Concept 1.0 | Variable Types

We cannot typically understand our data without summarizing it. We do this with tables and figures. There are many summary tools. Some more useful than others. Some are useful for some situations but not in others. The main differentiators between a good and bad summarization tools is the shape of the data and what you're trying to understand about the data.

There are generally three dimensions of data: Variable Type, Data Structure, and Number of Variables. We'll return to Data Structures next week. And later we'll talk about multivariate data. To start, we'll try to classify Variable Types

There are two main variable types, each with subtypes: Categorical Variables and Numerical Variables.

Excel Exercise 1.0

Lets start by downloading the data. It's good to stay organized. We're going to work with many datasets this semester. So lets create a folder to hold our files. If you're not familiar with folders, you can simply right click and create one. Lets name it ECON 0150. Then inside we'll create another one called Exercise 1.0. Then we can put `employment_status.csv` into it. This is what we call a 'comma separated value' file, which is just a format for storing data in a file. It's not worth thinking about at the moment except that when we see a file that ends in `.csv` we can know it's a data file. We can open csv files in many applications. If you have Excel installed, we can double click on the file and it should open in Excel. This is not an excel file yet. Excel has it's own format, which is a little confusing at first. No need to worry about it now. We'll get to that later.

Once we have the file open, we can see what the dataset contains. It's organized by rows and columns. Each row represents a household or individual in this case. This is the unit of observation. Each column represents a bit of information about the unit of observation. So in this case, we have a column called `Employment Status`, which tells us whether the household contains someone who is employed.

Here we can see the different kinds of values `Employment Status` can take. There's a way of summarizing this in Excel. If we click on a cell in the spreadsheet we can enter information into the cell. So you could add information like your name into the cell `D5` simply by clicking on the cell at row 5 and column D and entering your name. But here we want to find all the unique values that the column `Employment Status` can take. Excel has these amazing things called functions, which allow us to perform operations on data. Just like a mathematical function, we define it, give it inputs, and then the function will spit something out for us. To use a function, all we have to do is click on a cell, hit equals and then write out the function we're interested in. For our purposes here, we want to use the function `UNIQUE`. You'll notice that it requries us to give it some data. In this case we want to tell it to find the unique values in the B column. We can write out the list of entries in this column using the following command:

```
=UNIQUE(B1:B100)
```

When we hit enter this will spit out all the possible entries in the list. In this case, that's *Employed* or *Unemployed*. Very nice!

Now this tells us that it's a binary categorical variable because we're dealing with categories that are binary.

Python Exercise 1.0

Next, lets do this with python. The steps are fundamentally the same. Except, instead of needing to click a bunch of things each time we want to run the same steps, we have code that can do it for us in exactly the same way every time.

To open the data, instead of clicking on the data and opening it in Excel, we load the data into a notebook. I've already given you a simple place to get started. You can find the Exercise 1.0 notebook on the course page. Simply click on that link and it will open a notebook for you in Google Colab.

When you open it you'll notice a few things. First, this is running in your browser. This is very nice. You can install python on your computer, but for this class, there's really no need. Second, you'll notice that this doesn't look like Excel. And that's a good thing. The notebook has code cells, which are sets of instructions we can have the computer take on our behalf. And the computer has text cells, which is just a nice way to write out some english for us to document what we're doing. If we hit control/command enter, we can get the code to run.

What code allows us to do is perform the same kinds of operations like we did in excel, but have it run the same way every time, going step by step in the way we tell it. If you're like me, you may forget what you did with your data last time. When I work in Excel, I need to write down the steps I take so that if I need to go back, I can remember what I did. That's actually just what code does for us. It's a set of instructions we tell the computer. But instead of needing to follow those instructions ourselves, we just have the computer do them for us. And this allows us to easily communicate to others what we have done. We care about what we call 'Reproducible Science', and code is a big part of making sure others understand what we're doing and can easily copy our steps.

You'll learn through experimentation and study much of how code works. But let me give you a quick primer on some basic ideas.

Just like in math, we can define a variable. Lets define x to be the number 2. We can run this cell, which will tell the computer that x is equal to 2. If we want to see what x is, all we have to do is type x into the cell and run the cell. This will print out the value of x. We can even redefine variables. So for example, lets say I actually wanted x to be 1, I could just swap the 2 for a 1. If I print x now without rerunning the definition cell, then it's still going to tell me that x is 2. I didn't have the computer redefine x as 1 yet. But if I rerun the cell, it will redefine x as 1, which I can see when I ask the computer to print x. Similarly, we could have the computer define x as 1 *after* having it define x as 2. In this case, after I run that cell and print the value of x, I get that x is 1.

We can even do math with code. So lets say I want to add 1 + 1. Then all I have to do is write that out. I can define x to be the solution to 1 + 1 if I wanted. There's so much to explore here.

We can define x to be anything we like. For example, maybe I want x to be defined as my name. Where with numbers we just set x equal to the number, here I define x to be what we call a string, which is just some text. I need to put a ' or a " at the start and end, around my name. Everything inside the parentheses is now treated as text. Just like before, if I were to change my name, I could just redefine x to be my new name. And I could even define a different variable called `name` to be my name. This is better because now the variable name itself tells me a bit about what it contains.

And we can add strings. If I wanted to take my wife's name when we got married, I could just add `name` plus `Kraybill` which would just add the two strings together to get a new name.

The first instruction we typically give is to tell the computer to load some software. For our little example here, lets import numpy, which is python's numbers package. Python on it's own doesn't always have everything you'd want. If it did, it would be a huge peice of software. So instead, all coding languages like python have external packages that we import when we need them. Like numpy.

Numpy has a lot of things. But to show a simple example of how to work with a package, lets run:

```
import numpy as np
```

Just like my name is Taylor but some friends in college called me Little Tay, we can call numpy whatever we like. In this case, it's nice to not have to write out numpy every time we need it. So we just give it the nickname np.

We'll get to data in a second. But for a second, lets say I want to know the cosine of some angle theta. And lets say `theta = 50`. Numpy can get the cosine of an angle with:

```
np.cos(theta)
```

This tells python to look in the numpy library for the cos function and then run it on the angle theta. This is a simple example of a very powerful tool. Not only does this work like a calculator, but it allows us to write down the steps that we're running so that we can reproduce our output.

This is a set of powerful ideas that runs our modern world and runs modern science. And you will find it very useful in your projects not just this semester. There's a very good reason why we're spending time learning to think this way. It's not just for the nerds.

The data package that we typically use in python is what's called `pandas`, which is not a reference to the charasmatic megafauna, but is a portmanteau of panal and data, a type of data structure we'll talk about in a couple of days. We typically nickname it as pd:

```
import pandas as pd
```

I've given you some example usage. We use the string `file_path` to tell pandas where to look for the dataset `employment_status.csv`, then we add those strings together and have pandas load that data, calling the dataset `employment`. This is a variable just like `x`, except now it contains a dataset. We can look at the dataset by printing it in a cell. Or we can just look at the top of the dataset using `employment.head()`.

We see that there are rows and columns just like in excel. And in fact, this data frame in python is exactly the same idea as a dataset in excel. It's just now it's in the python world.

We can get a column in this dataframe by simply asking python for just that column:

```
employment['Employment Status']
```

This selects the column Employment Status from the dataset employment. If we want to find all the unique values like we did in excel, we can simply add `.unique()` to the end, which tells python to get the unique values in that column.

There's a bit more to get started with python, but as you can see, it's already a much more useful way of working with data. And it exactly mirrors what we can do in excel. It's just we can write down the steps so that we can have the computer do it for us and do it the same way every time.

Here we can see that Employment Status is a binary categorical variable. Lets now look at the other four.