Nhu M Vo

Lab #2

```
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
```

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

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

d.head()
```

Choose Files GSS.2006.csv

- **GSS.2006.csv**(text/csv) - 8232424 bytes, last modified: 6/7/2023 - 100% done

```
Saving GSS.2006.csv to GSS.2006.csv
```

	vpsu	vstrat	adults	ballot	dateintv	famgen	form	formwt	gender1	hompop	...	away7	gender14	old14	rela
0	1	1957	1	3	316	2	1	1	2	3	...	NaN	NaN	NaN	
1	1	1957	2	2	630	1	2	1	2	2	...	NaN	NaN	NaN	
2	1	1957	2	2	314	2	1	1	2	2	...	NaN	NaN	NaN	
3	1	1957	1	1	313	1	2	1	2	1	...	NaN	NaN	NaN	
4	1	1957	3	1	322	2	2	1	2	3	...	NaN	NaN	NaN	

5 rows × 1261 columns

1. Recode 2 variable into new categories. They can both be continuous-ish or both be nominal-ish, or one of each. Tell me what you did and explain the variable(s).

Create a number of categories for frequency of going to a bar/tavern: low, medium, high

```
## 3 options of low, medium, high ##

conditions = [
    (d['socbar'] < 3) & (d['socbar'] > 0),
    (d['socbar'] >2 ) & (d['socbar'] <6),
    (d['socbar'] > 5)]
choices = [1, 2, 3]
d['cut'] = np.select(conditions, choices, default=np.nan)
```

```
# Look at the results

d.cut.describe()
```

```
count    1989.000000
mean        2.582705
std         0.609138
min         1.000000
25%         2.000000
50%         3.000000
75%         3.000000
max         3.000000
Name: cut, dtype: float64
```

```
## How many of each category are there?

d.cut.value_counts()
```

```
3.0    1286
2.0     576
1.0     127
Name: cut, dtype: int64
```

Check the recoding: It is "1" for categories more than 0 and less than 3, it is "2" for categories from 3 to 5, and it is "3" for categories greater than 5 (high, medium, low, respectively). The columns for each 1,2,3 add up to 100%

Note: 1 - almost everyday, 2 - once of twice a week, 3 - several times a month, 4 - about once a month, 5 - several times a year, 6 - about once a year

```
res = pd.crosstab(d.socbar, d.cut)
res.astype('float').div(res.sum(axis=0), axis=1)
```

cut socbar	1.0	2.0	3.0
1.0	0.110236	0.000000	0.0000
2.0	0.889764	0.000000	0.0000
3.0	0.000000	0.211806	0.0000
4.0	0.000000	0.347222	0.0000
5.0	0.000000	0.440972	0.0000
6.0	0.000000	0.000000	0.2014
7.0	0.000000	0.000000	0.7986

```
#Simply another way to do the same thing as above:  A nice function someone wrote to do the same thing:

def binning(col, cut_points, labels=None):
  #Define min and max values:
  minval = col.min()
  maxval = col.max()

  #create list by adding min and max to cut_points
  break_points = [minval] + cut_points + [maxval]

  #if no labels provided, use default labels 0 ... (n-1)
  if not labels:
    labels = range(len(cut_points)+1)

  #Binning using cut function of pandas
  colBin = pd.cut(col,bins=break_points,labels=labels,include_lowest=True)
  return colBin
```

```
#Binning attend:

cut_points = [2,5]
labels = ["low","medium","high"]
d["socbar_cut"] = binning(d["socbar"], cut_points, labels)
##print pd.value_counts(d["attend_cut"], sort=False)##



## See it works the same way...

summary = d.socbar_cut.describe()
summary = summary.transpose()
summary
```

```
count     1989
unique       3
top       high
freq      1286
Name: socbar_cut, dtype: object
```

```
## See it works the same way...

d.socbar_cut.value_counts()
```

```
high      1286
medium     576
low        127
Name: socbar_cut, dtype: int64
```

```
## See it works the same way...

res = pd.crosstab(d.socbar, d.socbar_cut)
res.astype('float').div(res.sum(axis=0), axis=1)
```

socbar_cut	low	medium	high
socbar			
1.0	0.110236	0.000000	0.0000
2.0	0.889764	0.000000	0.0000
3.0	0.000000	0.211806	0.0000
4.0	0.000000	0.347222	0.0000
5.0	0.000000	0.440972	0.0000
6.0	0.000000	0.000000	0.2014
7.0	0.000000	0.000000	0.7986

Below is a binary recode for "how often you go to a bar/tavern": with 0 being <5 times, and 1 being more than 4 times OR everything greater than or equal to 5

```
##Simple binary cut##

conditions = [
    (d['socbar'] < 5) ,
    (d['socbar'] >4 )]
choices = [0,1]
d['high'] = np.select(conditions, choices, default=np.nan)
```

Check the recoding: It is 0 for all categories less than 5 and 1 for everything greater than or equal to 5

```
res = pd.crosstab(d.socbar, d.high)
res.astype('float').div(res.sum(axis=0), axis=1)
```

high	0.0	1.0
socbar		
1.0	0.031180	0.000000
2.0	0.251670	0.000000
3.0	0.271715	0.000000
4.0	0.445434	0.000000
5.0	0.000000	0.164935
6.0	0.000000	0.168182
7.0	0.000000	0.666883

Let's look at another variable - this is a question about: "Have you ever had sex with someone other than your husband or wife while you were married?"(evstray) with 1 being Yes, 2 being No. The second variable in this Lab Assignment that I'm going to recode is getting rid of the "Never married" (column 3).

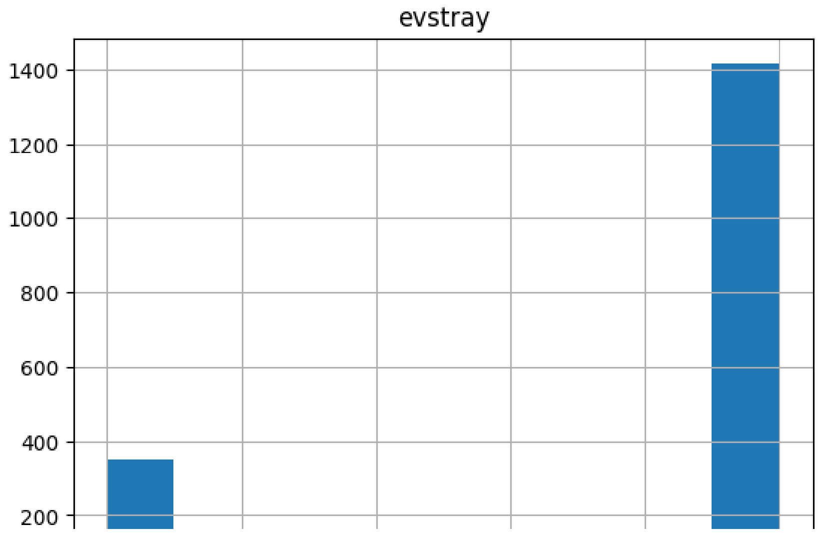
```
## Table 2 ##

my_tab = pd.crosstab(index=d["evstray"],  # Make a crosstab
                              columns="count")

d.loc[d['evstray'] == 3.0, 'evstray'] = np.nan
#look into location where value in evstray=3, replace it with np.nan (NaN value)

display(my_tab)
def compute_percentage(x):
      pct = float(x/my_tab['count'].sum()) * 100
      return round(pct, 2)

my_tab['percentage'] = my_tab.apply(compute_percentage, axis=1)


my_tab
```

col_0	count
evstray	
1.0	350
2.0	1414
3.0	623

col_0	count	percentage
evstray		
1.0	350	14.66
2.0	1414	59.24
3.0	623	26.10

```
d.hist(column='evstray')
```

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



2. Use one (or both) of your recoded variables to do a cross-tabulation. Explain your results.

"evstray" (Independent variable: whether or not one cheats, with 1 being yes, and 2 being no) is the column variable and "cut" (Dependent variable: how often an individual goes to a bar/tavern, with 1 being high frequency and 3 being low frequency) is the row variable. The below results indicate that for the people who cheat on their spouse, 9.09% are going to the bar frequently while for non-cheaters, 4.13% of them are going to the bar frequently, which is contrary with my hypothesis because people cheat on their spounses at the bar/tavern.

```
res = pd.crosstab(d.cut, d.evstray)
res.astype('float').div(res.sum(axis=0), axis=1)

display(d.evstray)
```

```
0       NaN
1       NaN
2       NaN
3       NaN
4       NaN
       ...
4505    2.0
4506    2.0
4507    NaN
4508    NaN
4509    2.0
Name: evstray, Length: 4510, dtype: float64
```

```
res = pd.crosstab(d.cut, d.evstray)
res.astype('float').div(res.sum(axis=0), axis=1)
```

evstray	1.0	2.0
cut		
1.0	0.090909	0.041322
2.0	0.367965	0.253099
3.0	0.541126	0.705579

Below, I'm using pandas "crosstab" function to get column percentages now.

(I put the addition "*100" at the end because I wanted it to show up as percentages)

```
res.p = pd.crosstab(d['cut'], d['evstray'], normalize='columns')*100
res.p
```

```
<ipython-input-18-dfeb775cf6e9>:1: UserWarning: Pandas doesn't allow columns to be created via a new attribute
  res.p = pd.crosstab(d['cut'], d['evstray'], normalize='columns')*100
```

evstray	1.0	2.0
cut		
1.0	9.090909	4.132231
2.0	36.796537	25.309917
3.0	54.112554	70.557851

3. Run a linear regression with 1 independent and 1 dependent variable; make all of the recodes necessary to make the model as easy to interpret as possible; and explain your results.

Asking the question: How do you think "evstray" - whether or not one cheats on their spouse (IV) should be related to "cut" - their frequency of going to a bar/tavern (DV). In other words, how does the likelyhood of cheating on a spouse (Yes/No) affect their frequency of going to the bar/tavern

```
lm = smf.ols(formula = 'evstray~cut', data = d).fit()
print (lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                evstray   R-squared:                       0.021
Model:                            OLS   Adj. R-squared:                  0.020
Method:                 Least Squares   F-statistic:                     25.89
Date:                Mon, 12 Jun 2023   Prob (F-statistic):           4.20e-07
Time:                        02:08:32   Log-Likelihood:                -572.91
No. Observations:                1199   AIC:                             1150.
Df Residuals:                    1197   BIC:                             1160.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      1.5479      0.052     29.642      0.000       1.445       1.650
cut            0.0989      0.019      5.088      0.000       0.061       0.137
==============================================================================
Omnibus:                      264.735   Durbin-Watson:                   1.996
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              468.245
Skew:                          -1.515   Prob(JB):                    2.10e-102
Kurtosis:                       3.441   Cond. No.                         14.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Conclusion: Knowing whether or not one cheats do not help predict how frequent they go to the bar/tavern, and vice versa. A coefficient of 0.09 indicates no obvious relationship between whether one cheats (Yes/No) and frequency of going to a bar/tavern. The p-values are - which means the results are statistically significant.

4. Plot two variables, either as a scatter plot or boxplot; add in trend/regression lines; and explain your results.

Here is plotting of "evstray" (whether or not one cheats) against "cut" (how often an individual goes to bar/tavern).

Note:

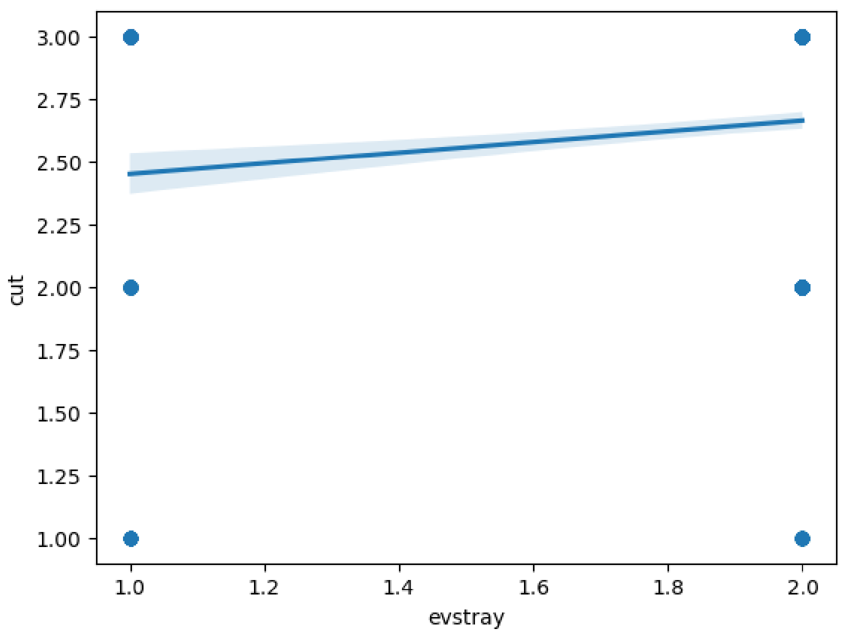
"cut": (Dependent variable: how often an individual goes to a bar/tavern, with 1 being high frequency and 3 being low frequency)

"evstray": (Independent variable: whether or not one cheats, with 1 being yes, and 2 being no)

The results below show that the individuals that cheat on their spouses go to tavern/bars less frequently than their counterparts who don't cheat

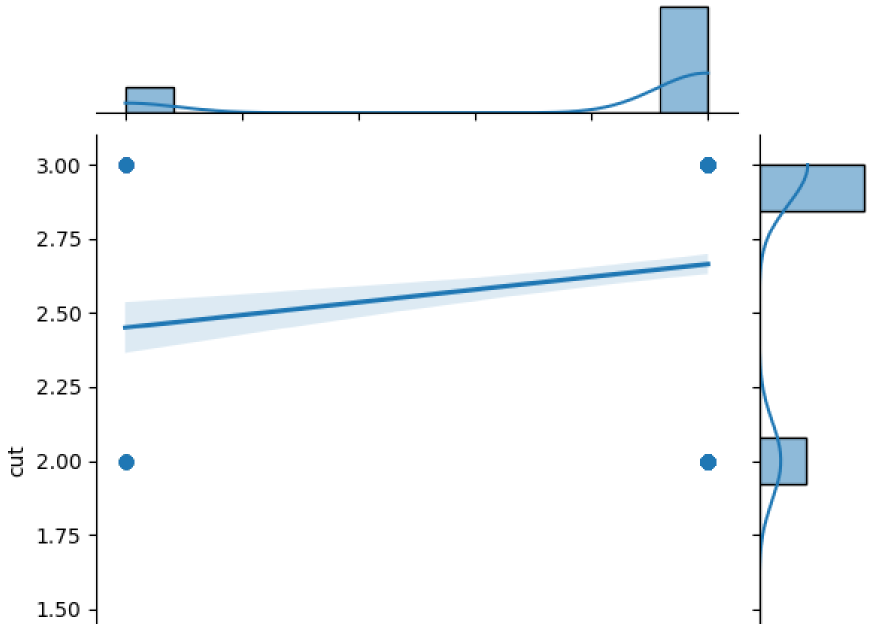
```
import seaborn as sns
sns.regplot(x=d['evstray'],y=d['cut'])
```

```
<Axes: xlabel='evstray', ylabel='cut'>
```




```
sns.jointplot(data=d, x='evstray', y='cut', kind="reg")
```

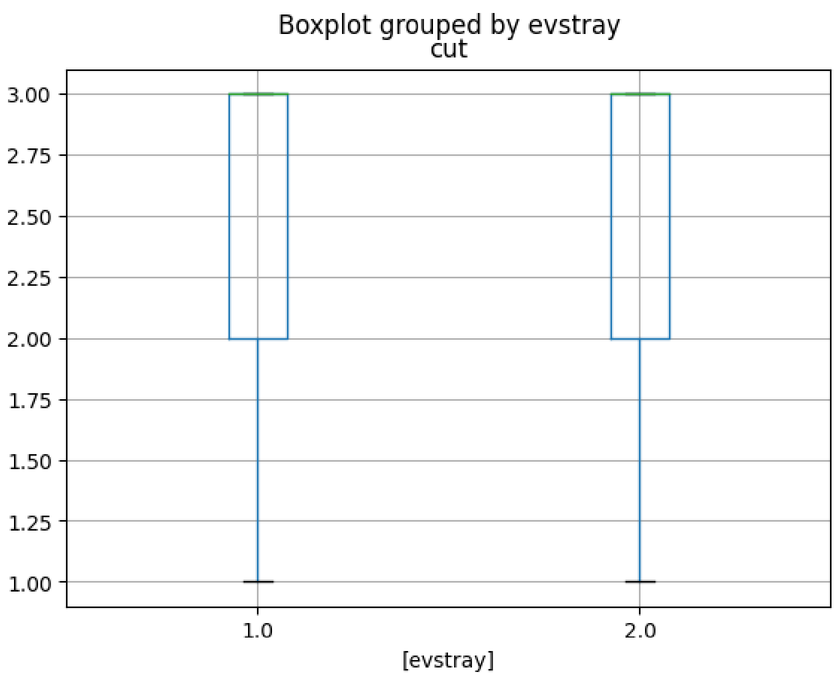
```
<seaborn.axisgrid.JointGrid at 0x7f3ff35cf010>
```



this jointplot with a negative regression line shows that the relationship between whether or not one cheats on their spouse is negatively correlated with how often they go to a pub/tavern

```
%matplotlib inline
d.boxplot(column='cut', by=['evstray'])
```

```
<Axes: title={'center': 'cut'}, xlabel='[evstray]'>
```



✓ 1s completed at 10:08 PM