Nhu Vo

Lab #1

more here

 # New Section

Some preliminary set up code (don't worry too much about this now):

```
from __future__ import division
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
import os
import matplotlib.pyplot as plt
```

Let's look a a survey of people's favorite candies. Grab the data online ...

```
import pandas as pd
url = 'https://raw.githubusercontent.com/fivethirtyeight/data/master/candy-power-ranking/candy-data.csv'
df = pd.read_csv(url)
```

Look at the data

```
df.head()
```

	competitorname	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	hard	bar	pluribus	sugarp
0	100 Grand	1	0	1	0	0	1	0	1	0	
1	3 Musketeers	1	0	0	0	1	0	0	1	0	
2	One dime	0	0	0	0	0	0	0	0	0	
3	One quarter	0	0	0	0	0	0	0	0	0	
4	Air Heads	0	1	0	0	0	0	0	0	0	

Let's look at the distribution of a variable, chocolate

```
df.chocolate.value_counts().sort_index()
```

```
0    48
1    37
Name: chocolate, dtype: int64
```

In percentage terms:

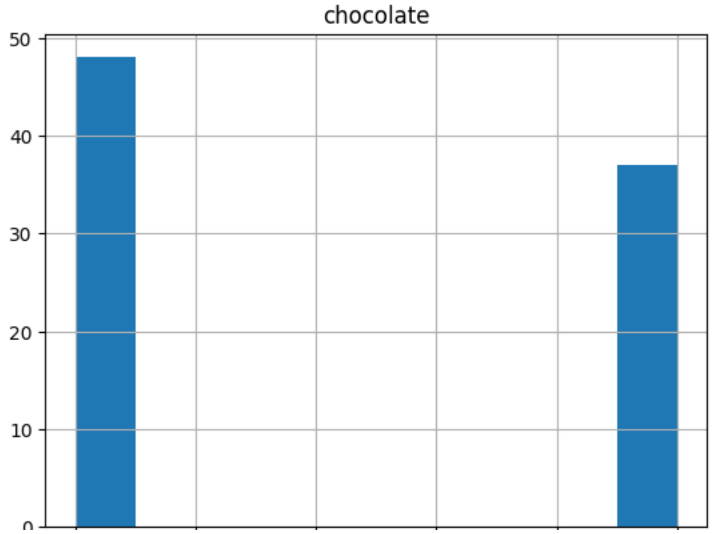
```
df.chocolate.value_counts(normalize=True).sort_index()*100
```

```
0    56.470588
1    43.529412
Name: chocolate, dtype: float64
```

Here is a visualization:

```
df.hist(column='chocolate')
```

```
array([[<Axes: title={'center': 'chocolate'}>]], dtype=object)
```



Look at subgroups. How much of a candy is sugar on average, by whether it has chocolate or not? Chocolate has 51.2% sugar content vs. non-chocolate candy is only 45.2% sugar content.

```
df.groupby(['chocolate'])['sugarpercent'].mean()
```

```
chocolate
0    0.452937
1    0.512000
Name: sugarpercent, dtype: float64
```

```
df.groupby(['chocolate'])['sugarpercent'].std()
```

```
chocolate
0    0.310936
1    0.241524
Name: sugarpercent, dtype: float64
```

What about 2 categorical variables. Can a candy be both chocolatey and fruity at the same time?

```
pd.crosstab(df.fruity, df.chocolate, normalize='columns')*100
```

chocolate	**0**	**1**
fruity		
0	22.916667	97.297297
1	77.083333	2.702703

probability when chocolate also has fruity: 2.7

We can visualize this relationship too:

```
d_pct = (df.groupby(['chocolate','fruity'])['chocolate'].count()/df.groupby(['chocolate'])['fruity'].count())
d_pct.unstack().plot.bar(stacked=True)
```

```
<Axes: xlabel='chocolate'>
```

A NEW EXAMPLE - VOTING ... grab the data online:

```
import pandas as pd
url1 = 'https://raw.githubusercontent.com/fivethirtyeight/data/master/non-voters/nonvoters_data.csv'
df1 = pd.read_csv(url1)
```

```
df1.head()
```

	RespId	weight	Q1	Q2_1	Q2_2	Q2_3	Q2_4	Q2_5	Q2_6	Q2_7	...	Q30	Q31	Q32	Q33	ppage	educ	race	gender
0	470001	0.7516	1	1	1	2	4	1	4	2	...	2	NaN	1.0	NaN	73	College	White	Female
1	470002	1.0267	1	1	2	2	3	1	1	2	...	3	NaN	NaN	1.0	90	College	White	Female
2	470003	1.0844	1	1	1	2	2	1	1	2	...	2	NaN	2.0	NaN	53	College	White	Male
3	470007	0.6817	1	1	1	1	3	1	1	1	...	2	NaN	1.0	NaN	58	Some college	Black	Female
																	High		

```
df1.income_cat.value_counts(normalize=True).sort_index()*100
```

```
$125k or more     23.886223
$40-75k           23.920493
$75-125k          27.895819
Less than $40k    24.297464
Name: income_cat, dtype: float64
```

The more education, the younger the voter.

```
df1.groupby(['educ'])['ppage'].mean()
```

```
educ
College                 49.393562
High school or less     54.309577
Some college            52.080702
Name: ppage, dtype: float64
```

Let's reorganize the categories to go up in a logical way:

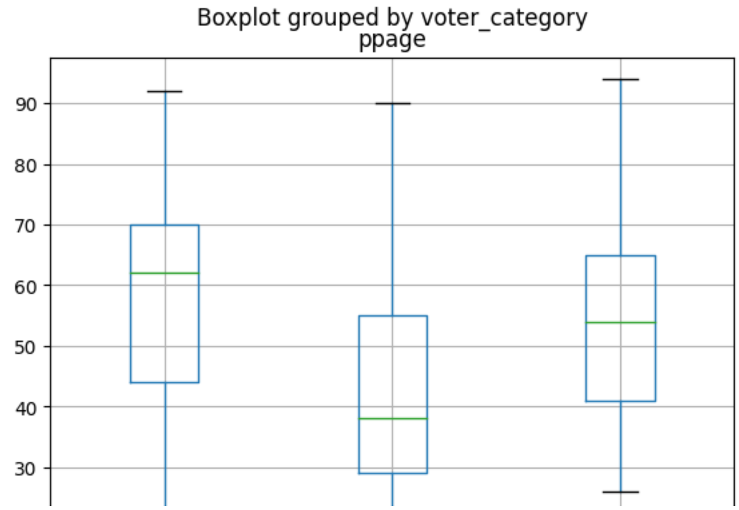
```
df1.educ = pd.Categorical(df1.educ,
                      categories=["High school or less","Some college","College"],
                      ordered=True)
```

```
df1.voter_category = pd.Categorical(df1.voter_category,
                      categories=["rarely/never","sporadic","always"],
                      ordered=True)
```

Always voters are on average older:

```
%matplotlib inline
df1.boxplot(column='ppage', by=['voter_category'])
```

```
<Axes: title={'center': 'ppage'}, xlabel='[voter_category]'>
```



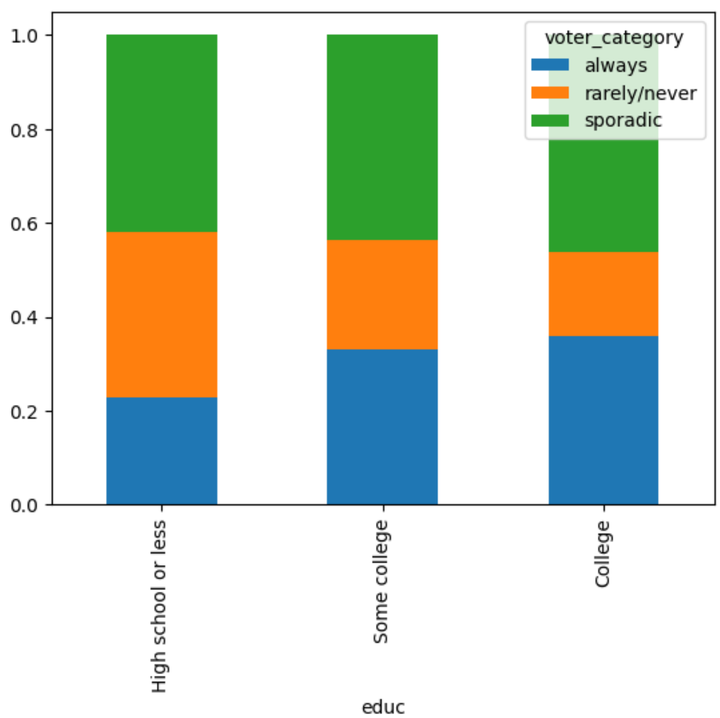
The more education, the more always voting:

```
pd.crosstab(df1.voter_category, df1.educ, normalize='columns')*100
```

educ	High school or less	Some college	College
voter_category			
always	22.884187	33.099415	35.793991
rarely/never	35.133630	23.216374	18.154506
sporadic	41.982183	43.684211	46.051502

```
d_pct = (df1.groupby(['educ','voter_category'])['voter_category'].count()/df1.groupby(['educ'])['voter_category'].count())
d_pct.unstack().plot.bar(stacked=True)
```

```
<Axes: xlabel='educ'>
```



Let's recode this variable:

```
df1["partyid"] = df1["Q30"] ## rename Q30 ##

partyid_temp = pd.Categorical(df1["partyid"], categories = [1, 2, 3, 4, 5, -1], ordered = True)  ## make ordered categories ##

df1["partyid"] = partyid_temp.rename_categories(["Repub", "Dem", "Ind", "Other", "None", "No response"])  ## give them labels ##

df1.loc[df1["partyid"] == "No response","partyid"] = np.nan ## get rid of no response ##

df1.partyid.value_counts(normalize=True).sort_index()*100 ## see if it worked ##
```

```
Repub          27.505183
Dem            34.588804
Ind            24.619903
Other           1.710435
None           11.575674
No response     0.000000
Name: partyid, dtype: float64
```

A FINAL EXAMPLE - GSS

```
from google.colab import files
uploaded = files.upload()

import io
g = pd.read_csv(io.BytesIO(uploaded['GSS.2006.lab.csv']))

g.head()
```

Choose Files GSS.2006.lab.csv

- **GSS.2006.lab.csv**(text/csv) - 38405325 bytes, last modified: 5/28/2023 - 100% done

```
Saving GSS.2006.lab.csv to GSS.2006.lab.csv
<ipython-input-7-12496f6038f7>:5: DtypeWarning: Columns (1265,1273,1277,1291,1293,1294,1507,1582,1583,1584,1585,1586,1587,158
  g = pd.read_csv(io.BytesIO(uploaded['GSS.2006.lab.csv']))
```

	Unnamed: 0	vpsu	vstrat	adults	ballot	dateintv	famgen	form	formwt	gender1	...	zspaneng	zwtss	zwtssnr	zwtssall	zi_a
0	0	1	1957	1	3	316	2	1	1	2	...	English	0.43	0.49	0.4297	45- yea (
1	1	1	1957	2	2	630	1	2	1	2	...	English	1.91	2.16	1.9096	18- yea (
2	2	1	1957	2	2	314	2	1	1	2	...	Spanish	0.86	0.97	0.8593	60- yea (

```
column_headers = list(g.columns.values)
print("The Column Header :", column_headers)
```

```
The Column Header : ['Unnamed: 0', 'vpsu', 'vstrat', 'adults', 'ballot', 'dateintv', 'famgen', 'form', 'formwt', 'gender1',
```

```
%%capture
## %%capture will suppress the output from this command ##
## I created a new variable called col and it is made up of the column names from d##
for col in g.columns:
    print(col)
```

```
g.big5a1.value_counts(normalize=True).sort_index()*100
```

```
1.0    13.297872
2.0    42.021277
3.0    16.821809
4.0    21.077128
5.0     6.781915
Name: big5a1, dtype: float64
```

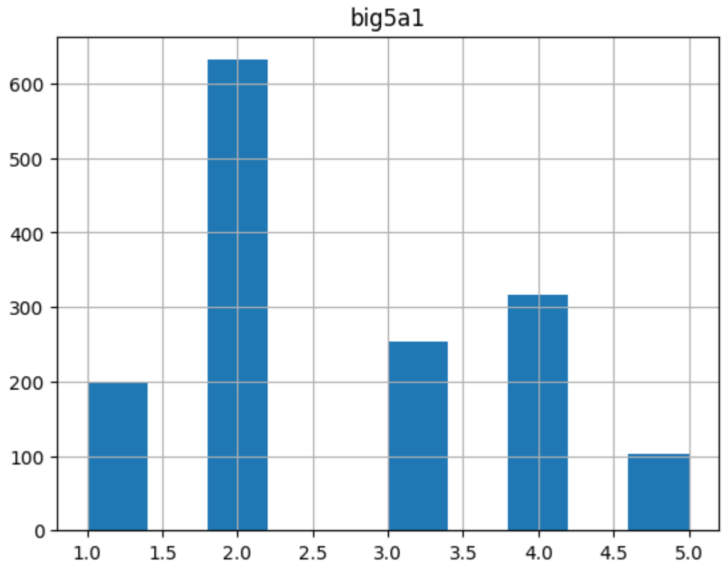
```
g.zbig5a1.value_counts(normalize=True).sort_index()*100
```

```
Agree                         41.633729
Can't choose                   0.790514
Disagree                      20.882740
Neither agree nor disagree    16.666667
```

```
No answer                    0.131752
Strongly agree              13.175231
Strongly disagree            6.719368
Name: zbig5a1, dtype: float64
```

```
g.hist(column='big5a1')
```

```
array([[<Axes: title={'center': 'big5a1'}>]], dtype=object)
```



I first looked at:

1) Categorical variable: widowed status whether or not the participant is currently married, separated, or divorced 2) Continuous variable: mental health status based on the question: Now thinking about your mental health, which includes stress, depression, and emotional problems, for how many days during the past 30 days was your mental health not good? (Ranging from 0-30 days, indicated in Table 458)

I looked at the differences in mean and standard deviations of poor mental health days, by marriage status. I wanted to see whether widowed people are unhappier than married ones. I hypothesized that non-widowed people are happier than those who are widowed. When running the mean code, I found that widowed people have a higher average of bad mental health days (4.04) compared to their non-widowed counterparts (2.75) I realized that cross-tabulating a categorical and continuous variable is not efficient to show visually because it'd show how frequent it would be for widows to have # of bad days, and computing the average is better for this information.

```
g.groupby(['widowed'])['mntlhlth'].mean()
```

```
widowed
1.0    4.041667
2.0    2.748943
Name: mntlhlth, dtype: float64
```

1 = male; 2 = female; females reported having more poor mental health days than males

```
g.groupby(['zwidowed'])['mntlhlth'].mean()
```

```
zwidowed
Don't know         NaN
No            2.748943
No answer     0.000000
Yes           4.041667
Name: mntlhlth, dtype: float64
```

```
g.groupby(['widowed'])['mntlhlth'].std()
```

```
widowed
1.0    8.858693
2.0    6.369239
Name: mntlhlth, dtype: float64
```

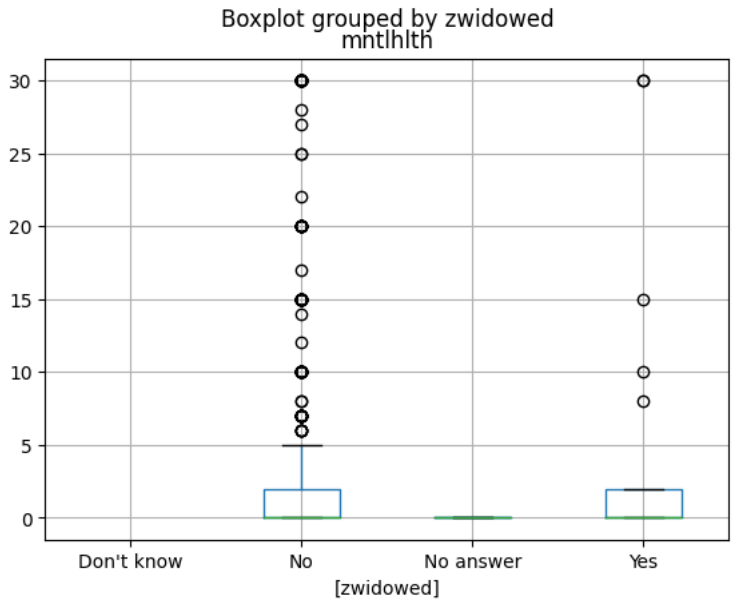
```
g.groupby(['zwidowed'])['mntlhlth'].std()
```

```
zwidowed
Don't know         NaN
```

```
No            6.369239
No answer     0.000000
Yes           8.858693
Name: mntlhlth, dtype: float64
```

```
%matplotlib inline
g.boxplot(column='mntlhlth', by=['zwidowed'])
```

```
<Axes: title={'center': 'mntlhlth'}, xlabel='[zwidowed]'>
```




```
pd.crosstab(g.widowed, g.mntlhlth, normalize='columns')*100
```

mntlhlth	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	10.0	12.0	14.0	15.0	17.0	20
widowed															
1.0	2.171137	0.0	2.105263	0.0	0.0	0.0	0.0	0.0	33.333333	3.448276	0.0	0.0	3.571429	0.0	[illegible]
2.0	97.828863	100.0	97.894737	100.0	100.0	100.0	100.0	100.0	66.666667	96.551724	100.0	100.0	96.428571	100.0	10

I wanted to look at some cross-tabulation data visually and picked another categorical variable to look at. Now, I'm looking at 2 categorical data: the same widowed status and health status. In the same vein as my first hypothesis between being widowed and mental health, I hypothesized that widowed individuals are more likely to say that they have the worst health. I computed the cross-tabulation to visualize the data and see that on average, there are more non-widowed individuals who say that their health is in "Excellent" condition (orange) than their widowed counterparts. On average, there are more widowed individuals who say that their health is in "Poor" condition (purple) than their non-widowed counterparts. In other words, non-widowed individuals are in better health conditions than their widowed-counter parts.

Combining what I found with widowed status and mental health: Although widowed individuals on average reported more "bad mental health days" than non-widowed individuals, non-widowed individuals on average report better health conditions than their widowed counterparts.

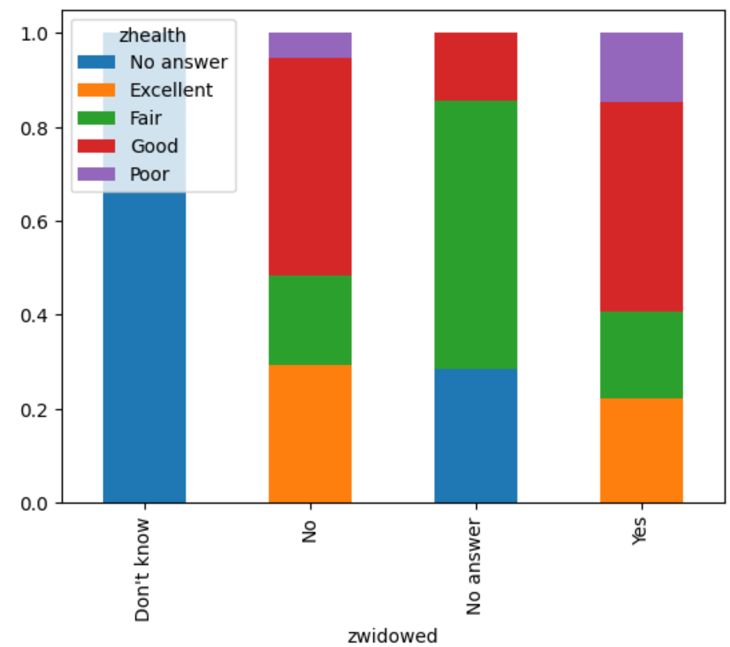
Although there shows a correlation between what I found above, it does not necessarily mean causation, ie. individuals who are widowed do not mean they're bound to have more bad mental health days, or non-widows would always have better health.

```
pd.crosstab(g.zwidowed, g.zhealth, normalize='columns')*100
```

zhealth	Excellent	Fair	Good	No answer	Poor
zwidowed					
Don't know	0.000000	0.000000	0.000000	33.333333	0.000000
No	98.273381	96.916300	97.735507	0.000000	94.029851
No answer	0.000000	0.881057	0.090580	66.666667	0.000000
Yes	1.726619	2.202643	2.173913	0.000000	5.970149

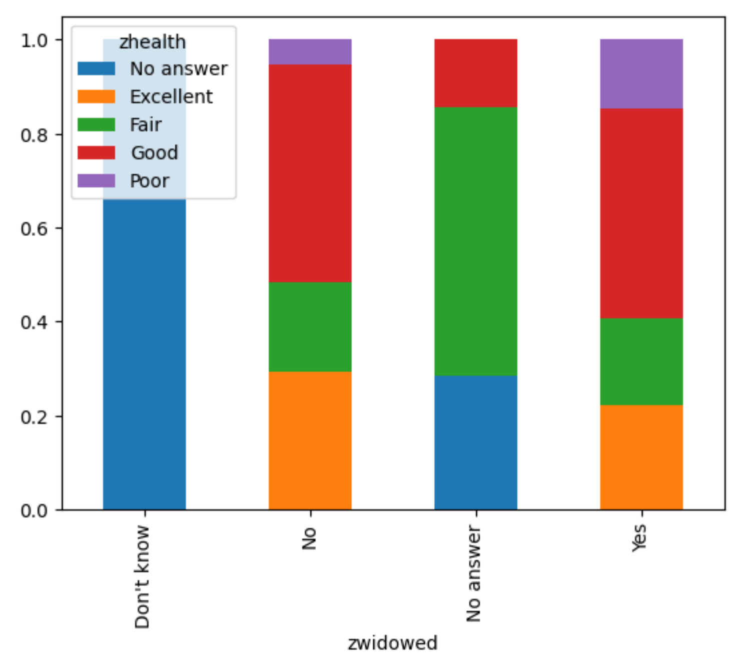
```
g_pct = (g.groupby(['zwidowed', 'zhealth'])['zwidowed'].count()/g.groupby(['zwidowed'])['zhealth'].count())
g_pct.unstack().plot.bar(stacked=True)
```

```
<Axes: xlabel='zwidowed'>
```




```
g_pct = (g.groupby(['zwidowed', 'zhealth'])['zwidowed'].count()/g.groupby(['zwidowed'])['zhealth'].count())
g_pct.unstack().plot.bar(stacked=True)
```

```
<Axes: xlabel='zwidowed'>
```



✓ 0s completed at 10:51 PM