HW 12: Dates

Graphical Analysis of Biological Data

By the end of this assignment, you should be able to achieve the following tasks in R:

- use R notebooks and R markdown;
- insert, write, and evaluate code chucks;
- make dates from different input formats,
- use the dates for calculations, and
- extract date components for graphing,
- use pipes;
- produce plots with `ggplot2`;
- use a typical workflow to wrangle and plot data; and
- confidently stage, commit, and push with Git.

These achievements belong to Learning Outcomes 2, 3, 4, 5, 6.

By the end of this assignment, you should demonstrate an ability to

Click on any blue text to visit the external website.

This assignment has two parts.

Note: If you contact me for help or (better yet) open an issue in the public discussion forum, please include the code that is not working and also tell me what you have tried.

Preparation

- Read R4ds Chapter 16: Dates and times sections 1-3. I suggest you run the examples but I am not requiring that you do so.

- Open your `.Rproj` project file in RStudio.

- Right-click and save these three files to your `data` folder.

  - lake_ice.csv
  - soil_co2.txt
  - thebes_discharge.csv

- Create an `hw12` folder inside the same folder as your project file.

- Make a file called `lastname_12.Rmd` and save it in your `hw12` folder. Use this file for both parts of the assignment.

- Add the YAML header as usual.

- Load the `tidyverse`, `here` and `lubridate` libraries. You *may* have to install the `lubridate` package first, but I do not think so. If you have to install it, do not include the installation code in your code chunks. Install from the console or the menu.

- **Important note:** The `lubridate` package includes a deprecated function called `here()` that might cause conflict with the `here` library you use to open files. The conflict yields this error:

```
Error in here("data", "soil_co2.txt") :
  unused arguments ("data", "soil_co2.txt")
```

You can avoid this problem by using `here::here("folder","file")` when you import your data files. This format tells R to use the `here()` function specifically from the `here` library. You will see this error if you experience the conflict.

- Source your `my_functions.R` script from the `scripts` folder. You'll need your `std_err` function again for the standard error of the mean.
- Remember to format your code properly.
- Commit early. Commit often. Push regularly.
- Lubridate cheatsheet

Part 1: Play date

- The following code chunk includes a list of pioneer women scientists. Add the code chunk to your file. You may copy and paste.
- Change `birth_date` to your actual birth date between the quotes. Use whatever format you normally use.

```
birth_date <- ""

# Elizabeth Garret Anderson was the first female physician
anderson <- "9 June 1836"

# Mary Anning discovered the first complete Plesiosaur fossil,
# and other important Jurassic fossils
anning <-  "1799/05/21"

# Alice Augusta Ball was a chemist who developed the
# first injectable treatment for leprosy.
ball <- "July 24, 1892"

# Sylvia Earle was the first chieft scientist at NOAA
earle <- "August, 30 (1935)"

# Rosalind Franklin used x-ray diffraction to
# discover DNA structure. Watson and Crick claimed credit.
franklin <- "25th of July, 1920"

# Ester Lederberg was the first to discover that viruses
# infect bacteria. Led to discovery of lambda bacteriophage
lederberg <- "December 18, 1922"

# Barbara McClintock discovered transposable elements (jumping genes),
# and developed staining techniques to study chromosomes
mcclintock <- "16061902"

# Let's also remember a dream
mlk_birth <- "1/15/29"
mlk_dream <- "1963 August 28"
```

Answer these question. Insert code chunk with the answer after each question.

1. Convert each date to POSIXct format, using `lubridate` functions, saving each to a unique variable. Print each result.
2. Check yours dates to be sure they are correct. If you spot any errors, correct them now using the proper `lubridate` function(s).
3. What day of the week were you born on? Show the full day name, not the abbreviated name or the day number.
4. What day of the week will your birthday be on *this* year? Use lubridate's `update` function. You must figure out how to do it entirely by code for full points (no hard-coding). In other words, you code should work just as well next year or five years in the future, without editing.
5. What day of the week would your birthday be on in the Year 2525, if Man is still alive? How old would you be? Here you may hard-code `2525`. Show the abbreviated day of the week.
6. How many days was it between the birth of Dr. Elizabeth Garrett Anderson and Alice Augusta Ball?
7. What is the duration between Martin Luther King Jr.'s birth date and his *I Had a Dream* speech.

Enter the eight POSIXct compatible birth dates, including your own, into a vector in the order they are given above. Name each element of the vector with the last name of each person (review Assignment 2, part 2). You can but do not have to enter the date of the *Dream* speech.

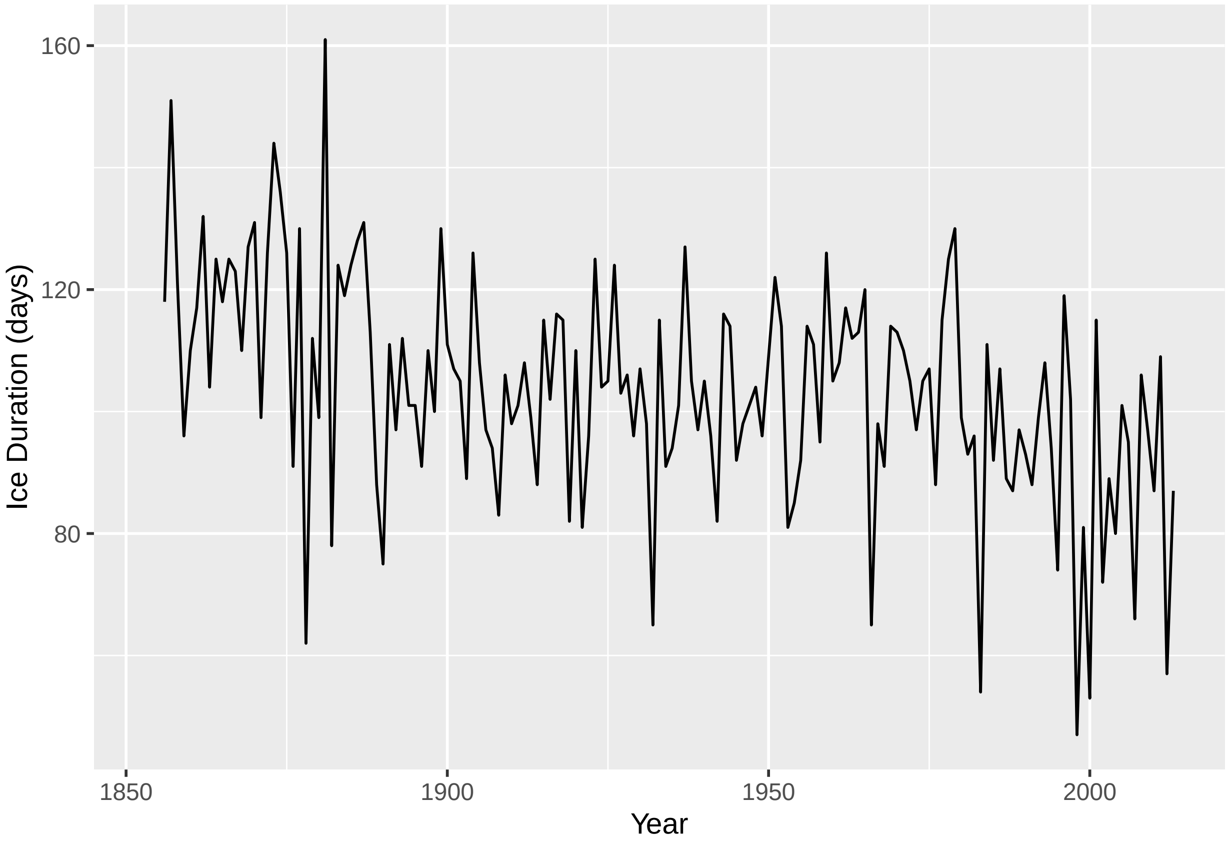
1. Of the eight birth dates from above, including your own, which of them are leap years? You *must*
   - start with the POSIXct dates in your vector, and
   - display the final result as *only* the years of the leap years, not the date. You should end up with 3-4 years, depending on whether your birth year was a leap year. If your results show the full birth dates, then you need to apply one of the `lubridate` functions.
2. Subtract each birth date in the vector from your own to calculate the number of days that have passed between their birth dates and yours. Sort the results from greatest number of days to fewest. You should be listed last (or near the end) with a difference of 0 days.

- Stage, commit, and push.

Part 2: Wrangling and plotting with dates

Lake Ice

- Your goal is to match this plot.

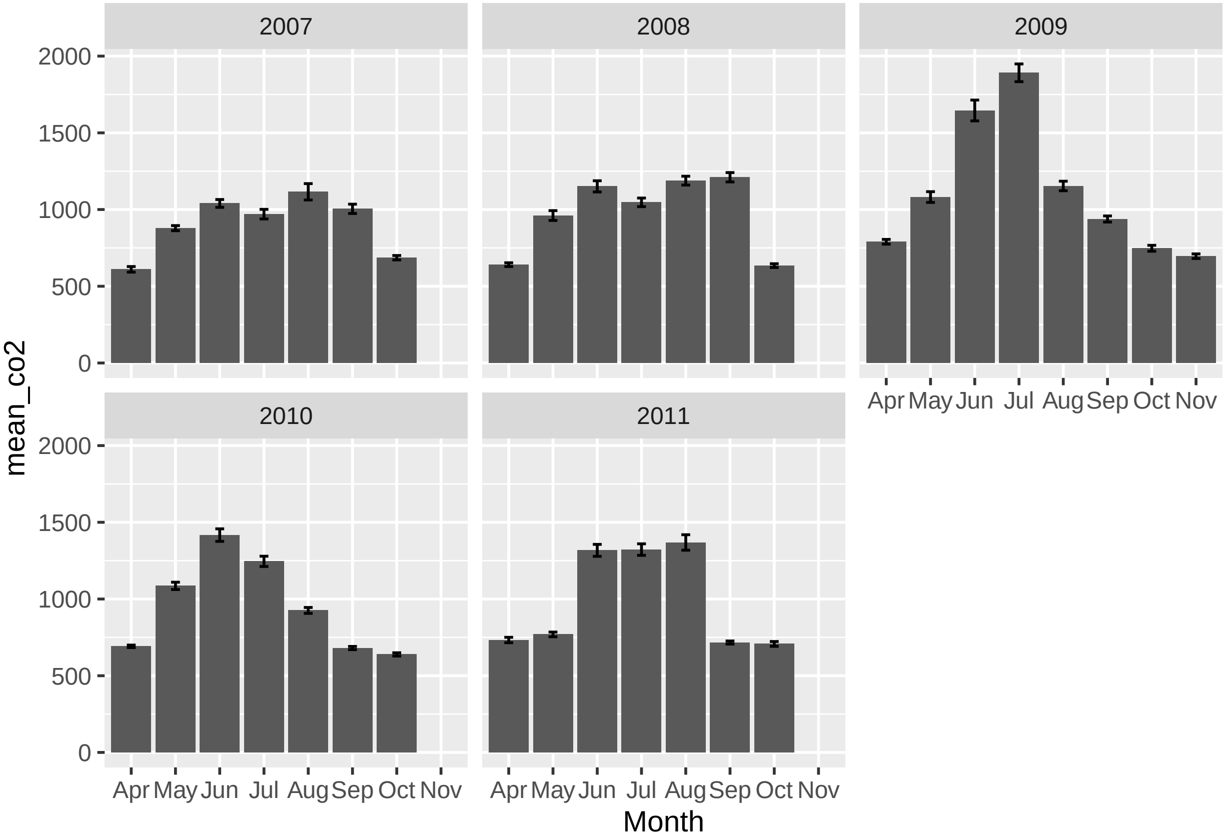


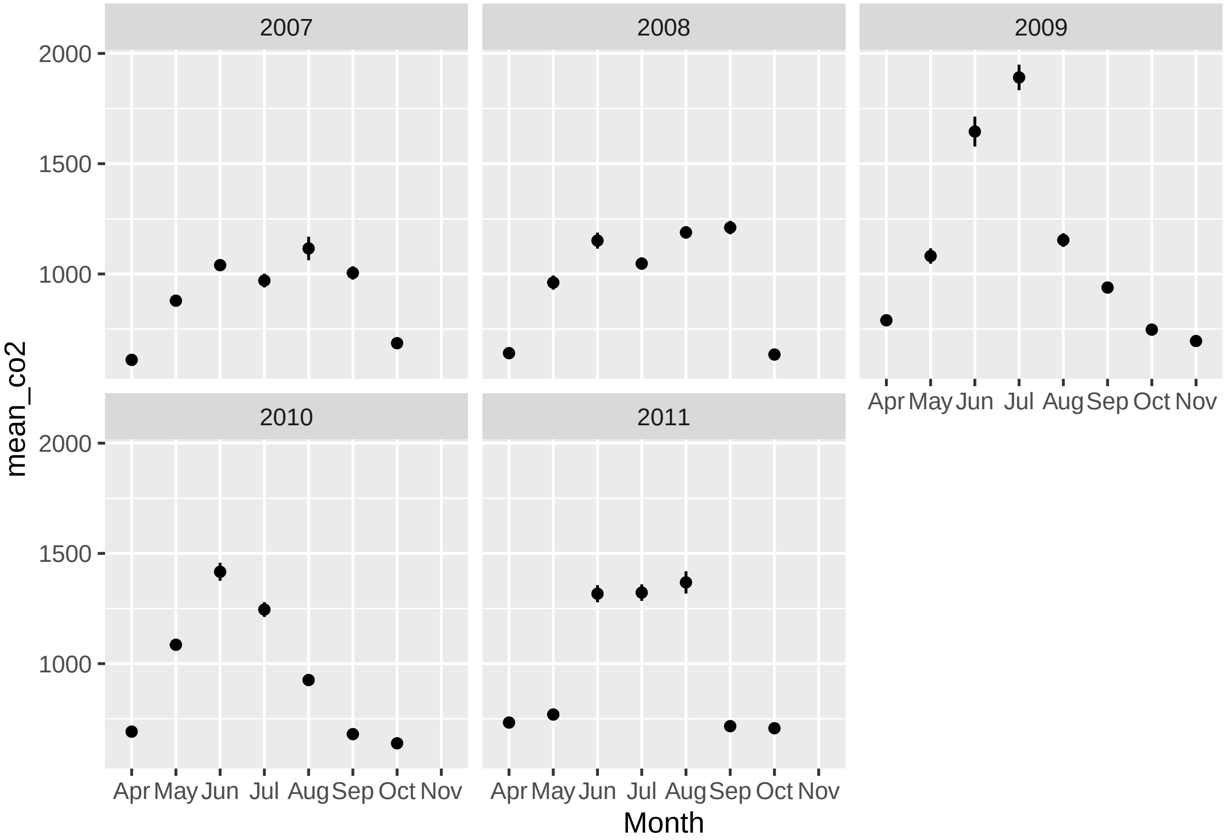
- Import `lake_ice.csv`. The data use "-999" to indicate data that are `NA`. You must handle this *as part of the import*, not part of the wrangling.
- Filter the data to save only Lake Mendota. Use the lake code `DMR1` or the lake name `LAKE MENDOTA`. Case matters!
- Select these six columns. You can do this in one of a few different ways but you should be able to do this very efficiently. Look back at Section 5.4 to review.
  - iceon_year
  - iceon_month
  - iceon_day
  - iceoff_year
  - iceoff_month
  - iceoff_day
- Use `lubridate` functions to assemble the ice on year, month, and date into an `ice_on` date column. Use the same process to assemble the ice off information to make an `ice_off` date column.
- Create a new `duration` column by subtracting `ice_on` from `ice_off`. Make a note of the unit of time for `duration`.

  Important: Subtracting the dates will result in a `difftime` object. Use `as.numeric(ice_off - ice_on)` to convert the result to a numeric object.
- Make a line graph to show how `duration` changed over `iceoff_year`. Use `na.rm = TRUE` where needed to remove the warnings about plotting with `NA` values.
- Make appropriate labels for the X- and Y-axes. Remember to include the unit of measurement for ice duration.
- Stage, commit and push.

Soil Carbon Dioxide

- Import the `soil_co2.txt` dataset.
- Establish a POSIXct-compatible date variable for 01 January 2006.
- Rename the columns so that they are syntactic.
- Make a `Date` column that adds the day of CO2 measurement to the 01 Jan 2006 reference date. Make sure `lubridate` is loaded.
- Create a Year column that extracts just the year from your new `Date` column.
- Create a `Month` column that extracts just the month from your new `Date` column. The month should be the three-letter text form (e.g., "Jan").
- Group your data by Year and Month.
- Summarize mean CO2 and standard error of the mean. You had to calculate standard error for a previous assignment so you can review your previous code.
- Make a column chart of mean CO2 for each month. Use either `geom_errorbar` or `geom_linerange` to plot the standard error of the mean. Use your `std_err` function in the `scripts` folder to calculate the standard error. Use `facet_wrap` on Year to make plots the results by year. If you use `geom_errorbar`, then use the `width` argument to make reasonable horizontal lines.
- Make the same plot but use points instead of columns, and use only `geom_linerange`.
- Your final plots should look like these. To me, the "dot version" is much cleaner.





- Stage, commit, and push.

Mississippi River Discharge

Import and wrangle

The USGS National Water Information Service monitors river depth and volume on rivers across the country. One monitoring gauge is located at Thebes, IL, a few miles southeast of Cape Girardeau.

The `thebes_discharge.csv` dataset contains the average daily discharge from 01 January 1988 to 31 December 2018. During that time, two major flood events occurred in this region, in 1993 and in 2011. Two major droughts also occurred across the regions, in 1988 and in 2012. Discharge is measured in cubic feet/second ($\mathrm{cf} \cdot \mathrm{s}^{-1}$)

This exercise will help you plot data to show overall trends in a larger data set and highlight the flood and drought events.

- Save `thebes_discharge.csv` into your `data` folder.
- Tidy the data. Use `Year` for the key and `discharge` for the value.
- I recommend dividing the discharge by 1000 to improve interpretation of the y-axis.
- Group by Year and Month.
- Summarize the mean discharge. If you end up with `NA` for Feb 2012, then you forgot to do something. Remember it and fix it.
- Add an Event column using `mutate` and `case_when`. If the year equals 1993 or 2011, then the value should be “Flood”. If the year equals 1988 or 2012, then the value should be “Drought”. Otherwise, the value should be “Normal”.

Make a second data set

- Filter your data set to include only the data for the four flood or drought event years. This smaller data set should have the same columns as the larger data set but only 48 rows (4 years, 12 months each). Your code should look something like the following. Try to think of the most efficient way of filtering your data. *Hint:* `%in%`.

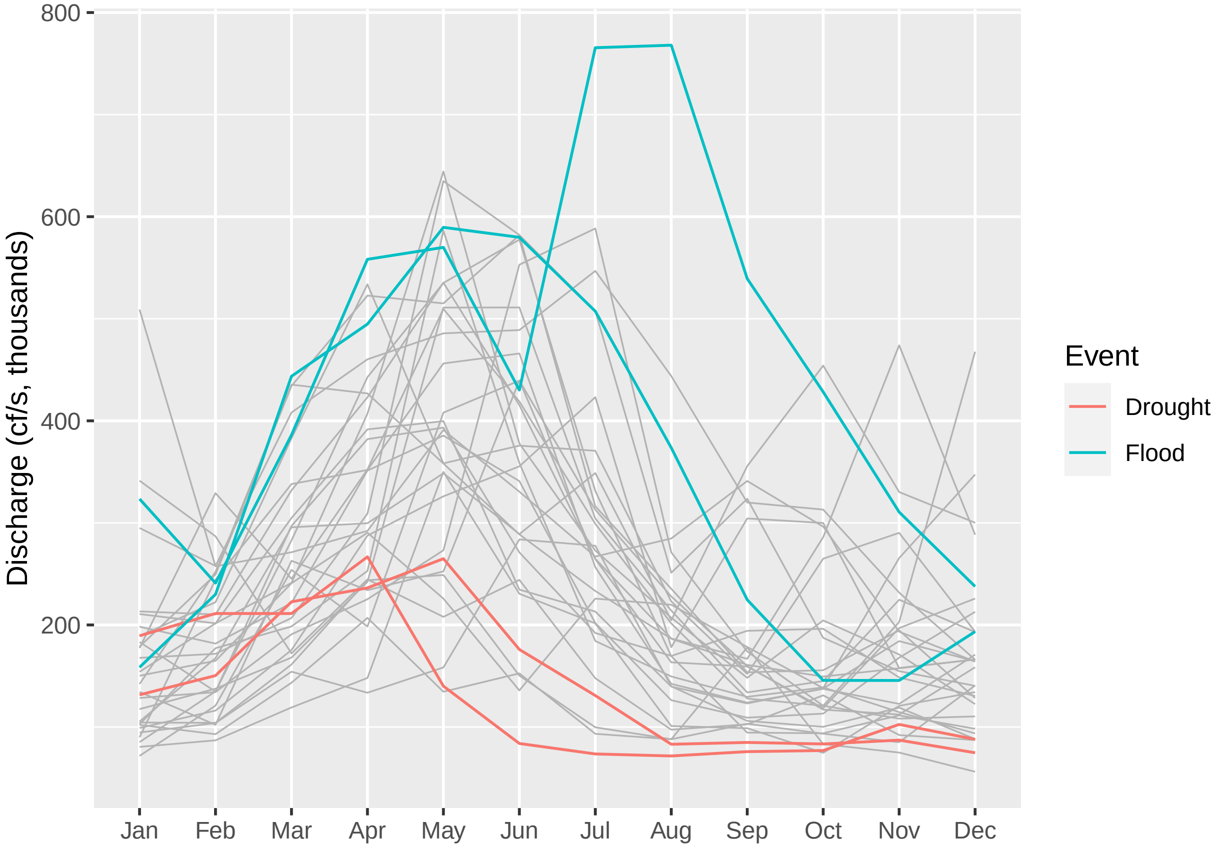
```
events <- big_data_set %>%
  filter(...)
```

Plot the data

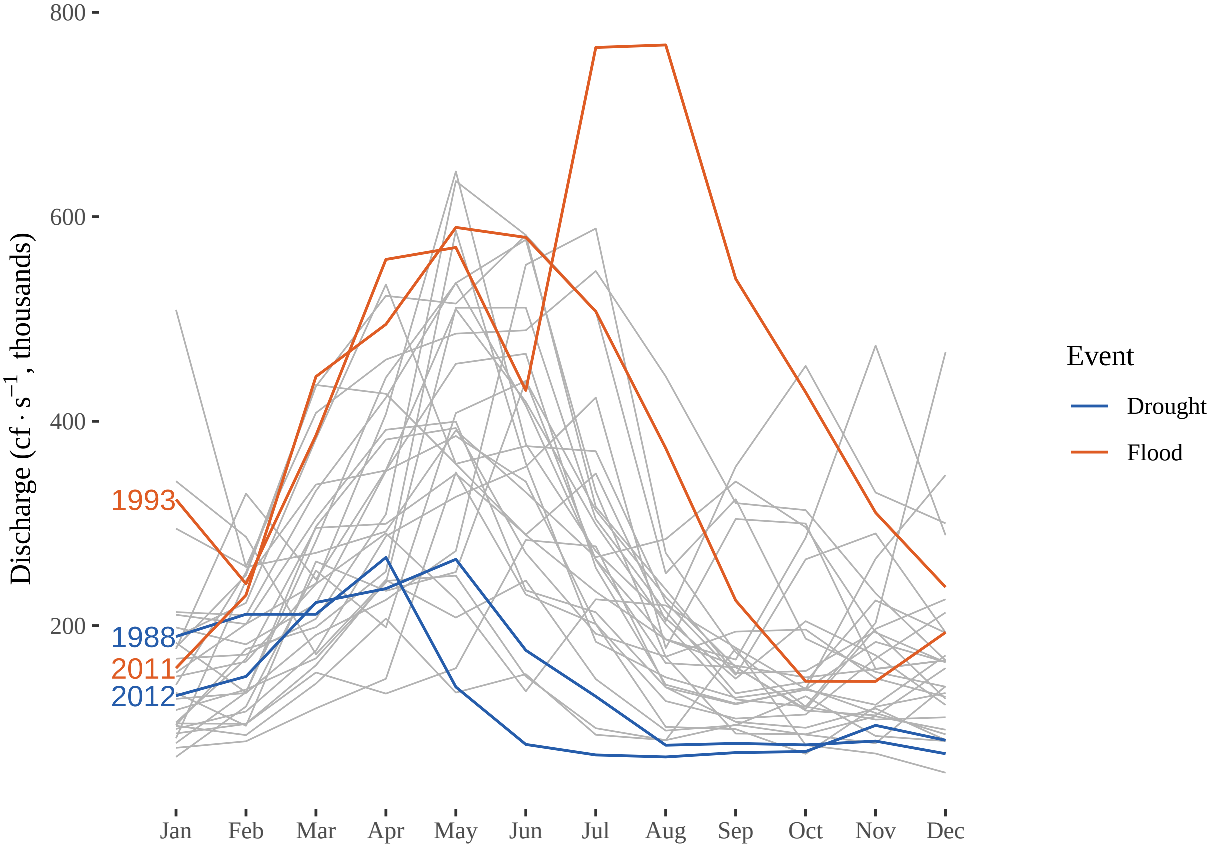
We will build the plot in layers. The first layer will show the overall trend for discharge for all years in the data set. These data are *not* the primary focus so we will use gray, thinner lines to deemphasize them.

- Plot a line graph of the mean discharge by month.
- Use `0.3` for size and `gray70` for color. Adjust the axis labels like that shown here. *Bonus pudding for dessert if you figure out how to make the Y-axis label like that in the final plot below.*
- Once you have this plot looking good like that shown here, save it in an object, e.g., `p1 <- plot code`.
- We will now add the second layer using the smaller data set., by using this code format. Mapping and grouping is the same as above, but add a `color = Event` argument to the `aes()` mapping function. **Important:** If you did not use `Event` as your event column name, then you must substitute the column name that you did use.

Your final plot should look like this.



Here is a bonus plot with further tweaks that coincides more with my tastes in plots and (I think) improves interpretation and the data-ink ratio. I think it still needs a few tweaks, but this is good for now. You will learn more about some of these tweaks in the next assignment.



- Stage, commit, and push.

---

et Vóila