19. Descriptive Statistics

September 13, 2024

1 1. Import necessary packages

[3]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory


# Any results you write to the current directory are saved as output.
```

2 2. Load the file

[5]:
```
df = pd.read_csv(r"E:\NIT DataSci Notes\21. 10th, 11th- Intro to Stats,␣
 ↪Descriptive Stats\PROJECT\Inc_Exp_Data.csv")
```

[6]:
```
df.head()
```

[6]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	\
0	5000	8000	3	2000	
1	6000	7000	2	3000	
2	10000	4500	2	0	
3	10000	2000	1	0	
4	12500	12000	2	3000	

	Annual_HH_Income	Highest_Qualified_Member	No_of_Earning_Members
0	64200	Under-Graduate	1
1	79920	Illiterate	1
2	112800	Under-Graduate	1

```
3            97200              Illiterate                    1
4           147000                Graduate                    1
```

3 3. Analyze the data

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 7 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Mthly_HH_Income          50 non-null     int64
 1   Mthly_HH_Expense         50 non-null     int64
 2   No_of_Fly_Members        50 non-null     int64
 3   Emi_or_Rent_Amt          50 non-null     int64
 4   Annual_HH_Income         50 non-null     int64
 5   Highest_Qualified_Member 50 non-null     object
 6   No_of_Earning_Members    50 non-null     int64
dtypes: int64(6), object(1)
memory usage: 2.9+ KB
```

[9]:
```
df.shape
```

[9]:
```
(50, 7)
```

[10]:
```
df.describe()
```

[10]:
```
       Mthly_HH_Income  Mthly_HH_Expense  No_of_Fly_Members  Emi_or_Rent_Amt  \
count        50.000000         50.000000          50.000000        50.000000
mean      41558.000000      18818.000000           4.060000      3060.000000
std       26097.908979      12090.216824           1.517382      6241.434948
min        5000.000000       2000.000000           1.000000         0.000000
25%       23550.000000      10000.000000           3.000000         0.000000
50%       35000.000000      15500.000000           4.000000         0.000000
75%       50375.000000      25000.000000           5.000000      3500.000000
max      100000.000000      50000.000000           7.000000     35000.000000

       Annual_HH_Income  No_of_Earning_Members
count      5.000000e+01              50.000000
mean       4.900190e+05               1.460000
std        3.201358e+05               0.734291
min        6.420000e+04               1.000000
25%        2.587500e+05               1.000000
50%        4.474200e+05               1.000000
75%        5.947200e+05               2.000000
max        1.404000e+06               4.000000
```

[11]:
```
df.describe().T
```

[11]:
```
                        count       mean            std      min       25%  \
Mthly_HH_Income          50.0   41558.00   26097.908979   5000.0   23550.0
Mthly_HH_Expense         50.0   18818.00   12090.216824   2000.0   10000.0
No_of_Fly_Members        50.0       4.06       1.517382      1.0       3.0
Emi_or_Rent_Amt          50.0    3060.00    6241.434948      0.0       0.0
Annual_HH_Income         50.0  490019.04  320135.792123  64200.0  258750.0
No_of_Earning_Members    50.0       1.46       0.734291      1.0       1.0

                            50%       75%        max
Mthly_HH_Income         35000.0   50375.0   100000.0
Mthly_HH_Expense        15500.0   25000.0    50000.0
No_of_Fly_Members           4.0       5.0        7.0
Emi_or_Rent_Amt             0.0    3500.0    35000.0
Annual_HH_Income       447420.0  594720.0  1404000.0
No_of_Earning_Members       1.0       2.0        4.0
```

[12]:
```
df.isna().any()
```

[12]:
```
Mthly_HH_Income             False
Mthly_HH_Expense            False
No_of_Fly_Members           False
Emi_or_Rent_Amt             False
Annual_HH_Income            False
Highest_Qualified_Member    False
No_of_Earning_Members       False
dtype: bool
```

4 4.What is the Mean Expense of a Household?

[14]:
```
df['Mthly_HH_Expense'].mean()
```

[14]:
```
18818.0
```

5 5.What is the Median Household Expense?

[16]:
```
df['Mthly_HH_Expense'].median()
```

[16]:
```
15500.0
```

6 6.What is the Monthly Expense for most of the Households?

[18]:
```
temp = pd.crosstab(index=df['Mthly_HH_Expense'], columns='count')
# .crosstab: Compute a simple cross tabulation of two (or more) factors.
temp.reset_index(inplace=True)
# .reset_index: reset_index() returns a new DataFrame with the index reset and␣
 ↪does not modify the original DataFrame.
# (inplace=True) :  modifies the original DataFrame in place, meaning that the␣
 ↪changes are applied directly to the DataFrame on which the method is called,␣
 ↪and no new DataFrame is returned.
```

[19]:
```
temp
```

[19]:

col_0	Mthly_HH_Expense	count
0	2000	1
1	4500	1
2	5000	1
3	6600	1
4	7000	1
5	8000	3
6	9000	3
7	10000	5
8	10500	1
9	12000	3
10	12300	1
11	13000	1
12	15000	3
13	16000	1
14	18000	1
15	19000	1
16	20000	6
17	22000	1
18	25000	8
19	30000	1
20	40000	2
21	45000	1
22	48000	1
23	50000	2

[20]:
```
# lets make a filter to get Monthly Expense for most of the Households
# --> temp[FILTER]
temp[temp['count'] == df.Mthly_HH_Expense.value_counts().max()]
```

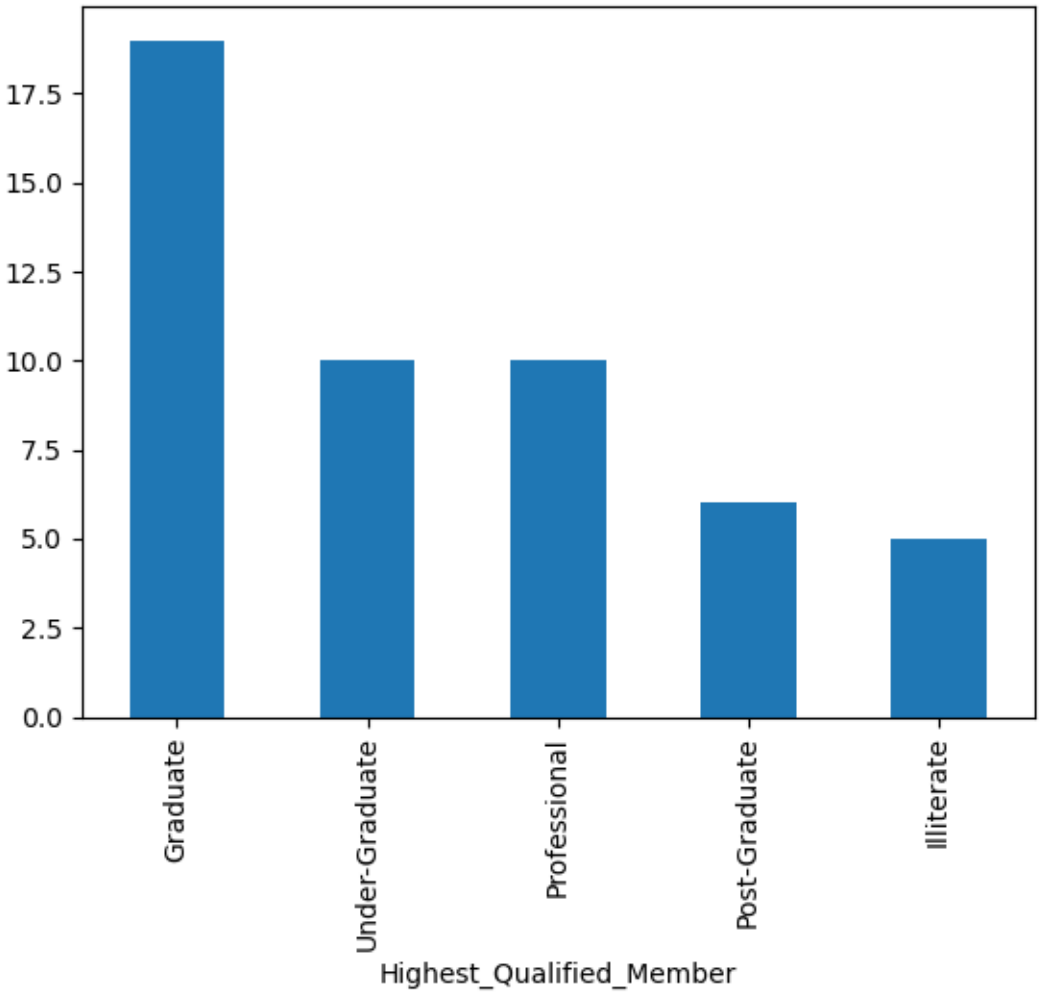
[20]:

col_0	Mthly_HH_Expense	count
18	25000	8

7 7.Plot the Histogram to count the Highest qualified member

[22]:
```
df["Highest_Qualified_Member"].value_counts().plot(kind='bar')
```

[22]:
```
<Axes: xlabel='Highest_Qualified_Member'>
```



8 8.Calculate IQR(difference between 75% and 25% quartile)

[89]:
```
IQR = df["Mthly_HH_Expense"].quantile(0.75)-df["Mthly_HH_Expense"].quantile(0.
↪25)
# .quantile : Return values at the given quantile over requested axis.
IQR
```

[89]:
```
15000.0
```

9 9.Calculate Standard Deviation for first 4 columns.

[26]:
```
df.head(4)
```

[26]:
```
   Mthly_HH_Income  Mthly_HH_Expense  No_of_Fly_Members  Emi_or_Rent_Amt  \
0             5000              8000                  3             2000
1             6000              7000                  2             3000
2            10000              4500                  2                0
3            10000              2000                  1                0

   Annual_HH_Income Highest_Qualified_Member  No_of_Earning_Members
0             64200           Under-Graduate                      1
1             79920               Illiterate                      1
2            112800           Under-Graduate                      1
3             97200               Illiterate                      1
```

[91]:
```
pd.DataFrame(df.iloc[:, 0:5].std().to_frame()).T
# to_frame:  convert a pandas Series into a DataFrame.
# .T :  Transposing means swapping the rows and columns of the DataFrame or␣
 ↪Series.
```

[91]:
```
   Mthly_HH_Income  Mthly_HH_Expense  No_of_Fly_Members  Emi_or_Rent_Amt  \
0     26097.908979      12090.216824           1.517382      6241.434948

   Annual_HH_Income
0     320135.792123
```

10 10.Calculate Variance for first 3 columns.

[82]:
```
pd.DataFrame(df.iloc[:, 0:4].std().to_frame()).T
```

[82]:
```
   Mthly_HH_Income  Mthly_HH_Expense  No_of_Fly_Members  Emi_or_Rent_Amt
0     26097.908979      12090.216824           1.517382      6241.434948
```

11 11.Calculate the count of Highest qualified member.

[41]:
```
df["Highest_Qualified_Member"].value_counts().to_frame().T
```

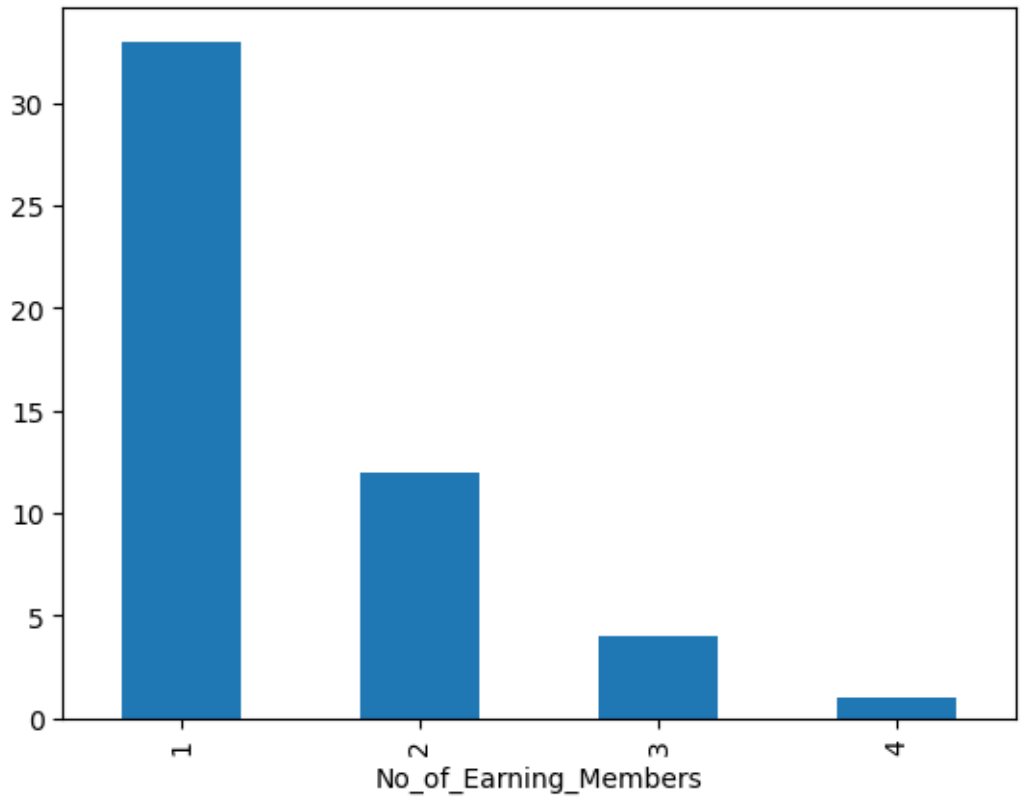
[41]:
```
Highest_Qualified_Member  Graduate  Under-Graduate  Professional  \
count                           19              10            10

Highest_Qualified_Member  Post-Graduate  Illiterate
count                             6           5
```

12 12.Plot the Histogram to count the No_of_Earning_Members

[54]:
```
df['No_of_Earning_Members'].value_counts().plot(kind='bar')
```

[54]:
```
<Axes: xlabel='No_of_Earning_Members'>
```



13 13.Suppose you have option to invest in Stock A or Stock B. The stocks • have different expected returns and standard deviations. The expected return of Stock A is 15% and Stock B is 10%. Standard Deviation of the returns of these stocks is 10% and 5% respectively.

13.1 Which is better investment?

[60]:
```
#Here we need to calculate the coeff of variation

# Coeff_of_var = std deviation / mean

Coeff_of_var_StockA=10/15
print(Coeff_of_var_StockA)
```

```
Coeff_of_var_StockB=5/10
print(Coeff_of_var_StockB)
```

```
0.6666666666666666
0.5
```

13.2 Comparison

13.2.1 Stock A has a CV of approximately 66.67%.

13.2.2 Stock B has a CV of 50%.

13.3 Interpretation

13.3.1 A lower CV indicates that the investment has less risk per unit of return.

13.3.2 Stock B has a lower CV (50%) compared to Stock A (66.67%).

13.3.3 Therefore, based on the coefficient of variation, Stock B is the better investment as it offers a lower risk relative to its return compared to Stock A.

13.4 –> STOCK B