Practical 3 - JosiahTeh

December 8, 2021

1 First Name: Josiah

2 Last Name: Teh

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[3]:
```
nesarc = pd.read_csv('nesarc.csv', low_memory=False)
pd.set_option('display.float_format', lambda x:'%f'%x)
```

3 From Prac 1

4 Columns/Data used in Prac 1

[4]:
```
nesarc['S2AQ5B'] = pd.to_numeric(nesarc['S2AQ5B'], errors='coerce') #convert␣
 ↪variable to numeric
nesarc['S2AQ5D'] = pd.to_numeric(nesarc['S2AQ5D'], errors='coerce') #convert␣
 ↪variable to numeric
nesarc['S2AQ5A'] = pd.to_numeric(nesarc['S2AQ5A'], errors='coerce') #convert␣
 ↪variable to numeric
nesarc['S2BQ1B1'] = pd.to_numeric(nesarc['S2BQ1B1'], errors='coerce') #convert␣
 ↪variable to numeric
nesarc['AGE'] = pd.to_numeric(nesarc['AGE'], errors='coerce') #convert variable␣
 ↪to numeric
```

5 From Prac 2

6 A subset of nesarc data, with the following criteria

7 Age from 26 to 50

8 Beer drinking status - S2AQ5A = Y

[5]:
```
sub1=nesarc[(nesarc['AGE']>=26) & (nesarc['AGE']<=50) & (nesarc['S2AQ5A']==1)]
sub2=sub1.copy()
```

9 From Prac 2

10 SETTING MISSING DATA

[6]:
```
sub2['S2AQ5D']=sub2['S2AQ5D'].replace(99, np.nan)

sub2['S2AQ5B']=sub2['S2AQ5B'].replace(8, np.nan)
sub2['S2AQ5B']=sub2['S2AQ5B'].replace(9, np.nan)
sub2['S2AQ5B']=sub2['S2AQ5B'].replace(10, np.nan)
sub2['S2AQ5B']=sub2['S2AQ5B'].replace(99, np.nan)

sub2['S2BQ1B1']=sub2['S2BQ1B1'].replace(9, np.nan)
```

11 From Prac 2

12 Recode data

[7]:
```
recode2 = {1:30, 2:26, 3:14, 4:8, 5:4, 6:2.5, 7:1}
sub2['BEER_FEQMO']= sub2['S2AQ5B'].map(recode2)

recode3 = {2:0, 1:1}
sub2['S2BQ1B1']= sub2['S2BQ1B1'].map(recode3)
```

13 Plot bar chart for S2BQ1B1

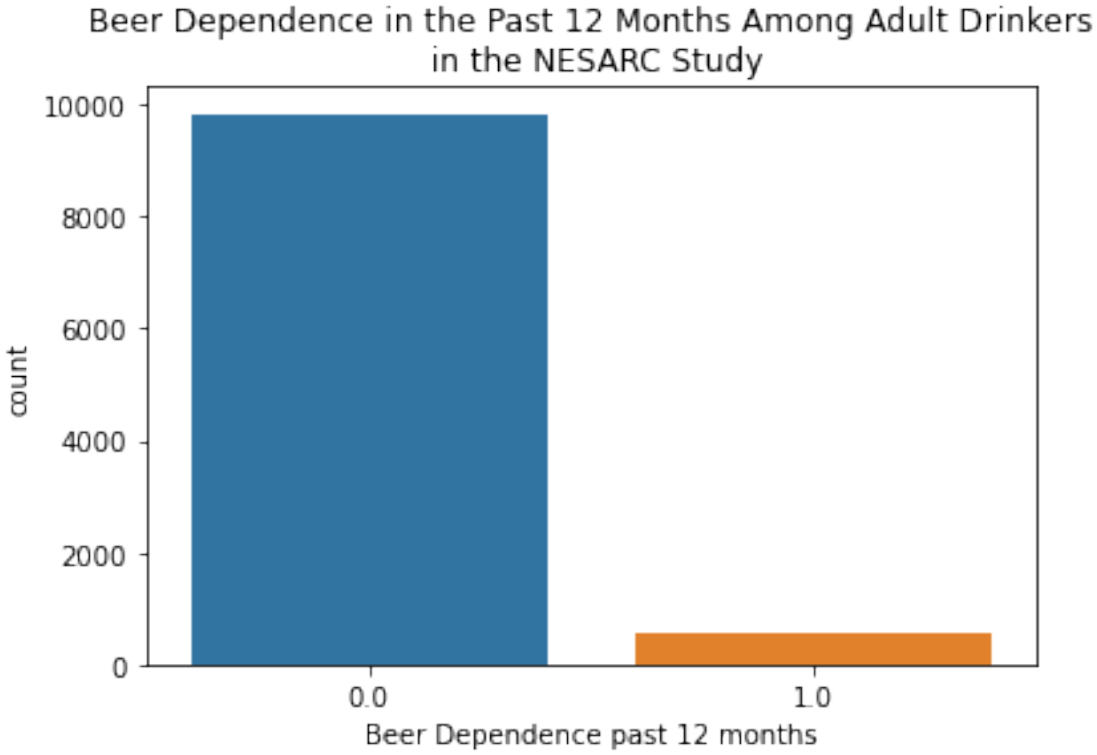
14 convert S2BQ1B1 to category data type

[8]:
```
# hint lecture cell 7
sub2["S2BQ1B1"] = sub2["S2BQ1B1"].astype('category')
```

15 Plot bar chart for S2BQ1B1

[10]:
```
# hint lecture cell 8
%matplotlib inline
sns.countplot(x='S2BQ1B1', data = sub2)
plt.xlabel('Beer Dependence past 12 months')
plt.title('Beer Dependence in the Past 12 Months Among Adult Drinkers'+ '\n' +␣
 ↪' in the NESARC Study')
```

[10]:
```
Text(0.5, 1.0, 'Beer Dependence in the Past 12 Months Among Adult Drinkers\n in
the NESARC Study')
```



16 Visualizing Quantitative Variable - histogram

17 From Prac 2

18 Create a secondary variable to estimate the number of beer consumed per month

19 NUMBEERMO_EST

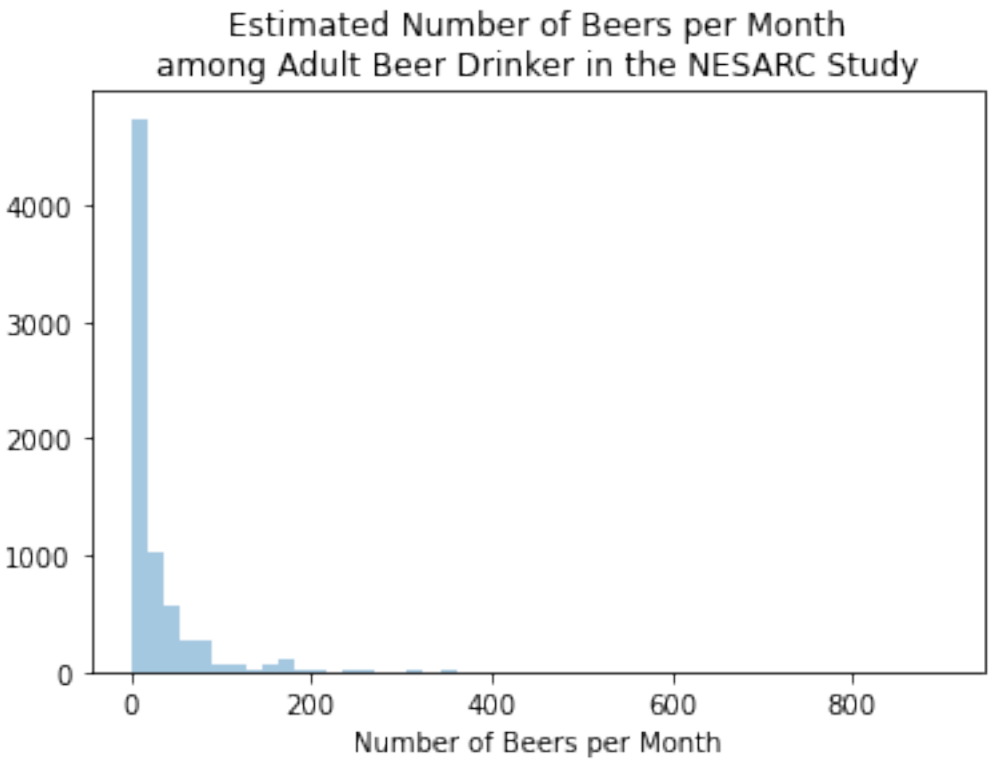
[11]:
```
# hint lecture cell 9
# A secondary variable multiplying the number of beers comsumed and the approx␣
 ↪number of beers consumed/day
sub2['NUMBEERMO_EST']=sub2['BEER_FEQMO'] * sub2['S2AQ5D']
```

20 Visualise the number of beers consumed per month (NUMBEERMO_EST) using a histogram

[12]:
```
# hint lecture cell 10
%matplotlib inline
sns.distplot(sub2['NUMBEERMO_EST'].dropna(), kde = False)
plt.xlabel('Number of Beers per Month')
plt.title('Estimated Number of Beers per Month' + '\n' + 'among Adult Beer␣
 ↪Drinker in the NESARC Study')
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[12]:
```
Text(0.5, 1.0, 'Estimated Number of Beers per Month\namong Adult Beer Drinker in
the NESARC Study')
```



21 Calculate the spread and centre of NUMBEERMO_EST

22 Use describe()

[13]:
```
# hint lecture cell 11
# standard deviation and other descriptive statistics for quantitative variables
print('describe number of beers drinking per month')
desc1 = sub2['NUMBEERMO_EST'].describe()
print (desc1)
```

```
describe number of beers drinking per month
count   7303.000000
mean      27.765713
std       49.201312
min        1.000000
25%        4.000000
50%       12.000000
75%       28.000000
max      900.000000
Name: NUMBEERMO_EST, dtype: float64
```

23 Alternative method

24 Calculate descriptive statistics of NUMBEERMO_EST

25 Use mean(), std(), min(), max(), median(), mode()

[14]:
```
# hint lecture cell 12
print('mean')
mean1 = sub2['NUMBEERMO_EST'].mean()
print (mean1)

print('std')
std1 = sub2['NUMBEERMO_EST'].std()
print (std1)

print('min')
min1 = sub2['NUMBEERMO_EST'].min()
print (min1)

print ('max')
max1 = sub2['NUMBEERMO_EST'].max()
print (max1)

print ('median')
median1 = sub2['NUMBEERMO_EST'].median()
print (median1)

print ('mode')
mode1 = sub2['NUMBEERMO_EST'].mode()
print (mode1)
```

```
mean
27.765712720799673
std
49.201312205771465
min
1.0
max
900.0
median
12.0
mode
0   8.000000
dtype: float64
```

26 Calculate descriptive statistics for categorical data

27 S2BQ1B1 - Beer Dependence

28 Use describe()

[15]:
```
# hint lecture cell 11
print ('describe beer dependence')
desc2 = sub2['S2BQ1B1'].describe()
print (desc2)
```

```
describe beer dependence
count    10406.000000
unique       2.000000
top          0.000000
freq      9829.000000
Name: S2BQ1B1, dtype: float64
```

29 What if categorical data was considered as quantitative data

30 S2BQ1B1 - Beer Dependence

31 Convert S2BQ1B1 to quantitative data and

32 Calculate descriptive statistics

33 Use describe()

[16]:
```
sub2['S2BQ1B1'] = pd.to_numeric(sub2['S2BQ1B1']) # convert a numerical variable␣
 ↪to quantitatie
```

[17]:
```
#hint lecture cell 11
print ('describe beer dependence')
desc3 = sub2['S2BQ1B1'].describe()
print (desc3) #descriptor don't have sense
```

```
describe beer dependence
count   10406.000000
mean        0.055449
std         0.228865
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: S2BQ1B1, dtype: float64
```

34 Visualising 2 variable

35 Categorical -> Quantitative - Bar chart

36 Create a secondary variable

37 CARTONPERMONTH - number of beer carton consumed per month

38 assume that there is 24 beer cans in a carton

[18]:
```
# hint lecture cell 16
sub2['CARTONPERMONTH'] = sub2['NUMBEERMO_EST'] / 24
```

[19]:
```
# hint lecture cell 17
c2= sub2.groupby('CARTONPERMONTH').size()
print (c2)
```

```
CARTONPERMONTH
0.041667     477
0.083333     407
0.104167     414
0.125000     172
0.166667     429
            ...
21.666667      1
22.500000      2
26.000000      1
30.000000      2
37.500000      1
Length: 75, dtype: int64
```

39 Group CARTONPERMONTH into 5 groups

40 1 - 5 cartons

41 6 - 10 cartons

42 10 - 15 cartons

43 15 - 20 cartons

44 20 - 25 cartons

45 25 - 30 cartons

46 30 - max cartons

[20]:
```
# hint lecture cell 18
sub2['CARTONCATEGORY'] = pd.cut(sub2.CARTONPERMONTH, [0, 5, 10, 15, 20, 25, 30,␣
↪38])
```

[21]:
```
# hint lecture cell 19
# change format from numeric to categorical
sub2['CARTONCATEGORY'] = sub2['CARTONCATEGORY'].astype('category')
```

47 Print describe of CARTONCATEGORY

[22]:
```
# hint lecture cell 11
print('describe CARTONCATEGORY')
desc4 = sub2['CARTONCATEGORY'].describe()
print (desc4)
```

```
describe CARTONCATEGORY
count       7303
unique         7
top       (0, 5]
freq        7002
Name: CARTONCATEGORY, dtype: object
```

48 Print carton category counts

[23]:
```
# hint lecture cell 20
print('carton category counts')
c7 = sub2['CARTONCATEGORY'].value_counts(sort=False, dropna=True)
print(c7)
```

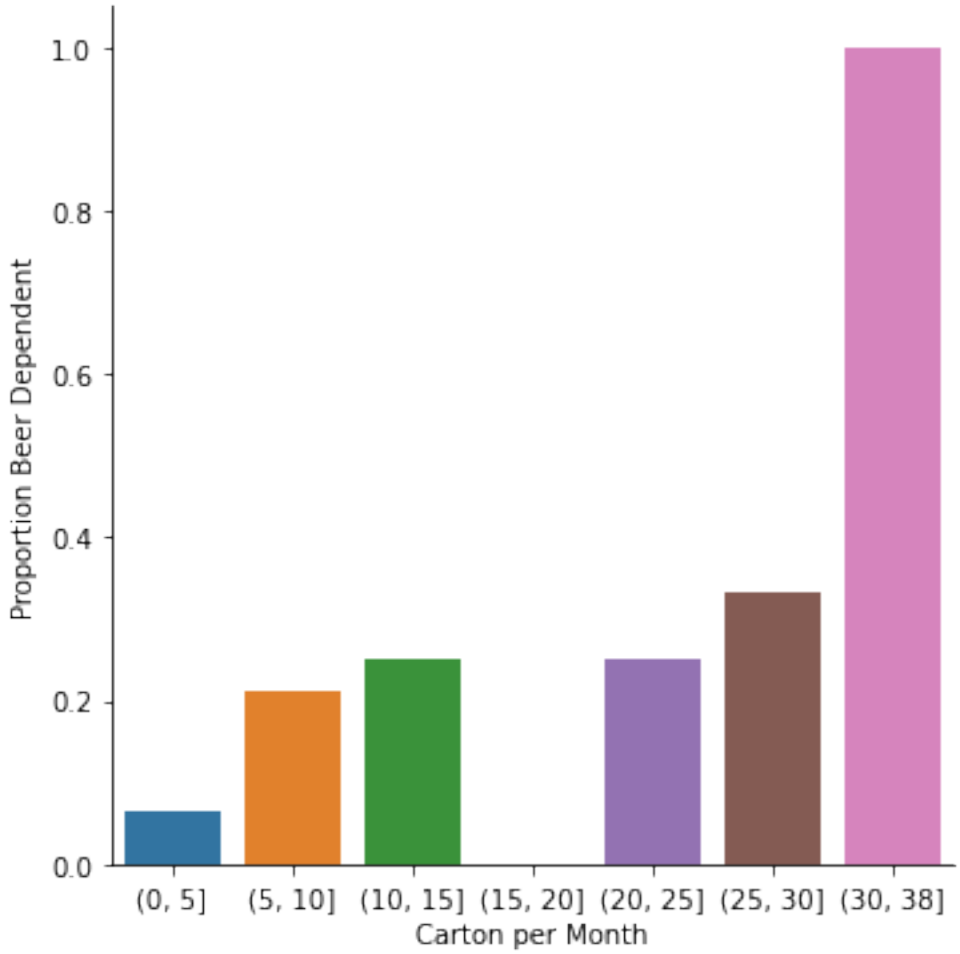
```
carton category counts
(0, 5]      7002
(5, 10]      235
(10, 15]      57
(15, 20]       1
(20, 25]       4
(25, 30]       3
(30, 38]       1
Name: CARTONCATEGORY, dtype: int64
```

49 Chart of bar chart showing the relationship between carton of beer consumed per month (CARTONCATEGORY) and Beer Dependent (S2BQ1B1)

[24]:
```
# hint lecture cell 21
sns.factorplot(x='CARTONCATEGORY', y='S2BQ1B1', data=sub2, kind='bar', ci=None)
plt.xlabel('Carton per Month')
plt.ylabel('Proportion Beer Dependent')
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\categorical.py:3714:
UserWarning: The `factorplot` function has been renamed to `catplot`. The
original name will be removed in a future release. Please update your code. Note
that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in
`catplot`.
  warnings.warn(msg)
```

[24]:
```
Text(6.799999999999997, 0.5, 'Proportion Beer Dependent')
```



50 Visualising 2 variable

51 Categorical -> Categorical - Bar chart

52 Rename race from 1-5 to “White”, “Black”, “NatAm”, “Asian”, “Hispanic”

[25]:
```
# you can rename categorical variable values for graphing if original values␣
 ↪are not informative
# first change the variable format to categorical if you haven't already done so
sub2['ETHRACE2A'] = sub2['ETHRACE2A'].astype('category')
# second create a new variable (PACKCAT) that has the new variable value labels
sub2['ETHRACE2A']=sub2['ETHRACE2A'].cat.rename_categories(["White", "Black",␣
 ↪"NatAm", "Asian", "Hispanic"])
```

53 Function to get 'CARTON_ADAY)

[26]:
```
def CARTON_ADAY (row):
   if row['BEER_FEQMO'] >= 30 :
      return 1
   elif row['BEER_FEQMO'] < 30 :
      return 0

sub2['CARTON_ADAY'] = sub2.apply (lambda row: CARTON_ADAY (row),axis=1)

c4= sub2.groupby('CARTON_ADAY').size()
print(c4)
```

```
CARTON_ADAY
0.000000    6897
1.000000     417
dtype: int64
```

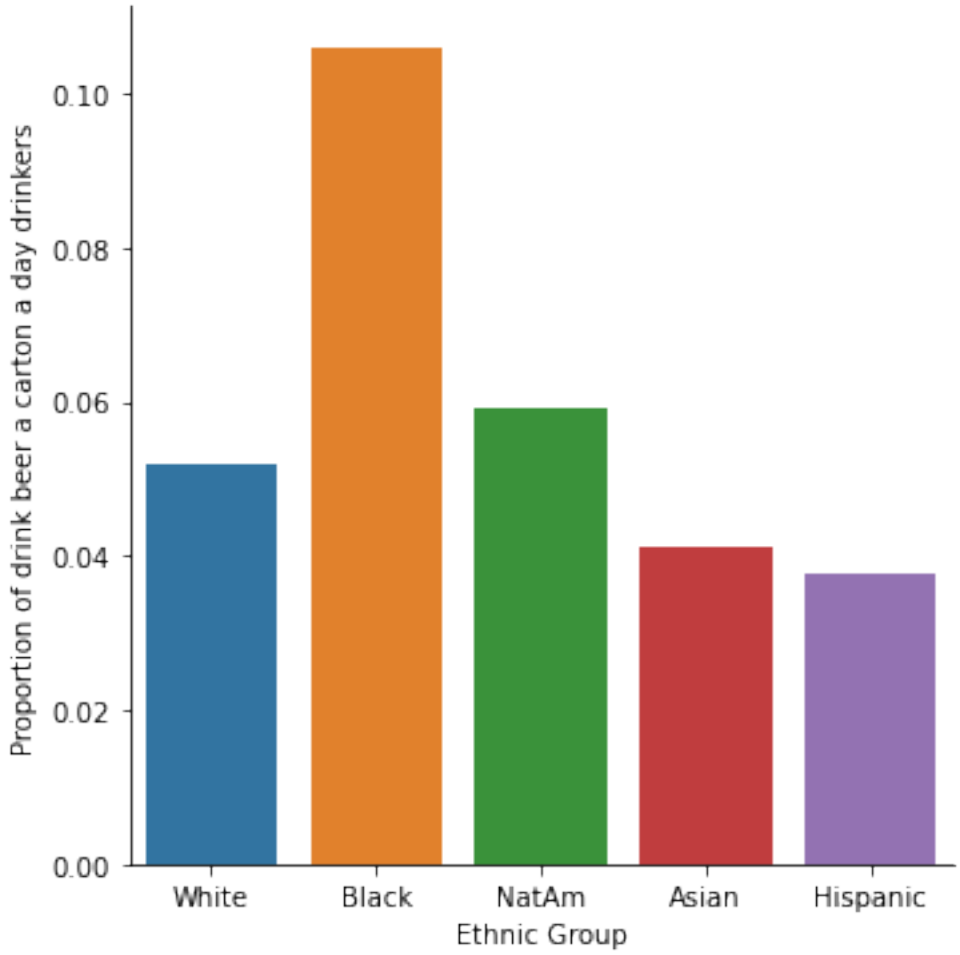
54 Bar Graph showing the relationship between race (ETHRACE2A) and

55 CARTON_ADAY

[27]:
```
# hint lecture cell 24
sns.factorplot(x='ETHRACE2A', y='CARTON_ADAY', data=sub2, kind='bar', ci=None)
plt.xlabel('Ethnic Group')
plt.ylabel('Proportion of drink beer a carton a day drinkers')
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\categorical.py:3714:
UserWarning: The `factorplot` function has been renamed to `catplot`. The
original name will be removed in a future release. Please update your code. Note
that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in
`catplot`.
  warnings.warn(msg)
```

[27]:
```
Text(0.42499999999999716, 0.5, 'Proportion of drink beer a carton a day
drinkers')
```



56 Visualising 2 variable

57 Categorical -> Quantitative - box plot

58 convert age to category data type

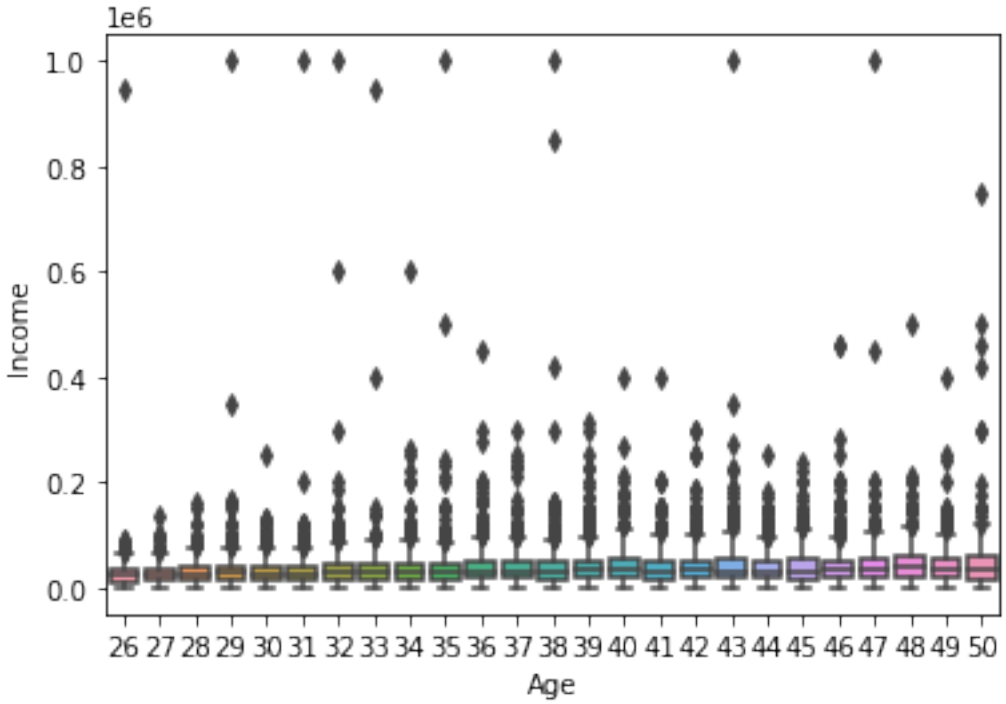
59 convert income (S1Q10A) to numeric data type

[28]:
```
sub2['AGE'] = sub2['AGE'].astype('category')
sub2['S1Q10A'] = pd.to_numeric(sub2['S1Q10A'])
```

60 Box plot to show the relationship between age and income (S1Q10A) among adults aged 26 - 50 years old.

[31]:
```
# hint lecture cell 26
%matplotlib inline
sns.boxplot(x='AGE', y='S1Q10A', data=sub2)
plt.xlabel('Age')
plt.ylabel('Income')
```

[31]:
```
Text(0, 0.5, 'Income')
```



61 Visualising 2 variable

62 Quantitative -> Quantitative - scatter plot

63 Read in gapminder.csv

[32]:
```
pd.set_option('display.float_format', lambda x:'%.2f'%x)

gapminder = pd.read_csv('gapminder.csv', low_memory=False)
gapminder.head()
```

[32]:
```
       country   incomeperperson alcconsumption armedforcesrate  \
0  Afghanistan                              .03        .5696534
1      Albania  1914.99655094922            7.29       1.0247361
2      Algeria  2231.99333515006             .69        2.306817
3      Andorra  21943.3398976022           10.17
4       Angola  1381.00426770244            5.57       1.4613288

  breastcancerper100th      co2emissions  femaleemployrate hivrate  \
0                 26.8          75944000  25.6000003814697
1                 57.4  223747333.333333  42.0999984741211
2                 23.5  2932108666.66667  31.7000007629394      .1
3
4                 23.1         248358000  69.4000015258789       2

    internetuserate lifeexpectancy     oilperperson polityscore  \
0  3.65412162280064         48.673                            0
1  44.9899469578783         76.918                            9
2  12.5000733055148         73.131  .42009452521537           2
3                81
4  9.99995388324075         51.093                           -2

  relectricperperson   suicideper100th        employrate urbanrate
0                     6.68438529968262  55.7000007629394     24.04
1   636.341383366604  7.69932985305786  51.4000015258789     46.72
2   590.509814347428   4.8487696647644              50.5     65.22
3                     5.36217880249023                       88.92
4   172.999227388199  14.5546770095825  75.6999969482422      56.7
```

64 convert 'oilperperson' and 'relectricperperson' to numeric

[35]:
```
# hint lecture cell 28
gapminder['oilperperson'] = pd.to_numeric(gapminder['oilperperson'],␣
 ↪errors='coerce')
gapminder['relectricperperson'] = pd.
 ↪to_numeric(gapminder['relectricperperson'], errors='coerce')
```

65 drop NAN data

[36]:
```
gapminder_clean=gapminder.dropna()
```

66 Scatter plot to show the relationship between Electrcity Use Per Person (relectricperperson) and Oil Use Per Person (oilperperson)

[43]:
```
# hint lecture cell 30
%matplotlib inline
plt.figure()
scatterplot = sns.regplot(x="relectricperperson", y="oilperperson",␣
↪fit_reg=False, data=gapminder_clean)
plt.xlabel('Electrcity Use Per Person')
plt.ylabel('Oil Use Per Person')
plt.title('Scatterplot for the Association Between Electrcity Use Per Person' +␣
↪'\n' + 'and Oil Use Per Person')
```

[43]:
```
Text(0.5, 1.0, 'Scatterplot for the Association Between Electrcity Use Per
Person\nand Oil Use Per Person')
```