Welcome to PH142: PPDAC and Starting to look at Data

Guess the date

In this year, UC Berkeley established a statistics department (split from mathematics) and hired David Blackwell - the first African American to recieve tenure at UC Berkeley, and the first African American elected to the National Academy of Science (10 years after his appointment)

Quote from Dr. Blackwell

Basically, I'm not interested in doing research and I never have been... I'm interested in understanding, which is quite a different thing. And often to understand something, you have to work it out yourself because no one else has done it.

- quoted from a 2007 New York Times article

Today's Goals

Welcome and orientation to the class - answer questions

My goals for our time together

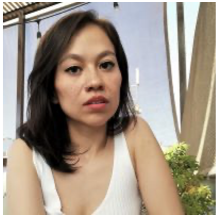
Talk about the framework we use in the class (PPDAC)

Introduce some concepts for working with data in R

Who am I?

Who am I?

Our Teaching team



Our Fabulous Summer 2022 Teaching Team!

https://ph142-ucb.github.io/su22/staff/

Logistics

Lecture/Section/Office Hours/Piazza

Rationale for structure

When in doubt - check the website and the piazza announcements

Logistics

Compressed format . . . .

Frequently asked questions so far

Do I have to attend lecture/section?

Do I need the textbook?

Do I need to know programming?

How to get help with code

- Ask questions during labs/discussion sections, office hours, or on Piazza discussion forum. Use the appropriate thread!
- Develop your online search skills. For example if you have a `ggplot2` question, begin your google search with "r ggplot" and then describe your issues, e.g., "r ggplot how do I make separate lines by a second variable".
- The most common links that will appear are:
  - https://stackoverflow.com: Crowd-sourced answers that have been up-voted. The top answer is often the best one.
  - https://ggplot2.tidyverse.org/: The official ggplot2 webpage is very helpful.
  - https://community.rstudio.com/: The RStudio community page.
  - https://rpubs.com/: Web pages made by R users that often contain helpful tutorials.

Frequently asked questions so far

Figure 2: Will I get an A?

There's an app for that...

Ongoing evolution of the course

From Derivation to hands on programming

Co-Development of course with Dr. Riddell

a pre-emptive appology

(credit to xkcd.com for the comic)

What is this class?

What is this class?

Figure 3: What do you think of when you think about statistics?

My goals for you

Foundational concepts in probability and biostatistics

How to answer questions with data:

- your ability to critically assess statistical information presented to you in scientific and non-scientific fora
- your sense of how to approach answering real world questions with data
- develop your statistical intuition around variability and chance
- develop your toolkit for visualization, summarizing and testing simple relationships
- your ability to concisely and accurately describe statistical methods and results

This is not a math class

Statistics is often classified as a branch of math, but I'd argue that it is more important to focus on the connections that statistics has with science (how we can learn about the world through data)

Though it is true that statistics uses math (and sometimes fairly advanced math!), not much math is needed to learn introductory statistics

In this class we will try, as much as possible, to emphasize concepts and help you develop your statistical intuition

This is not a programming class

Statistics is often viewed as "just computer programming," but this is an incorrect and dangerous characterization: computer programming is simply a tool for conducting statistical analysis

The use of computer programming in statistics is—and should be—quite different than approaches to non-statistical programming

We are using r programming in this course because it is an extremely useful skill, facilitates computation, and is desired in the job market

This is a relevant class

I hope to convince everyone here that statistics is relevant to everyone

As is more and more apparent, public health statistics have relevance to important policy decisions

You also make many decisions during your day that are influenced by statistics

Statistics is not just relevant for public health, but also for other professions, including: education, journalism and law

As we'll try to illustrate via the recurring "statistics is everywhere" segments, statistics is useful for understanding the news and the world around us

Statistics is Everywhere

Advice about how to be happier

How to have a better day during the pandemic

June 30, 2020

(photo courtesy of Pexels)

Passively browsing social media is not good for you — and other useful findings on resilience and happiness from the Positive Emotions and Psychophysiology Lab.

Website link

Advice about how to be happier

In their web posting they say:

We'd like to join those calling for a terminological change. What is needed is not social distancing but physical distancing and social solidarity. During these "challenging times", it's even more important than usual that people stay connected and help each other. So to have a better day during the pandemic, it's vital that everyone MARCH together:

- Minimize passive scrolling through social media.
- Accept negative emotion.
- Really connect with people.
- Care for yourself.
- Help others."

Advice about how to be happier

How did they get to these conclusions?

What should we ask about their methods?

Should we be convinced to change our behavior based on these data?

Advice about how to be happier

Methods:

At the Positive Emotions and Psychophysiology Lab at UNC Chapel Hill, our team recently collected data from over 600 adults around the United States, asking about their experiences and behaviors from the past day.

Problems with scientific reporting

Consequences of poor communication

PPDAC - the approach we will use to answering questions with statistics

Problem

A clear statement of what we are trying to achieve.

Three main problem types

- Descriptive: learning about some particular attribute of a population
- Causative/Etiologic: do changes in an explanatory variable cause changes in a response variable?
- Predictive: how can we best predict the value of the response variable for an individual?

Problem type?

- Insurance company: What is the probability (how likely is it) that a 25 year old unmarried male driver has a car accident?
- Health department: How many cases of influenza have we seen this season compared to last season?
- Health care system: If we treat patients with diabetes using medication X, will thier insulin regulation be better or worse than medication y?

Plan

The procedures we use to carry out the study.

- Census or sample from the target population?
  - How was the sampling conducted?
  - Was the sample random?
- Is the study prospective or retrospective?
- Is the study observational or experimental?

Data

The data which is collected according to the Plan.

- How many observations to we have?
- How reliable are the measures?

Analysis

The data is summarized and analysed to answer the questions posed by the Problem.

We use our knowledge about probabilities to assess the role of chance in our findings.

Conclusion

Conclusions are drawn about what has been learned about answering the Problem.

PPDAC Example 1: A smoking behaviour study

PPDAC Example

Problem: Suppose we wish to study the smoking behavior of California residents aged 14-20 years.

In particular, we are interested in the *prevalence* of current smoking by gender.

What type of problem is this?

PPDAC Example

Plan: We need to first choose a time period, because we know that smoking behavior has changed immensely over time. It is unfeasible to gather these data for all residents in California who are 14-20 years old.

Instead we conduct a *random sample* of size n persons. We collect their: age, gender, and smoking status.

Note that we need to decide how large n should be, and how to obtain the random sample. The latter question is, in particular, very important if we want to ensure that our sample is representative of the population of interest. Time and money also constrain how the sample will be collected.

PPDAC Example

Data: Suppose that a random sample of 200 persons aged 14-20 was selected, yielding these data:

Gender	Number of smokers	Number of non-smokers	Total
Teen girls and women	32	66	98
Teen boys and men	27	75	102
Total	59	141	200

PPDAC Example

Analysis: The proportion of women in the sample who smoke is $32/98 = 33\%$. The proportion of men in the sample who smoke is $27/102 = 26\%$.

We would also like some idea as to how close this estimate is likely to be from the actual proportion in the population.

If we selected a second random sample of the same size, we would likely estimate different proportions for men and women. We will learn how to estimate the precision of these estimates.

PPDAC Example

Conclusion: 33% of girls and women aged 14-20 and 26% of boys and men of the same age group are current smokers in California in 2018 (plus a measure of uncertainty).

Example 2: Life expectancy for non-Hispanic black and white men and women in California between 1969-2013

Introduction

Life expectancy is one of the core measures used in public health to comment on the well-being of groups of people. Differences in life expectancy by race/ethnicity, for individuals living in the same region can reflect underlying inequalities in policies, access to care, food environments, structural and systemic racism, among other potential causes.

Research objective (Problem)

The purpose of this short report is to visualize life expectancy among black and white men and women in California between 1969 and 2013.

We are interested in whether there are differences by group and whether these differences have changed over time.

What type of problem is this?

Plan

Death certificates in the United States include race/ethnicity, age at death, and date of death and capture all deaths of US residents. These data are aggregated by the CDC's National Cancer Institute into the SEER*Stat software. Previously, Riddell et al.[1], analyzed these data to compute estimated trends in life expectancy for non-Hispanic black and white men and women, for 40 US states between 1969 and 2013. States without enough data were excluded from these analyses.

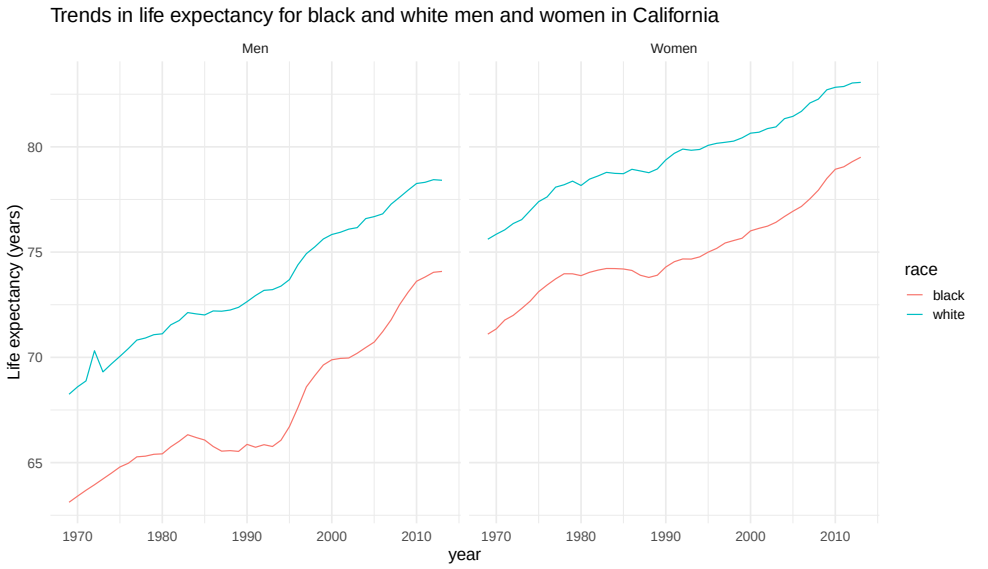
To carry out this short report, we will use data from Riddell et al. to visualize trends in life expectancy as part of an exploratory data analysis. In particular, we will plot time trends for black and white men and women in California.

Data

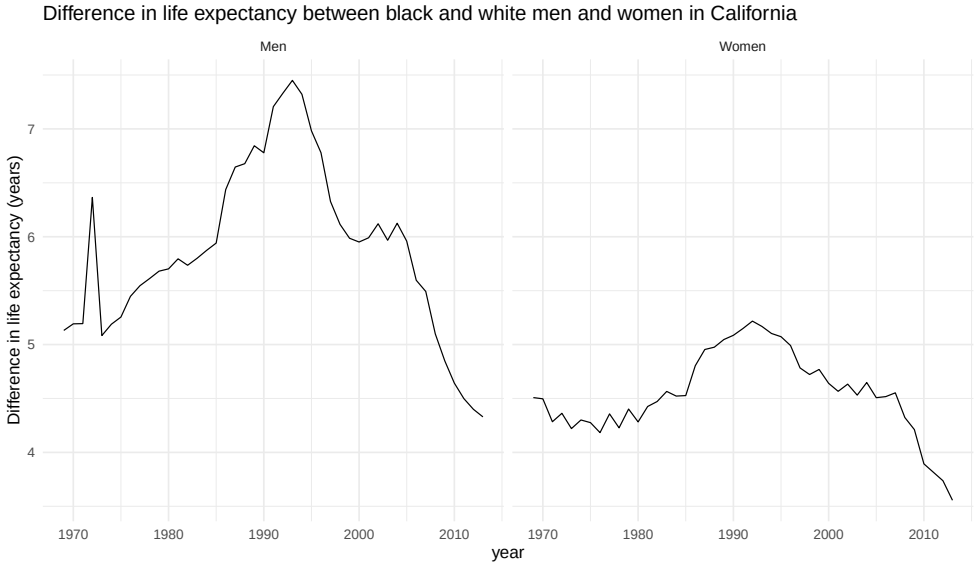
Here are the first few rows of these data for California:

state	stabbrs	year	sex	Census_Region	Census_Division	LE	race
California	CA	1969	Female	West	Pacific	75.61137	white
California	CA	1969	Male	West	Pacific	68.24766	white
California	CA	1970	Female	West	Pacific	75.84916	white
California	CA	1970	Male	West	Pacific	68.59865	white
California	CA	1971	Female	West	Pacific	76.05663	white

Analysis



Analysis



Conclusion

The difference in life expectancy in 1969 between non-Hispanic blacks and whites was 5.1 years for men and 4.5 for women in California.

By 2013, the difference was 4.3 years for men and 3.6 for women in California.

Starting to work with data

R

Your lab today will be about getting oriented in the R and R studio environment.

Here we will talk about some tasks you will do in R to start exploring a dataset

What do you have?

1. What is a data frame

- Identifying the unit of analysis
- Differentiating between the types of variables

2. Get the data into R
3. Figure out what's in the dataset
4. Start to manipulate the dataset

What is a data frame?

- A data frame is a data set.
- We read data into R from common sources like Excel spreadsheets (.xls or .xlsx), text files (.txt), comma separate value files (.csv), and other formats.
- The simplest format of data contains one row for each individual in the study.
- The first column of the data identifies the individual (perhaps by a name or an ID variable).
- Subsequent columns are variables that have been recorded or measured.

Describing your data: what are you working with?

Unit of analysis

The unit of analysis is the major entity you are working with:

- Bacteria
- Laboratory test results
- Individual People
- Groups of people (couples, households)
- Villages
- Countries

Type of Variable

- Categorical variable: A variable that has grouping levels. Mathematically you can calculate the proportion (%) of individuals in each level of the category.
  - Nominal variables: have no underlying order or rank. E.g., hospital ID, HIV status (yes/no variables), race
  - Ordinal variables: can be ordered or ranked. E.g., socio-economic status, BMI categories
- Quantitative variable: A continuous, numeric variable that you can perform mathematical operations on. Mathematically, we can you take the median or average of these variables
  - Discrete variables: can be counted. E.g., number of brain lesions, number of previous births
  - Continuous variables: can be measured precisely, with a ruler or scale. E.g, annual income, blood alcohol content, gestational age at birth

Lake data from Baldi and Moore (B&M)

- Exercise 1.25 from Edition 4 of B&M
- Data from a study of mercury concentration across 53 lakes
- I've placed these data in my working directory

readr is a package in R

- you will learn more abut packages in lab today
- once the package has been "installed" you will need to call it into active use,
- To access readr's functions we load the library like this:

```
library(readr)
```

`read_csv()` to load the lake data in R

- `read_csv()` is a function from the `readr` library used to import csv files.
- code template: `your_data <- read_csv("pathway_to_data.csv")`
- The `<-` is called the assignment operator. It says to save the imported data into an object called `your_data`.

```
lake_data <- read_csv("mercury-lake.csv")
```

Four R functions to get to know a dataset

- `head(your_data)`: Shows the first six rows of the supplied dataset
- `dim(your_data)`: Provides the number of rows by the number of columns
- `names(your_data)`: Lists the variable names of the columns in the dataset
- `str(your_data)`: Summarizes the above information and more

```
# notice that if I put a # in front of a line of code it will not run
#head(lake_data)
#dim(lake_data)
#names(lake_data)
str(lake_data)
```

```
## spec_tbl_df [9 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ lakes      : chr [1:9] "Alligator" "Annie" "Apopka" "Blue Cypress" ...
##  $ ph         : num [1:9] 6.1 5.1 9.1 6.9 4.6 7.3 5.5 7.3 8.2
##  $ chlorophyll: num [1:9] 0.7 3.2 128.3 3.5 1.8 ...
##  $ mercury    : num [1:9] 1.23 1.33 0.04 0.44 1.2 0.27 0.33 0.17 1.87
##  $ number_fish: num [1:9] 5 7 6 12 12 14 5 8 3
```

Some basic functions to begin working with your data

Function 1: rename()

What do you think rename does?

First print the names of the variables:

```
names(lake_data)
```

```
## [1] "lakes"       "ph"          "chlorophyll" "mercury"     "number_fish"
## [6] "age_data"
```

Run the rename() function and assign it to `lake_data_tidy`:

```
lake_data_tidy <- rename(lake_data, name_of_lake = lakes)
```

Function 1: rename()

Then reprint the variable names:

```
names(lake_data_tidy)
```

```
## [1] "name_of_lake" "ph"           "chlorophyll"  "mercury"      "number_fi
## [6] "age_data"
```

Function 1: rename() multiple variables at once

You can rename multiple variables at once:

```
lake_data_tidy <- rename(lake_data,
                         name_of_lake = lakes,
                         ph_level = ph)
```

Code template for rename() function

```
new_dataset <- rename(old_dataset, new_name = old_name)
```

Another way to write the above code is to use the pipe operator: `%>%`

```
new_dataset <- old_dataset %>% rename(new_name = old_name)
```

The pipe will become very useful in a few slides...

Function 2: select()

Based on the output below, what do you think `select()` does?

```
smaller_data <- select(lake_data, lakes, ph, chlorophyll)
names(smaller_data)
```

```
## [1] "lakes"       "ph"          "chlorophyll"
```

Function 2: select()

- We use `select()` to select a subset of variables.
- This is very handy if we inherit a large dataset with several variables that we do not need.

Function 2: "negative select()"

We can also use "negative `select()`" to deselect variables. Suppose we wanted to keep all variables except for `age_data`:

```
smaller_data_2 <- select(lake_data, - age_data)
names(smaller_data_2)
```

```
## [1] "lakes"       "ph"          "chlorophyll" "mercury"     "number_fish"
```

We place a negative sign in front of `age_data` to remove it from the dataset.

Rewrite using the pipe operator

```
smaller_data <- lake_data %>% select(lakes, ph, chlorophyll)
smaller_data_2 <- lake_data %>% select(- age_data)
```

- Going forward, we will use the pipe operator to write code using any `dplyr` functions
- This is because we can use the pipe to stack many `dplyr` functions in a row

Function 3: arrange()

What does arrange do? First type `View(lake_data)` to look at the original data. Then run the code and examine its output below. What is different?:

```
View(lake_data)
lake_data %>% arrange(ph)
```

```
## # A tibble: 9 x 6
##   lakes            ph chlorophyll mercury number_fish age_data
##   <chr>         <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Brick           4.6         1.8    1.2            12 year old
## 2 Annie           5.1         3.2    1.33            7 recent
## 3 Catalina        5.5        13.2    0.33            5 recent
## 4 Alligator       6.1         0.7    1.23            5 year old
## 5 Blue Cypress    6.9         3.5    0.44           12 recent
## 6 Bryant          7.3        44.1    0.27           14 year old
## 7 Four Mile       7.3         0.4    0.17            8 recent
## 8 Henry           8.2        12.2    1.87            3 year old
```

Function 3: arrange() in descending order

```
lake_data %>% arrange(- ph)
```

```
## # A tibble: 9 x 6
##   lakes           ph chlorophyll mercury number_fish age_data
##   <chr>        <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Apopka         9.1       128.     0.04           6 recent
## 2 Henry          8.2        12.2    1.87           3 year old
## 3 Bryant         7.3        44.1    0.27          14 year old
## 4 Four Mile      7.3         0.4    0.17           8 recent
## 5 Blue Cypress   6.9         3.5    0.44          12 recent
## 6 Alligator      6.1         0.7    1.23           5 year old
## 7 Catalina       5.5        13.2    0.33           5 recent
## 8 Annie          5.1         3.2    1.33           7 recent
## 9 Brick          4.6         1.8    1.2           12 year old
```

Function 3: arrange() by two variables

```
lake_data %>% arrange(age_data, -ph)
```

```
## # A tibble: 9 x 6
##   lakes           ph chlorophyll mercury number_fish age_data
##   <chr>        <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Apopka         9.1       128.     0.04           6 recent
## 2 Four Mile      7.3         0.4    0.17           8 recent
## 3 Blue Cypress   6.9         3.5    0.44          12 recent
## 4 Catalina       5.5        13.2    0.33           5 recent
## 5 Annie          5.1         3.2    1.33           7 recent
## 6 Henry          8.2        12.2    1.87           3 year old
## 7 Bryant         7.3        44.1    0.27          14 year old
## 8 Alligator      6.1         0.7    1.23           5 year old
## 9 Brick          4.6         1.8    1.2           12 year old
```

Function 4: mutate()

- `mutate()` is one of the most useful functions!
- It is used to add new variables to the dataset. Suppose that someone told you that the number of fish sampled was actually in hundreds, such that 5 is actually 500. You can use mutate to add a new variable to your dataset that is in the hundreds:

```
lake_data_new_fish <- lake_data %>%
  mutate(actual_fish_sampled = number_fish * 100)

lake_data_new_fish
```

```
## # A tibble: 9 x 7
##   lakes           ph chlorophyll mercury number_fish age_data actual_fish_
##   <chr>        <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Alligator      6.1         0.7    1.23           5 year old
## 2 Annie          5.1         3.2    1.33           7 recent
## 3 Apopka         9.1       128.     0.04           6 recent
```

Use %>% to append several lines of code together

- We have saved many of new datasets in our environment!
- If these datasets were larger, they would take up a lot of space.
- Rather than saving a new dataset each time, we can make successive changes to one dataset like this:

```
tidy_lake_data <- lake_data %>%
  rename(name_of_lake = lakes) %>%
  mutate(actual_fish_sampled = number_fish * 100) %>%
  select(- age_data, - number_fish)
tidy_lake_data
```

```
## # A tibble: 9 x 5
##   name_of_lake    ph chlorophyll mercury actual_fish_sampled
##   <chr>        <dbl>       <dbl>   <dbl>                <dbl>
## 1 Alligator      6.1         0.7    1.23                  500
## 2 Annie          5.1         3.2    1.33                  700
## 3 Apopka         9.1       128.     0.04                  600
```

Use %>% to "pipe" several lines of code together

```
tidy_lake_data <- lake_data %>%
  rename(lake_name = lakes) %>%
  mutate(actual_fish_sampled = number_fish * 100) %>%
  select(- age_data, - number_fish)

#tidy_lake_data
```

Function 5: `filter()`

Filter is another very useful function! What might `filter()` do?

Function 5: `filter()`ing on numeric variables

We use filter to select which rows we want to keep in the dataset. Suppose you were only interested in lakes with `ph` levels of 7 or higher.

```
lake_data_filtered <- lake_data %>% filter(ph > 7)
lake_data_filtered
```

```
## # A tibble: 4 x 6
##   lakes        ph chlorophyll mercury number_fish age_data
##   <chr>     <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Apopka      9.1       128.     0.04           6 recent
## 2 Bryant      7.3        44.1    0.27          14 year old
## 3 Four Mile   7.3         0.4    0.17           8 recent
## 4 Henry       8.2        12.2    1.87           3 year old
```

Function 5: `filter()`ing on character/string variables

Let's try a few more ways to `filter()` the data set since subsetting data is so important:

```
lake_data %>% filter(age_data == "recent")
```

```
## # A tibble: 5 x 6
##   lakes           ph chlorophyll mercury number_fish age_data
##   <chr>        <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Annie          5.1         3.2    1.33           7 recent
## 2 Apopka         9.1       128.     0.04           6 recent
## 3 Blue Cypress   6.9         3.5    0.44          12 recent
## 4 Catalina       5.5        13.2    0.33           5 recent
## 5 Four Mile      7.3         0.4    0.17           8 recent
```

- == is read as "is equal to"

Function 5: `filter()`ing on character/string variables

```
lake_data %>% filter(age_data != "recent")
```

```
## # A tibble: 4 x 6
##   lakes         ph chlorophyll mercury number_fish age_data
##   <chr>      <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Alligator   6.1         0.7    1.23           5 year old
## 2 Brick       4.6         1.8    1.2           12 year old
## 3 Bryant      7.3        44.1    0.27          14 year old
## 4 Henry       8.2        12.2    1.87           3 year old
```

- `!=` is read as "is not equal to"

Function 5: `filter()`ing on character/string variables

```
lake_data %>% filter(lakes %in% c("Alligator", "Blue Cypress"))
```

```
## # A tibble: 2 x 6
##   lakes           ph chlorophyll mercury number_fish age_data
##   <chr>        <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Alligator      6.1         0.7    1.23           5 year old
## 2 Blue Cypress   6.9         3.5    0.44          12 recent
```

- `%in%` is the "in" operator. We are selecting rows where the variable `lakes` belongs to the specified list.
- The `c()` combines "Alligator" and "Blue Cypress" into a list

Function 5: multiple `filter()`s at once

```
lake_data %>% filter(ph > 6, chlorophyll > 30)
```

```
## # A tibble: 2 x 6
##   lakes     ph chlorophyll mercury number_fish age_data
##   <chr>  <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Apopka   9.1       128.     0.04           6 recent
## 2 Bryant   7.3        44.1    0.27          14 year old
```

```
#this is the same as:
lake_data %>% filter(ph > 6 & chlorophyll > 30)
```

```
## # A tibble: 2 x 6
##   lakes     ph chlorophyll mercury number_fish age_data
##   <chr>  <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Apopka   9.1       128.     0.04           6 recent
## 2 Bryant   7.3        44.1    0.27          14 year old
```

Function 5: `filter()` using "or"

```
lake_data %>% filter(ph > 6 | chlorophyll > 30)
```

```
## # A tibble: 6 x 6
##   lakes           ph chlorophyll mercury number_fish age_data
##   <chr>        <dbl>       <dbl>   <dbl>       <dbl> <chr>
## 1 Alligator      6.1         0.7    1.23           5 year old
## 2 Apopka         9.1       128.     0.04           6 recent
## 3 Blue Cypress   6.9         3.5    0.44          12 recent
## 4 Bryant         7.3        44.1    0.27          14 year old
## 5 Four Mile      7.3         0.4    0.17           8 recent
## 6 Henry          8.2        12.2    1.87           3 year old
```

- `|` is the OR operator. At least one of `ph > 6` or `chlorophyll > 30` needs to be true.

Functions 6 and 7: group_by() and summarize()

Let's execute the following code and see what it does.

```
lake_data %>%
  group_by(age_data) %>%
  summarize(mean_ph = mean(ph))
```

```
## # A tibble: 2 x 2
##   age_data mean_ph
##   <chr>       <dbl>
## 1 recent       6.78
## 2 year old     6.55
```

What happened?

Functions 6 and 7: group_by() and summarize()

Another one:

```
lake_data %>%
  group_by(age_data) %>%
  summarize(mean_ph = mean(ph),
            standard_deviation_ph = sd(ph))
```

```
## # A tibble: 2 x 3
##   age_data mean_ph standard_deviation_ph
##   <chr>      <dbl>                 <dbl>
## 1 recent      6.78                  1.59
## 2 year old    6.55                  1.56
```

Recap: What functions did we use?

1. `library()` to load `readr` and `dplyr`.
2. `read_csv()` to read csv files from a directory.
3. `head()`, `str()`, `dim()`, and `names()` to look at our imported data.
4. `rename()` to rename variables in a data frame.
5. `select()` to select a subset of variables.
6. `arrange()` to sort a dataset according to one or more variables.
7. `mutate()` to create new variables.
8. `filter()` to select a subset of rows.
9. `group_by()` and `summarize()` to group the data by a categorial variable and calculate a statistic.
10. `mean()` and `sd()` to calculate the mean and standard deviation of variables.

Recap: What operators did we use?

1. Assignment arrow: `<-`: This is our most important operator!
2. Greater than: `>` There are also:
   - Less than: `<`
   - Greater than or equal to: `>=`, and,
   - Less than or equal to: `<=`
3. Is equal to: `==`, and `!=` is not equal to
4. `%in%` to select from a list, where the list is created using `c()`, i.e., `lakes %in% c("Alligator", "Annie")`

Reference material: Additional material

- Data wrangling cheat sheet

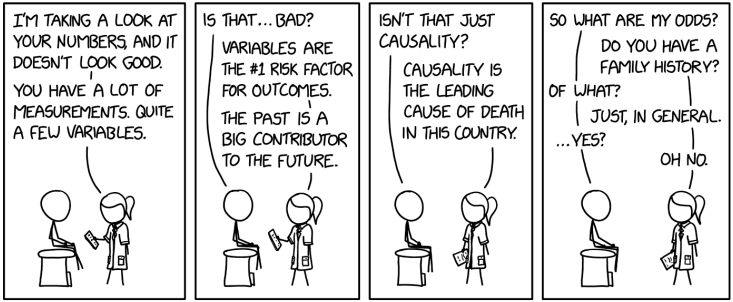
How to export from datahub and save onto your own computer

Some of you may want to edit this file in R markdown by adding notes, etc. In that case, you can make your edits on datahub and save your updated file on the cloud. You can additionally save your updated file locally on your computer. Here's how to do that:

1. In the File view window, click the checkbox beside the file you'd like to export
2. click More $>$ Export.

This will download the file to your computer's downloads folder.

Parting humor



courtesy of xkcd.com