numeric vectors are made up of numbers.
- int <- c(1, 2, 5, 12, 4)
- Character vectors are made up of character strings (words).
- char <- c('apple', 'pear', 'grape')
- Logical vectors are made up of boolean data (TRUE or FALSE).
- logi <- c(TRUE, FALSE, TRUE, TRUE)
- When the vectors appears to be "mixed", R uses a hierarchy to decide what kind of vector it is (vectors can only have 1 class of values)
- num_logi <- c(1,4,6,TRUE)
- R interprets TRUE as 1 and FALSE as 0. But it doesn't know how to interpret 4 or 6 in logical terms. Therefore, numeric takes precedence over logical.
- Character is treated before numeric, so if any value in your string is a character, the whole vector is turned into character.
- Character > numeric > logical
- Vectors can also have missing data (NA)
- weights <- c(25, 34, 12, NA, 42)
- R understands that NA means there is a missing value, it doesn't try to coerce it into a specific class.
- But when there are missing values, R can't do math.

- min(weights) returns NA
- If we want to do some math, we need to remove the NAs with na.rm=T
- min(weights, na.rm = T) --> T stands for TRUE and R knows this.
- quantile(complete_old$weight, probs = 0.25, na.rm = T)
- complete_old$weight is a vector being used as an argument in a function (quantile)
- Use ? in front of any function to pull up the help page. E.g., ?quantile
- There are many different ways to make vectors:
    - 1:10 -- gives you the sequence from 1 to 10 (that's what the : means)
    - seq(from = 1, to = 10, by = 2) -- gives you a sequence of numbers from 1 to 10 counting by 2. So: 1, 3, 5, 7, 9
    - seq(from = 0, to = 1, length.out = 50) -- give you 50 evenly spaced numbers between 0 and 1
    - rep("a", times = 12) -- gives you "a" repeated 12 times.
    - rep(c("a", "b", "c"), times = 4) -- repeats the vector a, b, c 4 times: a, b, c, a, b, c, ...
    - rep(c("a", "b", "c"), each = 4) -- repeats each value in the vector 4 times: a, a, a, a, b, b, b, b...
    - You can also combine rep() and seq() to make longer vectors: rep(seq(from = -3, to = 3, by =1), times = 3)
- Almost everything in R is made up of vectors or uses vectors. So we can combinbe vectors into more complex data types like data frames, matrices, multidimensional arrays, lists.
- **Factors**
- Factors are another class of vectors that are treated differently. Factors are essentially categories (e.g., male and female) and therefore it has "levels" which are the different values of each category (e.g., a vector made up of male and female has 2 levels, no matter how many times the males and females are repeated). We can specify when a vector is a factor rather than a character by using the function factor(). You can then check the number of levels with levels()
- sex <- factor(c("male", "female", "female", NA)); levels(sex)
- NAs don't count as a level.
- We can play around with factors with a handy package called "forcats" -- install.packages("forcats"); library(forcats)
- generally, R organises levels in alphabetical order. If you want to change the order (maybe you want a different order in a graph) you can use fct_relevel()
- fct_relevel(sex, c("male", "female"))
- You can also recode, or change the names of, the levels with fct_recode()
- fct_recode(sex, "M" = "male", "F" = "female")
- If you do want to include NA as a level you can use fct_na_value_to_level()
- fct_na_value_to_level(sex, "missing")
- You can turn vectors from one class to another. To turn a factor vector to a character vector use as.character()
- BUT BEWARE when trying to turn vectors into numeric vectors. Remember, R has a hierarchy (character > numeric). When changing a vector to numeric, you must first change it to character or R will change your numbers.
- f_num <- factor(c(1990, 1988, 1977))

- as.numeric(f_num) --> this won't give you the years you are looking for
- as.numeric(as.character(f_num) --> this will give you the correct years as numeric values.
- Careful when reassining a different value to the same object. This is not recommended since you may or may not change the values of other objects that were created using the original object.

Working with Data:

https://datacarpentry.org/R-ecology-lesson/working-with-data.html

Notes:

- we will be using the package tidyverse instead of base R to play around with data
- upload data using read_csv()
- use select() to choose specific columns to work with.
- select(surveys, plot_id, species_id, hindfoot_length)
- You can also do the inverse by putting a minus sign in front of the name in a column to exclude certain columns.
- select(surveys, -record_id, -year)
- You can also use the number of the columns rather than the names.
- select(surveys, c(3,4,5,10))
- You can also use "logistic binaries"--using a logistic function to determine which columns match your requirement
- select(surveys, where(is.numeric)) --> only keeps columns that are numeric
- select(surveys, where(anyNA)) --> which columns have NAs in them
- select() is for columns (has a c), if you want to choose rows we use filter() (has an r). To filter by row you need to specify which column we are looking at and which valued row(s) we want to choose.
- filter(surveys, year == 1985) --> make sure you use == not just = because we are matching values. == is for matching values, it tells R you are looking for TRUE or FALSE. = is used for assigning values.
    - You can also use > (greater than), >= (greater than of equal to), < (lesser than), <= (lesser than or equal to), or != (not equal to) .
    - You can also use " & " (and) and " | " (or) to
    - If you want to choose rows based on multiple values (or values contained within another vector) you can use %in%
    - filter(surveys, species_id %in% c('RM', 'DO')
    - You can also use "is.anyType" like is.na to select or exclude NAs
    - filter

Questions

-

-
-