Name: Gabby Burnette

Date: 2-11-21

GOAL: LEARN HOW TO CREATE GOOD SPREADSHEETS!

Purpose:

1. **What is a spreadsheet and how do they work**
2. **What separates a good dataset from a bad one**
3. **How can we apply best practices to create a spreadsheet that is useful to other**

What is clean data? Clean data are in a format that is ready to analyze. Characteristics of clean data include data that are:

Common symptoms of messy data include data that contain:

- Special characters (e.g. commas in numeric values)
- Numeric values stored as text/character data types
- Duplicate rows
- White space
- Missing data
- Zeros instead of null values
- “highlights” and colors used to denote information
- More than one type of info stored in a cell

Part A: For each spreadsheet identify at least one thing that makes it problematic. Then describe how you might go about fixing it (i.e., what could you do to make it ‘clean.’

For help see:
https://datacarpentry.org/spreadsheets-socialsci/02-common-mistakes/index.html
or watch the online video I filmed and put on AsULearn for Week 4

1. **Sheet 1**

Sample	**Date excavated**	**Date analyzed**	**Element**
1	**5/9/2019**	**1/5/20**	**Radius**
2	**5/1/2019**	**1/5/20**	**Radius**
3	**5/3/2019**	**1/6/20**	**Ulna**
4	**W**	**1/6**	**femur**

Problem:

Missing Information, problematic null value (why W?)

Better version:

Sample	**Date excavated**	**Date analyzed**	**Element**
1	**5/9/2019**	**1/5/20**	**Radius**
2	**5/1/2019**	**1/5/20**	**Radius**
3	**5/3/2019**	**1/6/20**	**Ulna**
4	**N/A**	**1/6/20**	**Femur**

2. **Sheet 2**

Good spreadsheets

	Pit A		Pit B		
Munsell	weight	Date	Munsell	weight	Date
10 yr 4/3	1.5	2020-02-03	10 yr 2/3	7.5	2020-02-05
10 yr 4/3	1.7		10 yr 2/3	3	
10 yr 4/3	2.0		10 yr 2/3		
10 yr 4/3	2.1	2020-02-04	10 yr 2/3	3.3	
10 yr 4/4	2		10 yr 2/3	9	

Problem:

Two Charts one Excel

Better version:

	Pit A				
Munsell	weight	Date			
10 yr 4/3	1.5	2020-02-03			
10 yr 4/3	1.7				
10 yr 4/3	2.0				
10 yr 4/3	2.1	2020-02-04			
10 yr 4/4	2				

Pit B		
Munsell	weight	Date

Good spreadsheets

10 yr 2/3	7.5	2020-02-05
10 yr 2/3	3	
10 yr 2/3		
10 yr 2/3	3.3	
10 yr 2/3	9	

3. **Sheet 3**

		Site	elements	Biological sex	Analysist	
	Skeleton UW-503					
Transect A		Oblion	Pelvis	Male	GP	
			femur	male	GP	
	Skeleton UW-504					
Transect B		PineTree	Cranium	Female	SL	
			Pelvis	Female	SL	
			Tibia	NA	SL	

Problem:

Good spreadsheets

Same Analyst

Better version:

		Site	Elements	Biological sex	Analyst	
	Skeleton UW-503					
Transect A		Oblion	Pelvis	Male	GP	
			Femur	Male		
	Skeleton UW-504					
Transect B		PineTree	Cranium	Female	SL	
			Pelvis	Female		
			Tibia	NA		

4. **Sheet 4**

		Week 1		Week 2		Week 3		
ID	SEX	date	weight	date	weight	Date	weight	
002	M							
005	F							
006	F							
010	M							

012	M							
015	M							

Problem:

Capitalization

Better version:

5. **Sheet 5**

Sample	date	Glucose level
101	2019-01-14	150.5
102	2019-01-14	93.6
103	2019-01-14	99.5
104	2019-01-15	108.0
105	2019-01-15	7.9
106	2019-01-15	102.7

Problem:

No highlighting to point out features

Better version:

6.

Sample	date	Glucose level
101	2019-01-14	150.5
102	2019-01-14	93.6
103	2019-01-14	99.5
104	2019-01-15	108.0
105	2019-01-15	7.9
106	2019-01-15	102.7

Part B: fixing/data cleaning examples

Sadly, most folks don't follow the above suggestions. Here is an example, modified from a colleague's assignment, that lets you see how to work with messy data

You can download the data for this part on this google drive folder: (https://drive.google.com/drive/u/1/folders/1zQtdbwQuVrxtHrFG7lXdW02ZV0r5cKnu)

The file are from New York City and give info on real estate sales for different boroughs (administrative divisions that make up New York City) …Let's say you wanted to help a friend who was trying to learn more about this and they gave you these files and asked you what you can learn. Put the data into your current project folder so that you can access it easily (if you don't have Excel on your machine don't worry about that. You should be able to open the file just to view it on the Drive)

Setup

To do this part of the lab make sure you run the following code as we will need these R libraries.

```
library(tidyverse)
install.packages("janitor") #this helps to clean our data
library(janitor)
library(readxl) # this package comes when you install the tidyverse
#package but is only loaded individually
```

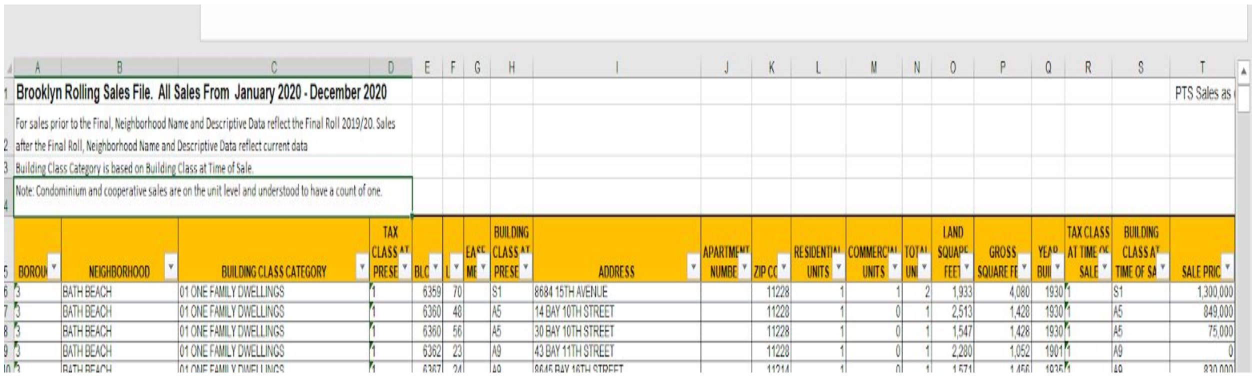
Step 1: take a look at the raw data

- Take a look at the Excel sheet either in the Google folder or after downloading and make some notes about what problems you see that you may want to edit/fix as you go

Step 2: get the data into R

In the folder you will see multiple versions of essentially the same file (one is an excel sheet and the other is in a CSV (with some of the text edited out…not ideal but c'est la vie).

- If you have Excel on your computer open it and take a look…you will see something like I copied below. Note that the first few lines are info that we don't need in our data table,,,



1 Brooklyn Rolling Sales File. All Sales From January 2020 - December 2020 — PTS Sales as

2 For sales prior to the Final, Neighborhood Name and Descriptive Data reflect the Final Roll 2019/20. Sales after the Final Roll, Neighborhood Name and Descriptive Data reflect current data

3 Building Class Category is based on Building Class at Time of Sale.

4 Note: Condominium and cooperative sales are on the unit level and understood to have a count of one.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
5	BOROU	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESE	BLC	L	EASE ME	BUILDING CLASS AT PRESE	ADDRESS	APARTMENT NUMBE	ZIP CC	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNI	LAND SQUARE FEET	GROSS SQUARE FE	YEAR BUI	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SA	SALE PRIC
6	3	BATH BEACH	01 ONE FAMILY DWELLINGS	1	6359	70		S1	8684 15TH AVENUE		11228	1	1	2	1,933	4,080	1930	1	S1	1,300,000
7	3	BATH BEACH	01 ONE FAMILY DWELLINGS	1	6360	48		A5	14 BAY 10TH STREET		11228	1	0	1	2,513	1,428	1930	1	A5	849,000
8	3	BATH BEACH	01 ONE FAMILY DWELLINGS	1	6360	56		A5	30 BAY 10TH STREET		11228	1	0	1	1,547	1,428	1930	1	A5	75,000
9	3	BATH BEACH	01 ONE FAMILY DWELLINGS	1	6362	23		A9	43 BAY 11TH STREET		11228	1	0	1	2,280	1,052	1901	1	A9	0
10	3	BATH BEACH	01 ONE FAMILY DWELLINGS	1	6367	24		A9	8645 BAY 16TH STREET		11214	1	0	1	1,571	1,456	1935	1	A9	830,000

- We are going to read the data in with read_xls(), a function in the readxl package
- The function read_xls takes a number of arguments (the stuff that goes in between the parenthesis)
  - path = where on your machine the file is stored
  - sheet = the name of the sheet (some excel files have multiple sheets)
  - na = what to count as missing value (by default it counts blank cells as missing

- skip = Minimum number of rows to skip before reading anything, be it column names or data. Leading empty rows are automatically skipped, so this is a lower bound. Ignored if range is give

- looking at these data it seems that all we need here is to tell it to skip the first 4 line
- on my computer, I have the data in my working directory in a folder called "datasets." You will have to change this to wherever you put the data. If you are having trouble PLEASE reach out. Getting a handle on this is one of the biggest hurdles and I am here to help!

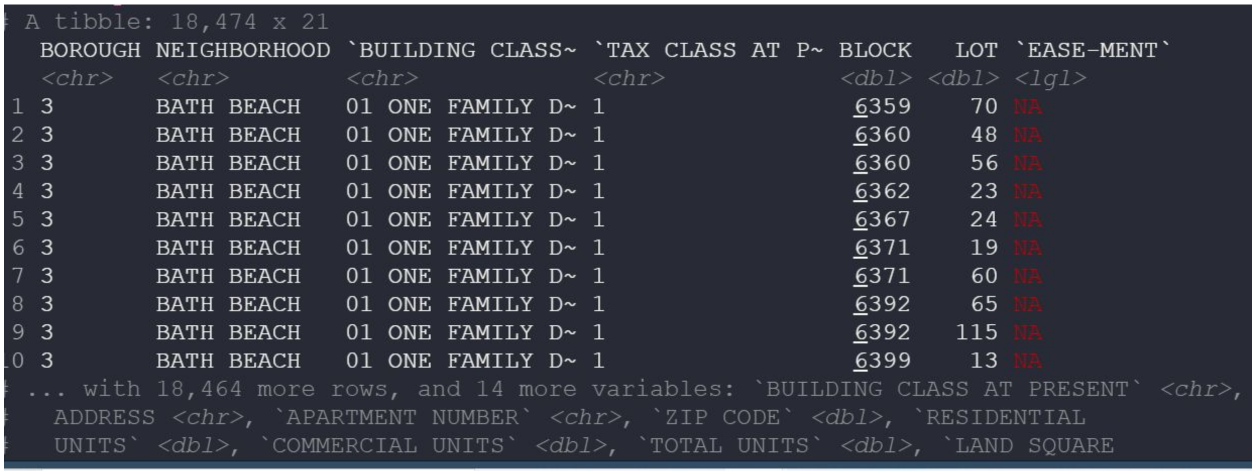
```
brooklyn <- read_xls(path ="datasets/rollingsales_brooklyn.xls", skip = 4)
```

Note: the 'path' is where the data sheet is located on the computer. If you write path = "" and hit the tab key between the quotation marks you should see the files in your current working directory pop up. That will let you choose the file you want

- We now have the dataset stored as an object called "brooklyn." To examine it we can just type the function's name into R

```
brooklyn
```

you will see something like this:



```
# A tibble: 18,474 x 21
  BOROUGH NEIGHBORHOOD `BUILDING CLASS~ `TAX CLASS AT P~ BLOCK   LOT `EASE-MENT`
  <chr>   <chr>        <chr>            <chr>            <dbl> <dbl> <lgl>
1 3       BATH BEACH   01 ONE FAMILY D~ 1                 6359    70 NA
2 3       BATH BEACH   01 ONE FAMILY D~ 1                 6360    48 NA
3 3       BATH BEACH   01 ONE FAMILY D~ 1                 6360    56 NA
4 3       BATH BEACH   01 ONE FAMILY D~ 1                 6362    23 NA
5 3       BATH BEACH   01 ONE FAMILY D~ 1                 6367    24 NA
6 3       BATH BEACH   01 ONE FAMILY D~ 1                 6371    19 NA
7 3       BATH BEACH   01 ONE FAMILY D~ 1                 6371    60 NA
8 3       BATH BEACH   01 ONE FAMILY D~ 1                 6392    65 NA
9 3       BATH BEACH   01 ONE FAMILY D~ 1                 6392   115 NA
10 3      BATH BEACH   01 ONE FAMILY D~ 1                 6399    13 NA
# ... with 18,464 more rows, and 14 more variables: `BUILDING CLASS AT PRESENT` <chr>,
#   ADDRESS <chr>, `APARTMENT NUMBER` <chr>, `ZIP CODE` <dbl>, `RESIDENTIAL
#   UNITS` <dbl>, `COMMERCIAL UNITS` <dbl>, `TOTAL UNITS` <dbl>, `LAND SQUARE
```

- Note that it tells us there are 11,874 rows and 21 columns of data…that is a lot of data!

- It also shows the first few rows and data.

- Now, enter the following into R to create two new objects that have the same data source but use different *functions* for reading in the data (read.csv and read_csv). Note that this time we are using the .csv file rather than the .xls file

```
brooklyn_csv_base <- read.csv("datasets/rollingsales_brooklyn_csv.csv", skip = 4)

brooklyn_csv_readr <- read_csv("datasets/rollingsales_brooklyn_csv.csv", skip = 4)
```

Now compare the three different datasets….what differences do you see in how the data are stored? One way to do this is with the Tidyverse's glimpse() function, which gives a nice summary of the object. You will notice that with glimpse you can see the name of each column and then the class of that column. Pay attention to each object and when the class of a column is different.

```
glimpse(brooklyn_csv_base)

glimpse(brooklyn_csv_readr)

glimpse(brooklyn)
```

jot down some notes about how these objects differ:

Different columns

Step 2: connecting datasets together

Note how each borough is setup the same in the sheets. This is a happy thing for us. Lets say we want to look at all the data at one go!

```
brooklyn <- read_excel("rollingsales_brooklyn.xls", skip = 4)

bronx <- read_excel("rollingsales_bronx.xls", skip = 4)
```

```
manhattan <- read_excel("rollingsales_manhattan.xls", skip = 4)

staten_island <- read_excel("rollingsales_statenisland.xls", skip = 4)

queens <- read_excel("rollingsales_queens.xls", skip = 4)



# Bind all dataframes into one, save as "NYC_property_sales"

NYC_property_sales <- bind_rows(brooklyn, bronx, manhattan, staten_island,
queens)



glimpse(NYC_property_sales)
```

Now we have all the data in one dataset. Note the column "BOROUGH" is a numerical code for the borough so that info is still present. We could rename them if we wanted to…

Step 3: Clean up the data: Let's circle back to Brooklyn dataset from above

- Take a look at the col names :

```
colnames(brooklyn)
```

take a moment and jot down some thoughts about what we might want to fix about the column names

spaces, long

To me, a few things stick out about the names of the columns that may make data analysis tricky

1. All capitalized

2. Spaces in the col names

<u>We can fix this in a number of ways:</u>

- **To get the capitalization fixed:**
  Take a look at the function below. A few things are happening here. The "%>%" is a function from the tidyverse that makes coding a bit easier. It is called a pipe and basically it says "then". So you can read the code as "take the column names from brooklyn then use the str_to_title function. This function, also part of the Tidyverse, take a string (an ordered sequences of characters) and changes the case of the letters. Play around and see what other str_to_ functions exist and what they do

```
colnames(brooklyn) %>% str_to_title()
```

- But this code doesn't change anything. to save this we just assign it to colname(brooklyn)

```
colnames(brooklyn) <- colnames(brooklyn) %>% str_to_title()
```

<u>Now, we want to remove the spaces</u>

Thankfully Tidyverse gives us that with the useful str_replace_all() function, Here, the str_replace_all() takes two arguments. The first one is what we want to replace and the second it what we want to put in its place.

```
colnames(brooklyn) %>% str_replace_all(" ", "_")
```

In this instance, we are replacing a space with a underscore. How would you replace the space with it with a . ?

Remember to save it:

```
colnames(brooklyn) <- colnames(brooklyn) %>% str_replace_all(" ",
"_")
```

fun fact: you could do this all in a single line like this: colnames(Brooklyn) <- colnames(brooklyn) %>% str_to_title() %>% str_replace_all(" ", "_")

<u>Protip:</u>

Now that you did that I'm going to teach you a trick using the janitor package we installed at the beginning. In that package is a function called clean_names() which will change the case and remove spaces in one go!

```
brooklyn %>% clean_names()
```

Part 3: Try it on your dataset

Take a look at the data you are examining for this course. Thinking about what we talked about this week and in this lab is there anything in your dataset that needs to be 'cleaned to that it can work well. If so, what steps might you think about here

Part 4: assessment

1. How can you apply 'clean' spreadsheet techniques to other work you do?
    a. In GIS i often have a hundred titles for different files so this would definitely help.
2. After doing this lab, what sorts of reasons might you give for the importance of good data organization?
    a. It keeps your computer workspace from getting messy.
3. "Reading in" data to R, especially *messy* data, is one of the most frustrating parts of data analysis. Do you think you are getting the hang of it? What could make this easier?
    a. A cheat sheet with all the codes and what they do would definitely help.
4. Next week we are starting to think about data visualization. For your dataset what sorts of figures might you want to make?

Good spreadsheets

a. I can definitely see bar graphs being useful for my specific data.