Chapter 5

April 21, 2025

1 Hands-On Data Preprocessing in Python

Learn how to effectively prepare data for successful data analytics

AUTHOR: Dr. Roy Jafari

2 Chapter 5: Data Visualization

2.1 Summarizing a population

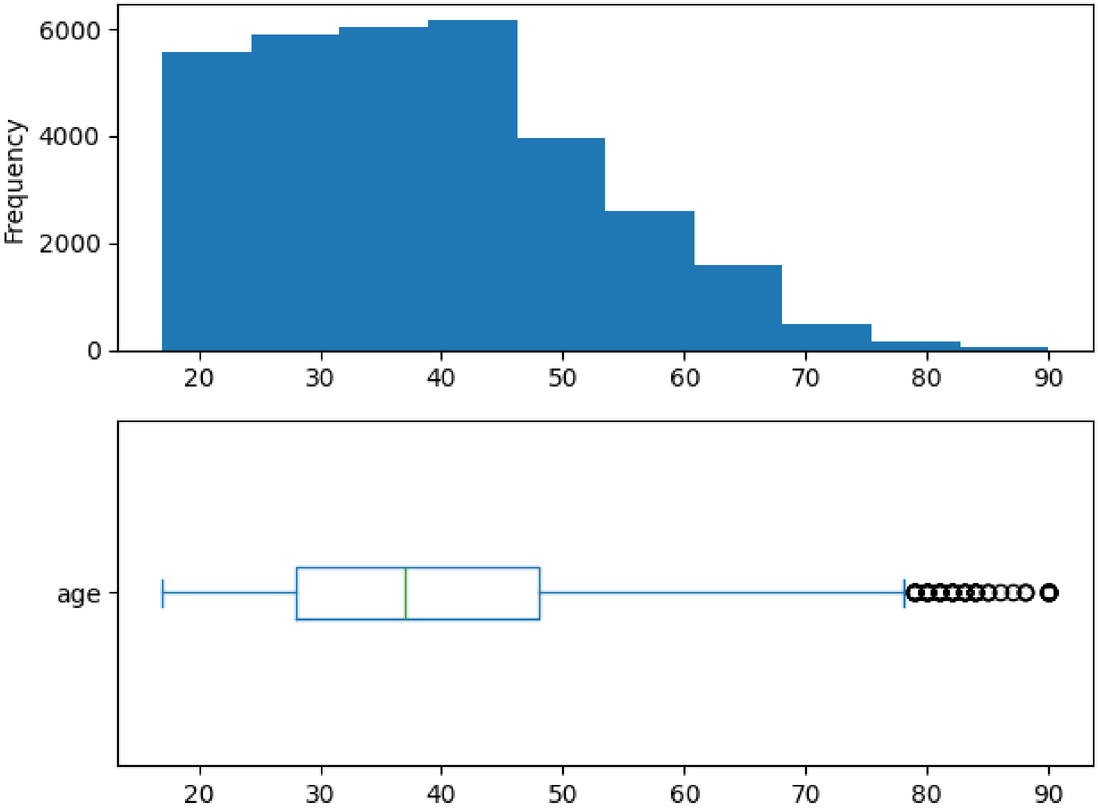
2.1.1 Example of summarizing numerical attributes

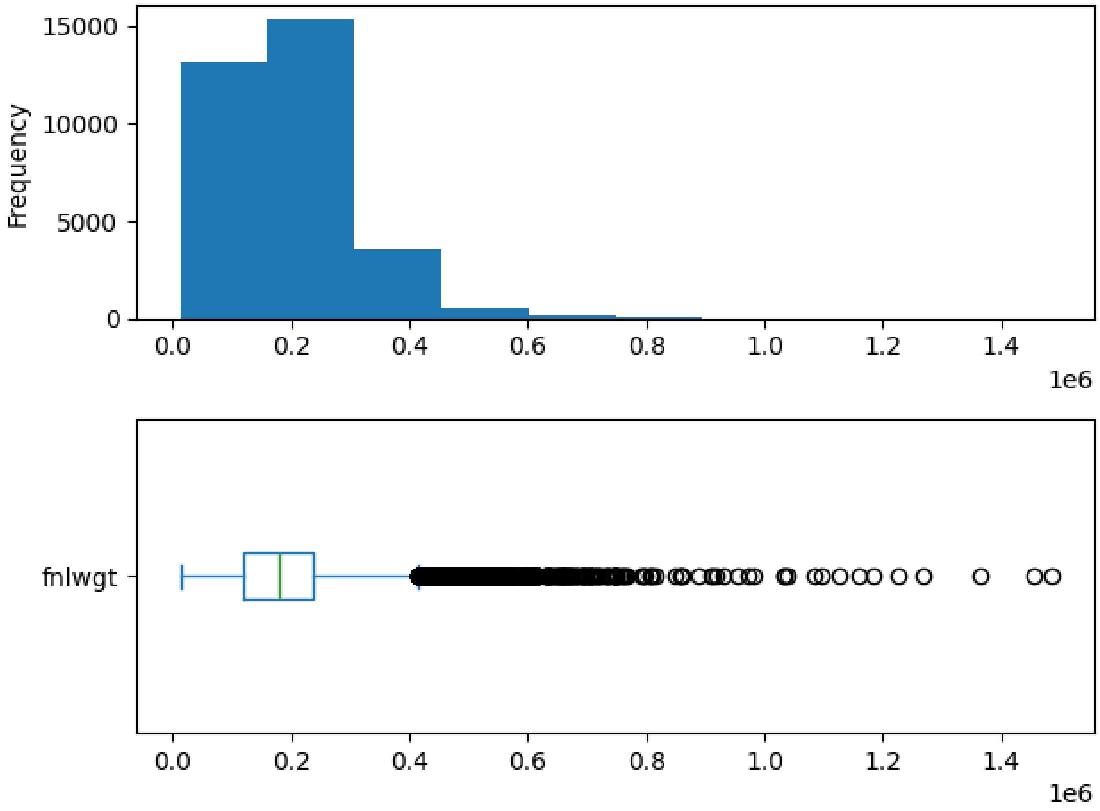
[4]:

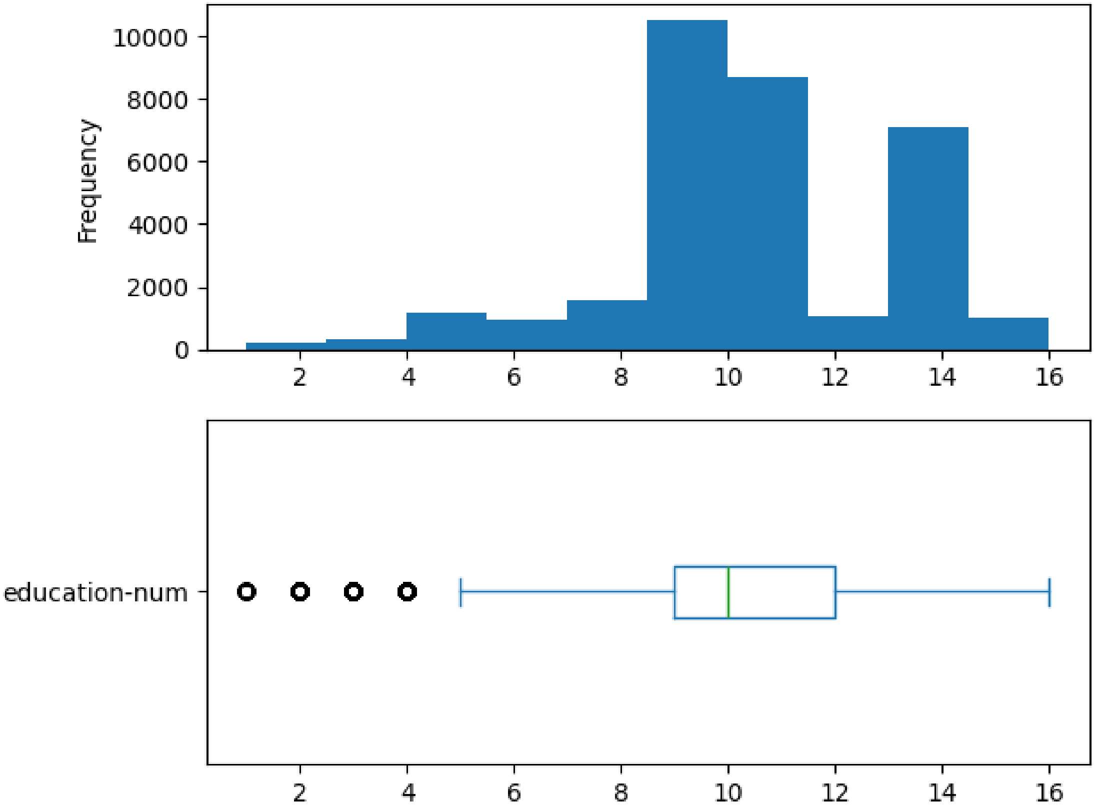
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

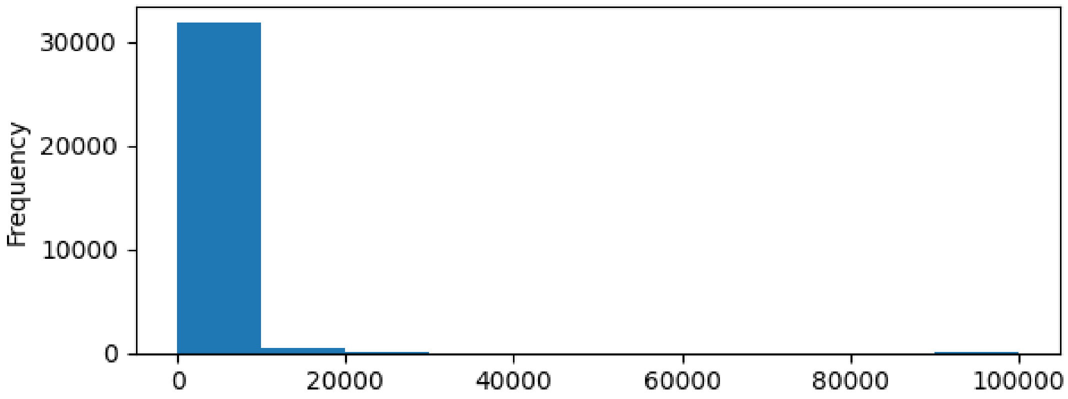
[5]:

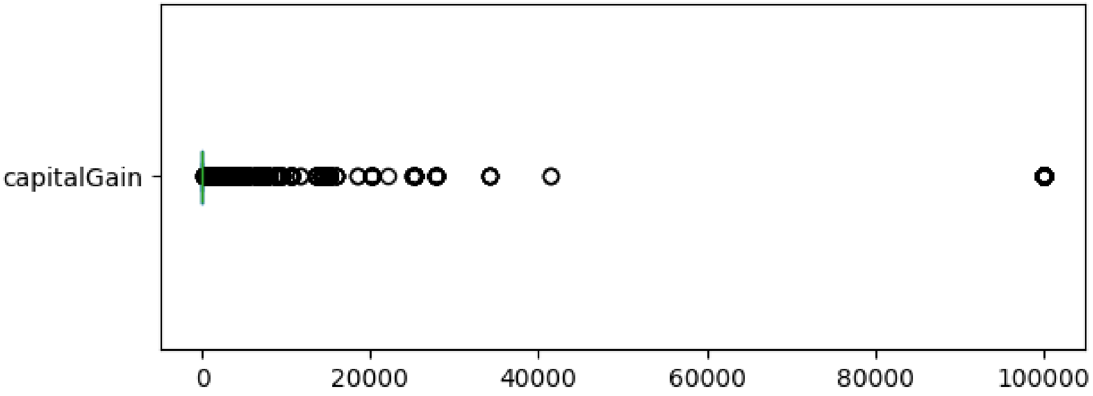
```python
adult_df = pd.read_csv('adult.csv')

numerical_attributes = ['age', 'fnlwgt', 'education-num', 'capitalGain',
                        'capitalLoss', 'hoursPerWeek']

for att in numerical_attributes:
    plt.subplot(2,1,1)
    adult_df[att].plot.hist()
    plt.subplot(2,1,2)
    adult_df[att].plot.box(vert=False)
    plt.tight_layout()
    plt.savefig('{}.png'.format(att), dpi=600)
    plt.show()
```

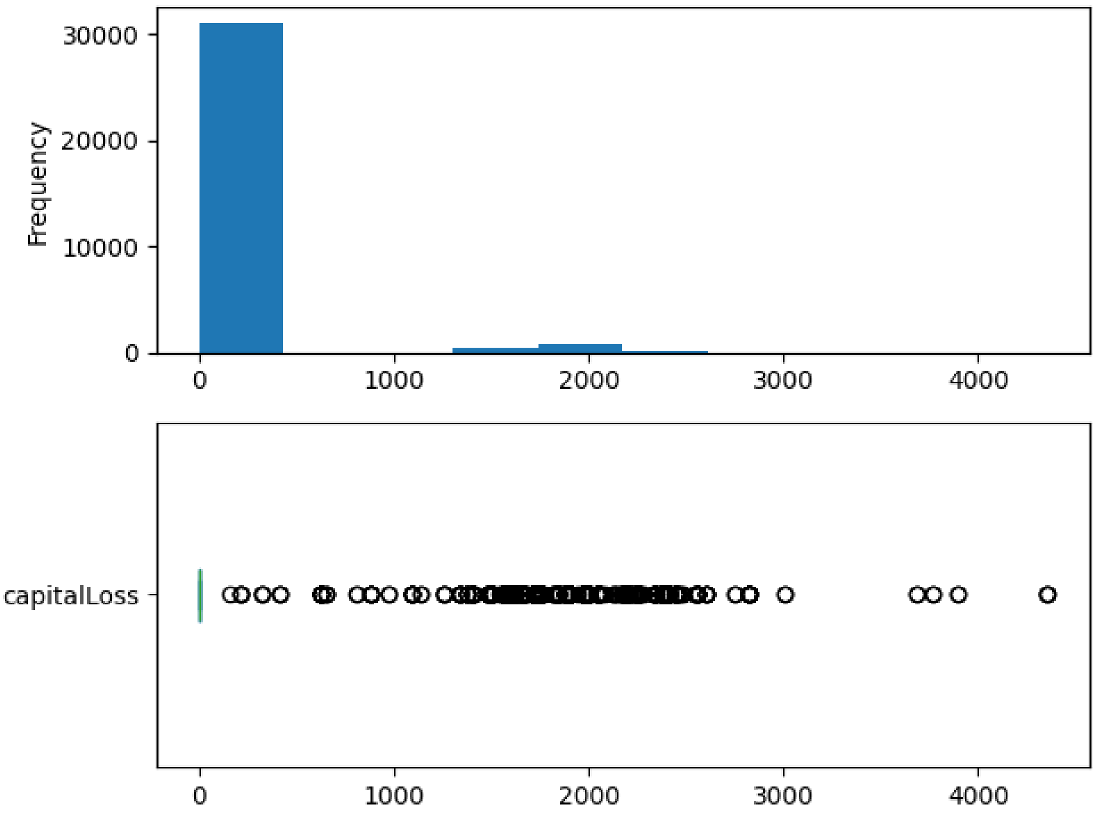

6000
4000
2000
0
Frequency
20
30
40
50
60
70
80
90
age
20
30
40
50
60
70
80
90

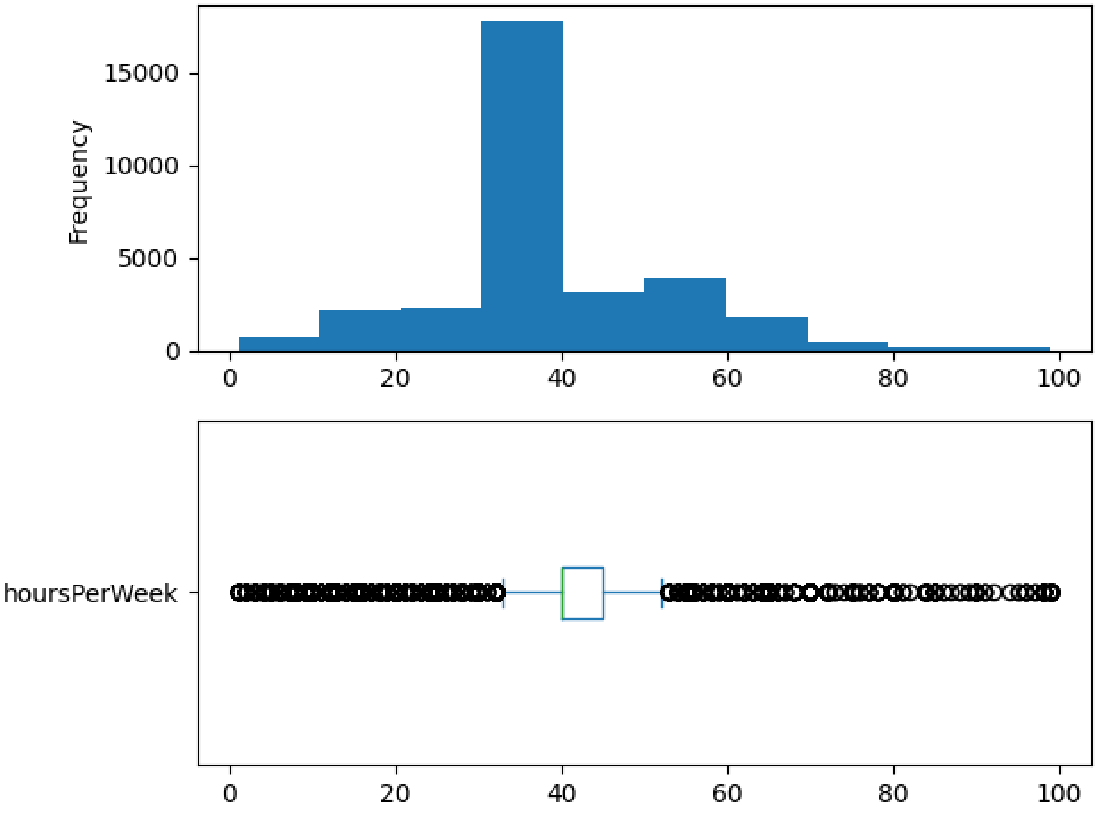

Frequency
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6
fnlwgt
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6


Frequency
10000
8000
6000
4000
2000
0
2
4
6
8
10
12
14
16
education-num
2
4
6
8
10
12
14
16


Frequency
30000
20000
10000
0
0
20000
40000
60000
80000
100000

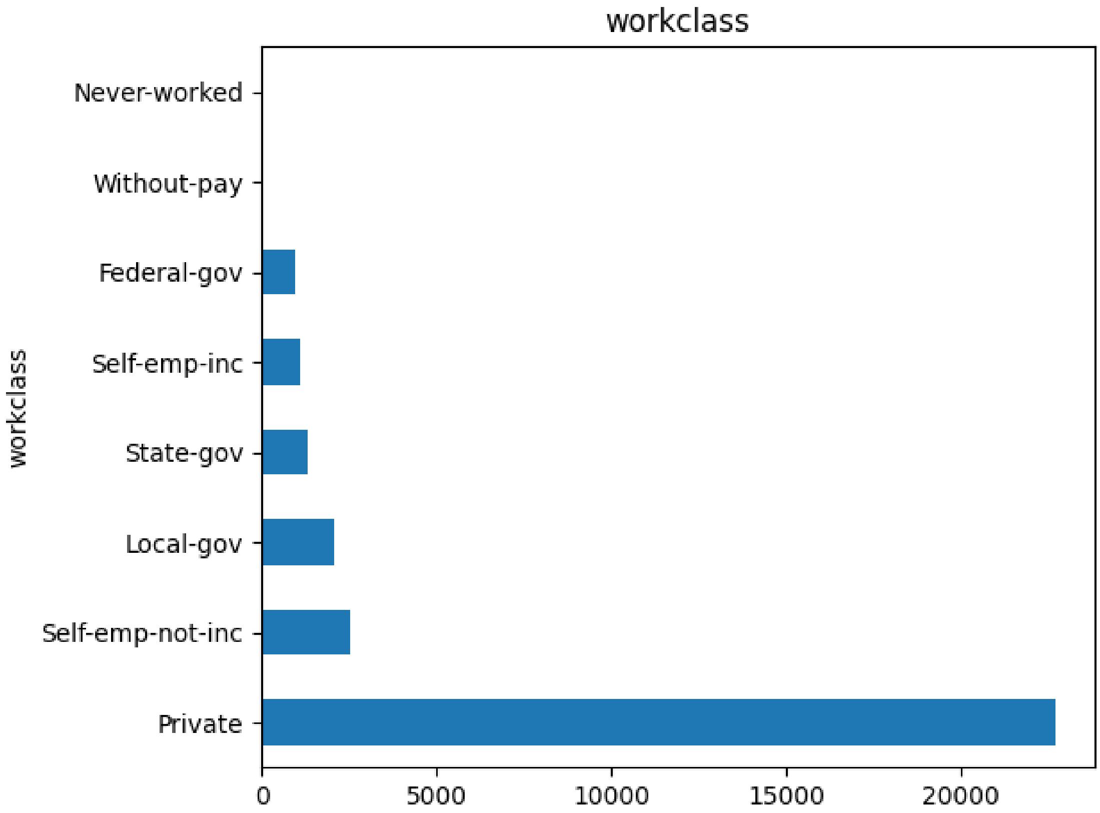

capitalGain
0
20000
40000
60000
80000
100000

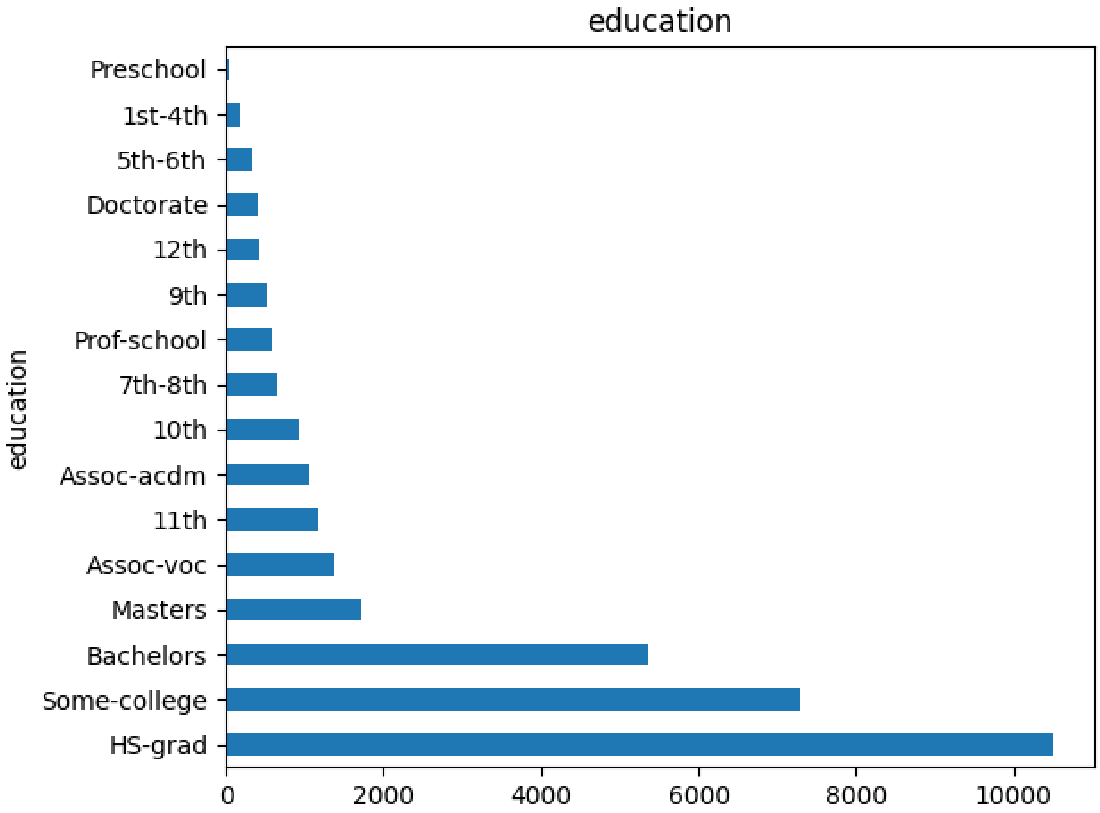

30000
20000
10000
0
Frequency
0
1000
2000
3000
4000
capitalLoss
0
1000
2000
3000
4000

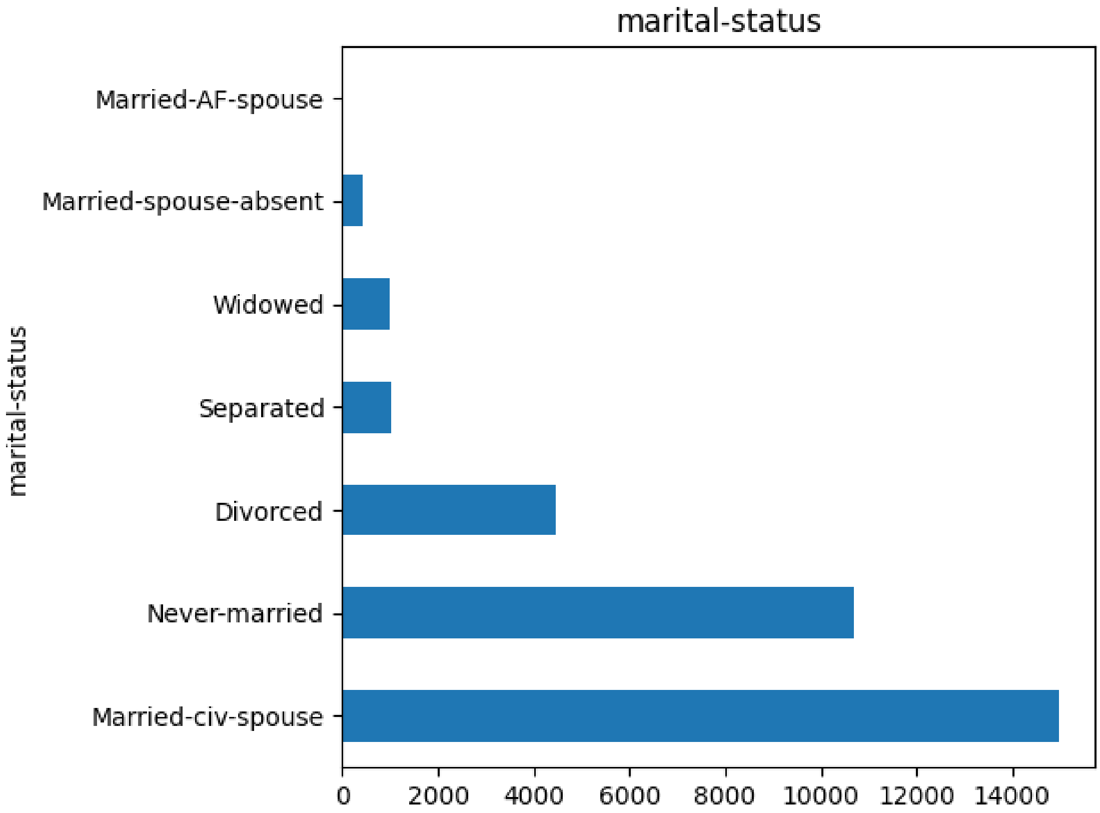


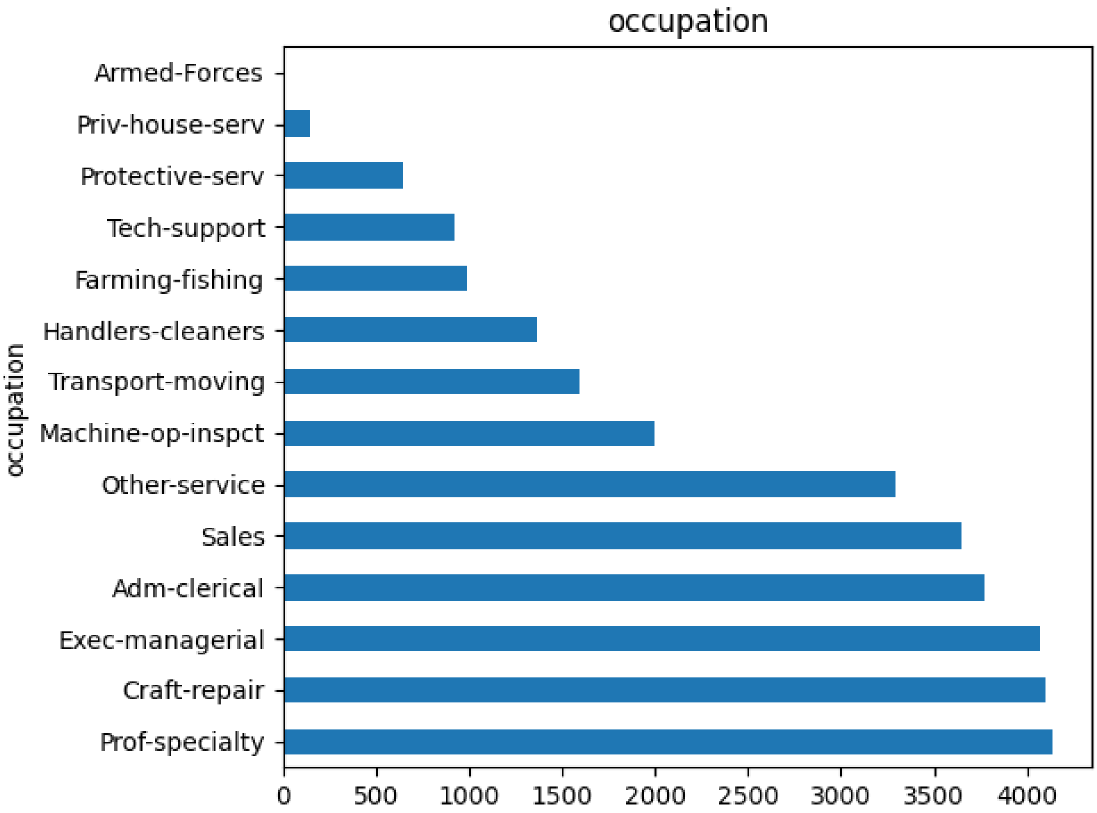
2.1.2 Example of summarizing categorical attributes

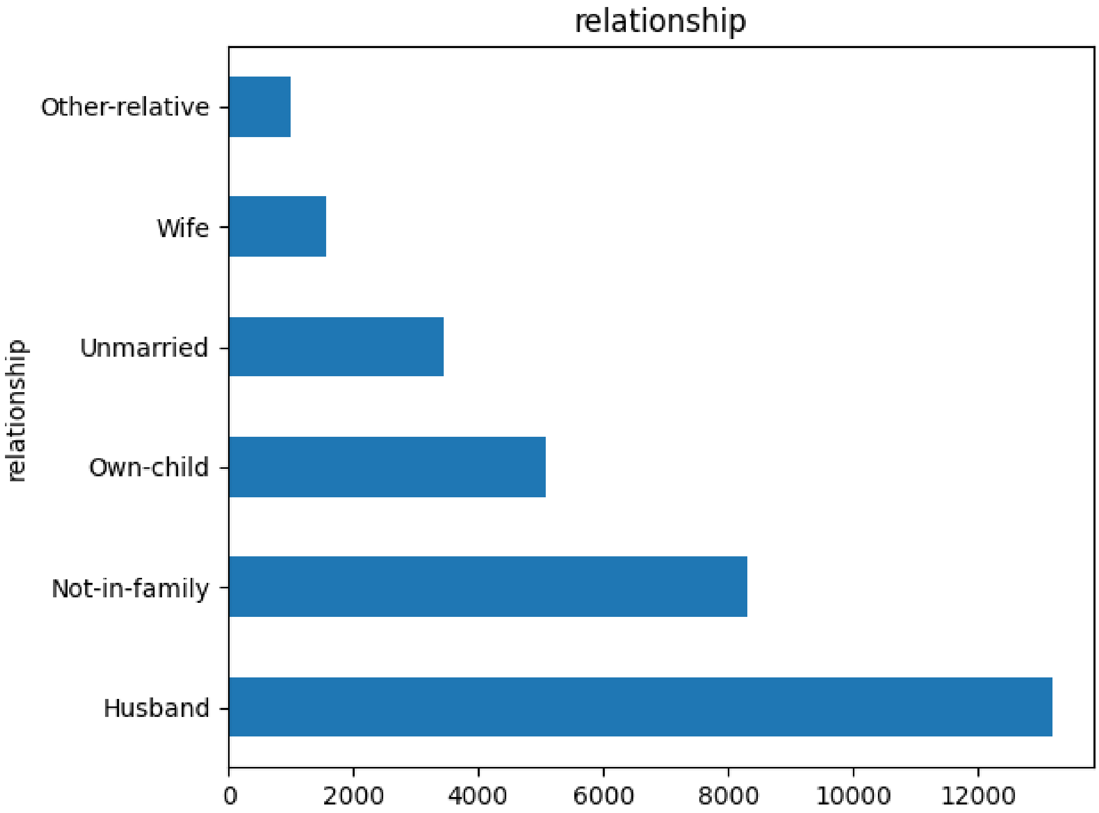
[6]:
```
categorical_attributes = ['workclass', 'education', 'marital-status',
                          'occupation', 'relationship', 'race',
                          'sex','nativeCountry','income']

for att in categorical_attributes:
    adult_df[att].value_counts().plot.barh()
    plt.title(att)
    plt.tight_layout()
    plt.savefig('{}.png'.format(att), dpi=600)
    plt.show()
```

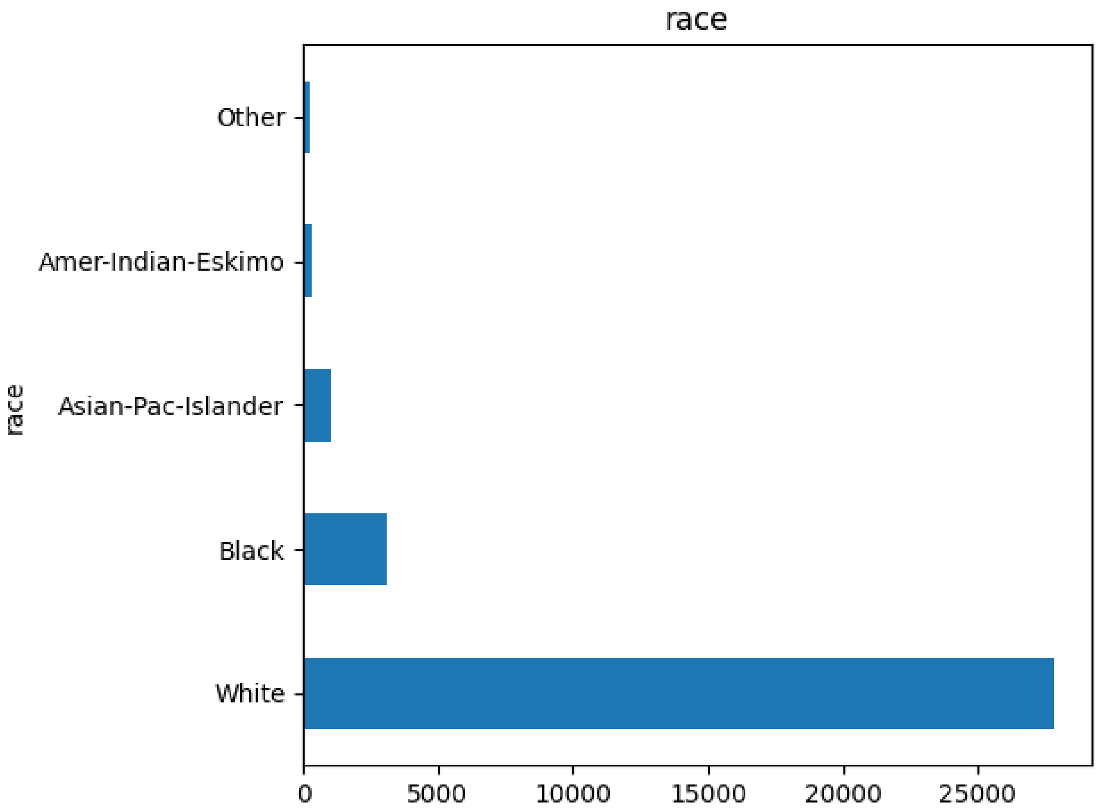

workclass
workclass
Never-worked
Without-pay
Federal-gov
Self-emp-inc
State-gov
Local-gov
Self-emp-not-inc
Private
0
5000
10000
15000
20000

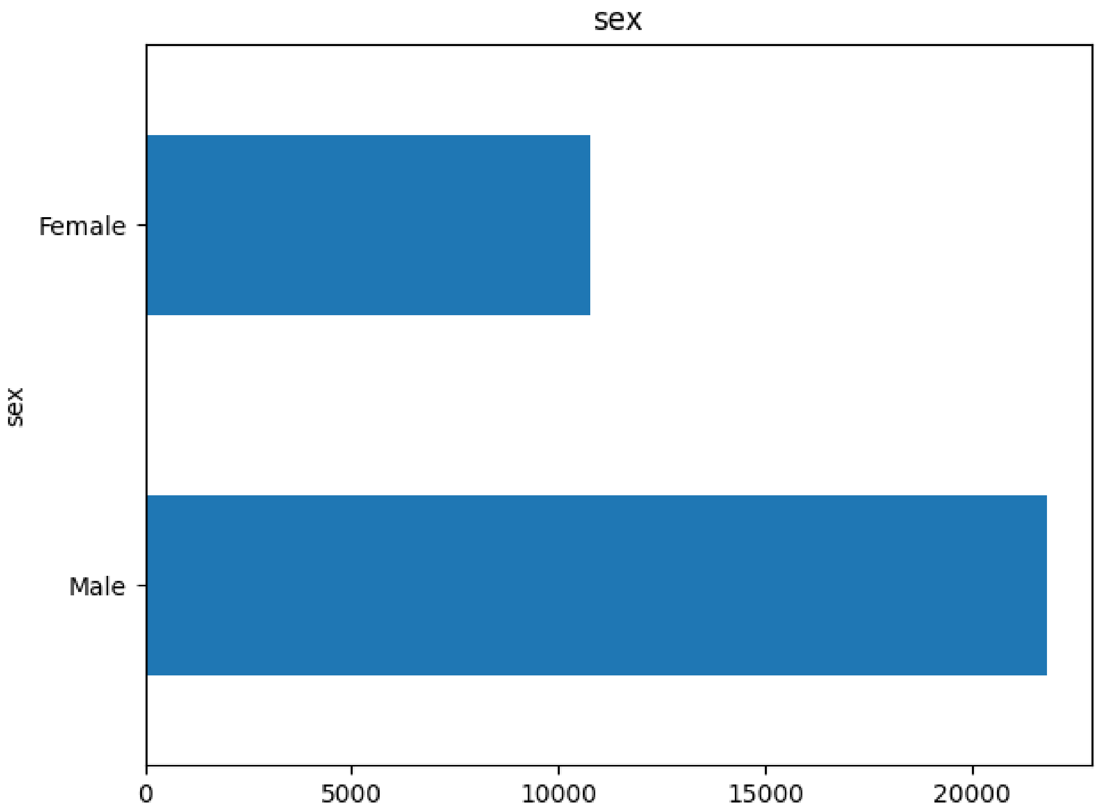

education
education
Preschool
1st-4th
5th-6th
Doctorate
12th
9th
Prof-school
7th-8th
10th
Assoc-acdm
11th
Assoc-voc
Masters
Bachelors
Some-college
HS-grad
0
2000
4000
6000
8000
10000

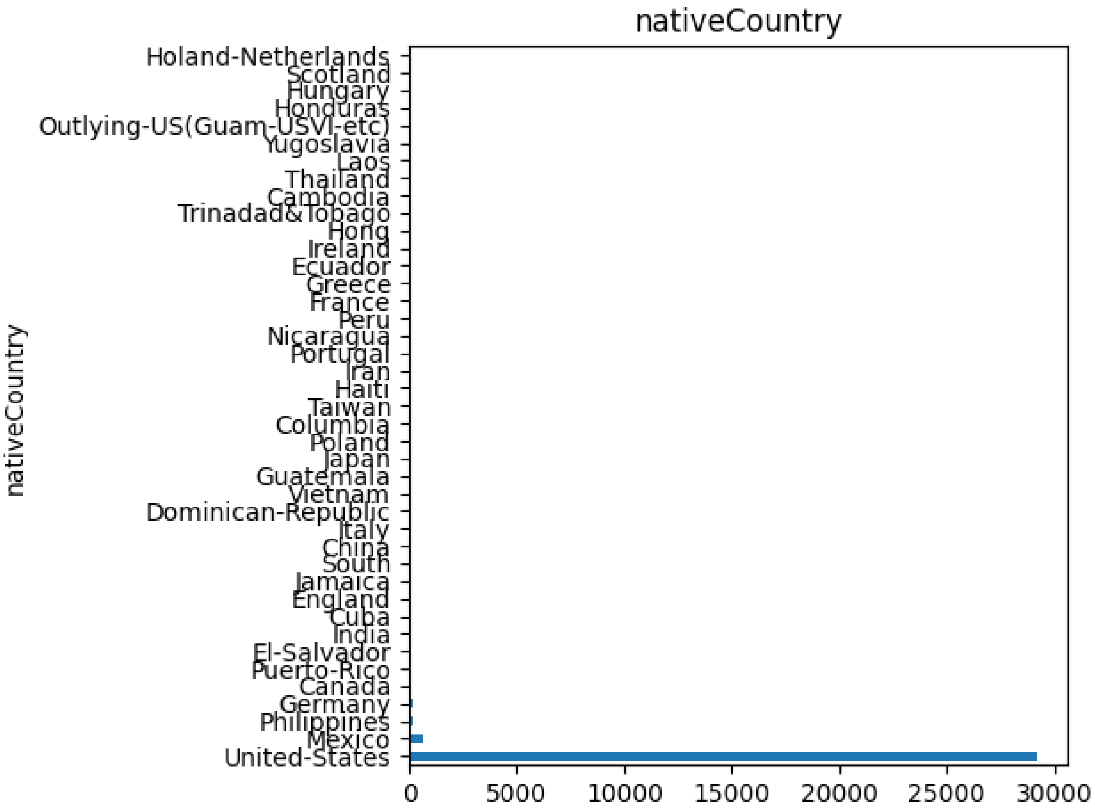

marital-status
marital-status
Married-AF-spouse
Married-spouse-absent
Widowed
Separated
Divorced
Never-married
Married-civ-spouse
0
2000
4000
6000
8000
10000
12000
14000

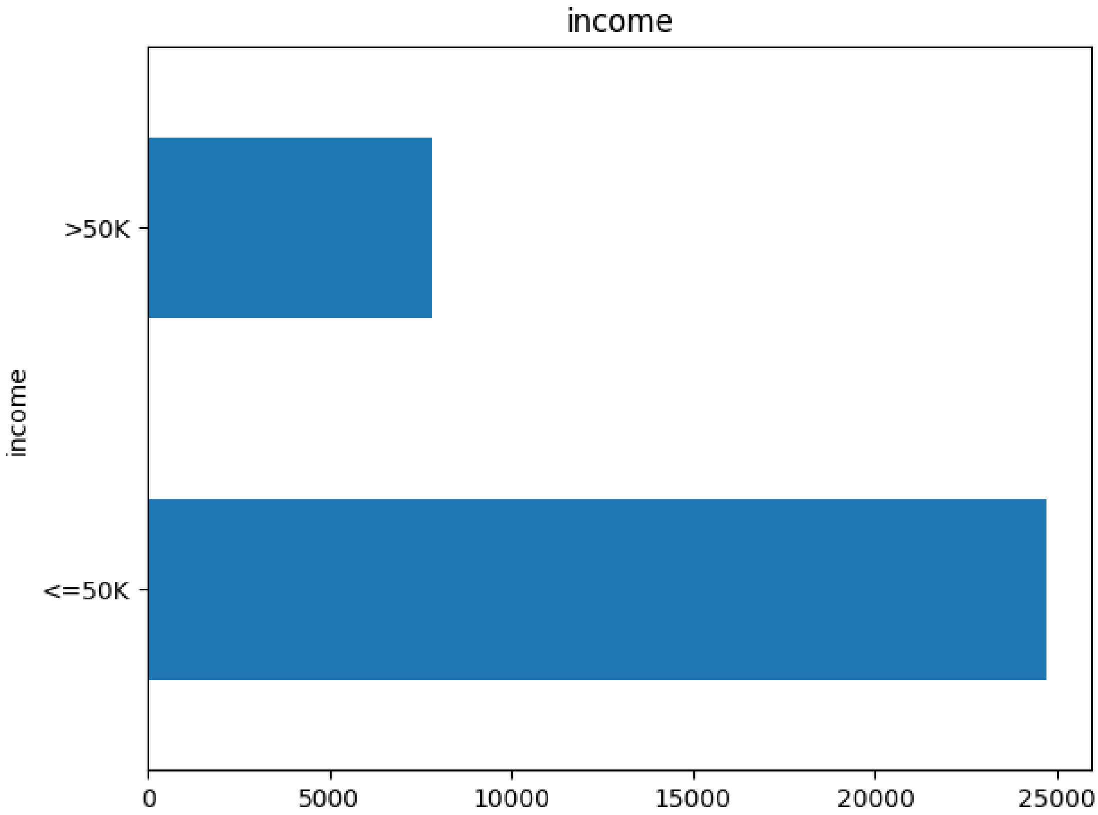

occupation
occupation
Armed-Forces
Priv-house-serv
Protective-serv
Tech-support
Farming-fishing
Handlers-cleaners
Transport-moving
Machine-op-inspct
Other-service
Sales
Adm-clerical
Exec-managerial
Craft-repair
Prof-specialty
0
500
1000
1500
2000
2500
3000
3500
4000


relationship
relationship
Other-relative
Wife
Unmarried
Own-child
Not-in-family
Husband
0
2000
4000
6000
8000
10000
12000


race
race
Other
Amer-Indian-Eskimo
Asian-Pac-Islander
Black
White
0
5000
10000
15000
20000
25000


sex
Female
Male
sex
0
5000
10000
15000
20000


nativeCountry
nativeCountry
Holand-Netherlands
Scotland
Hungary
Honduras
Outlying-US(Guam-USVI-etc)
Yugoslavia
Laos
Thailand
Cambodia
Trinadad&Tobago
Hong
Ireland
Ecuador
Greece
France
Peru
Nicaragua
Portugal
Iran
Haiti
Taiwan
Columbia
Poland
Japan
Guatemala
Vietnam
Dominican-Republic
Italy
China
South
Jamaica
England
Cuba
India
El-Salvador
Puerto-Rico
Canada
Germany
Philippines
Mexico
United-States
0
5000
10000
15000
20000
25000
30000

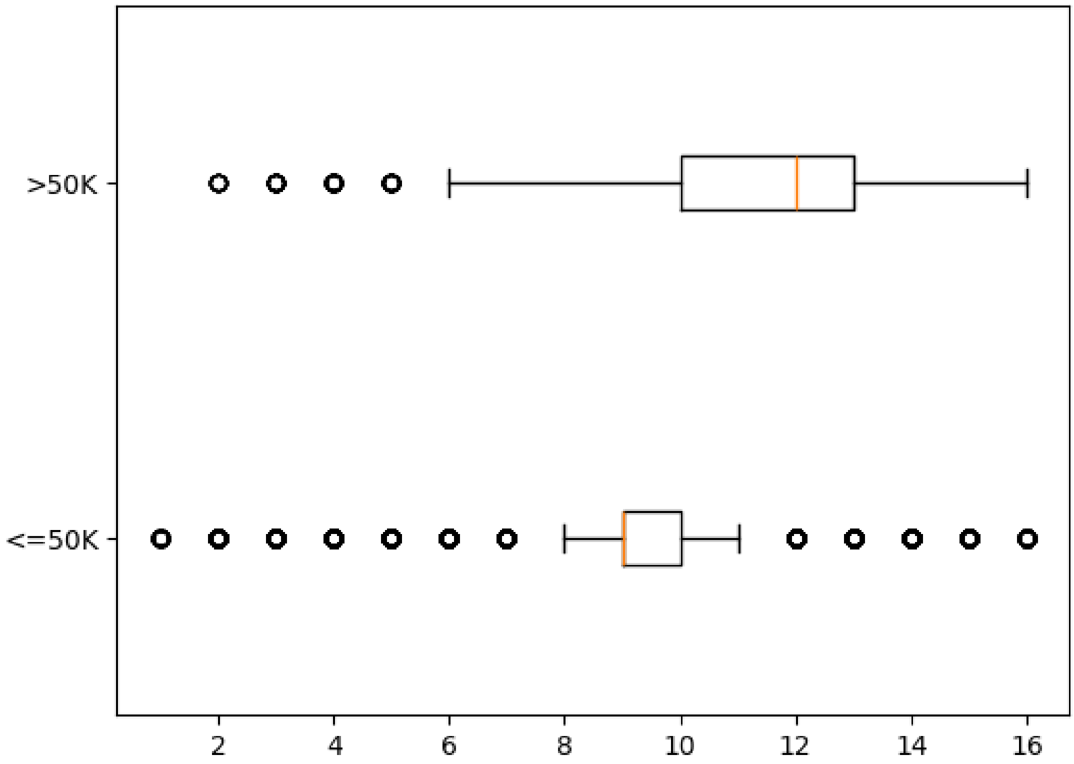


2.2 Comparing populations

2.2.1 Example of comparing populations using boxplots

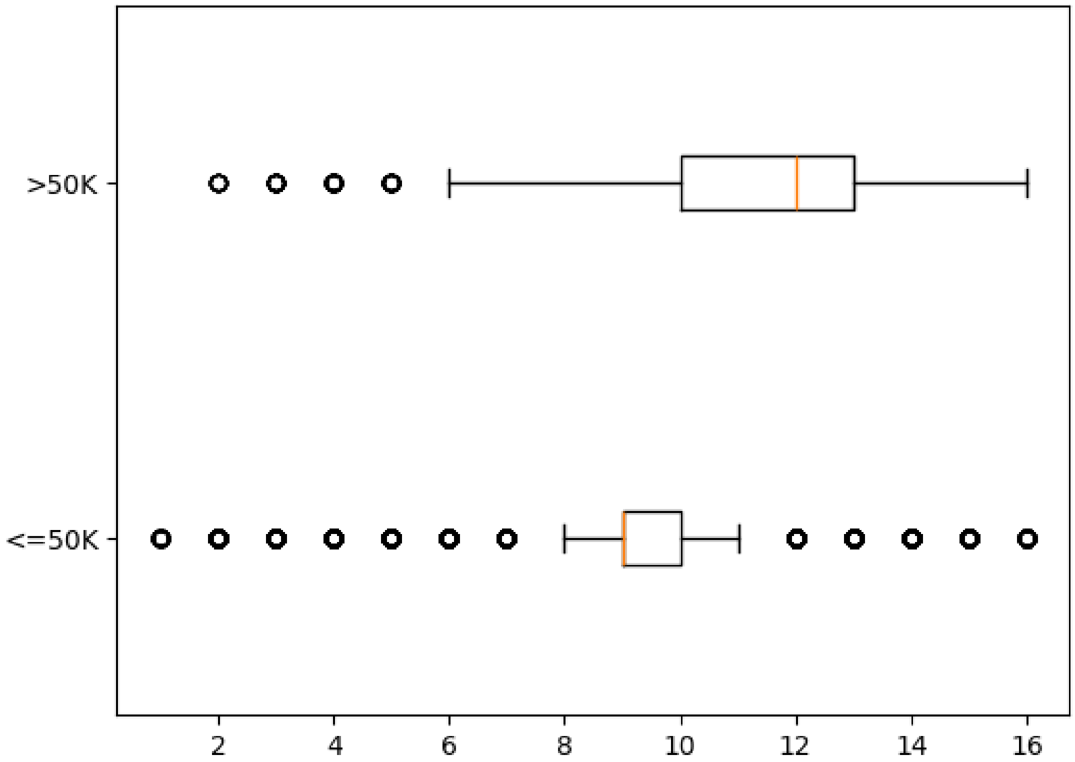
[7]:

```
income_possibilities = adult_df.income.unique()

box_sr = pd.Series('',index = income_possibilities)

for poss in income_possibilities:
    BM = adult_df.income == poss
    box_sr[poss] = adult_df[BM]['education-num']

plt.boxplot(box_sr,vert=False)
plt.yticks([1,2],income_possibilities)
plt.show()
```



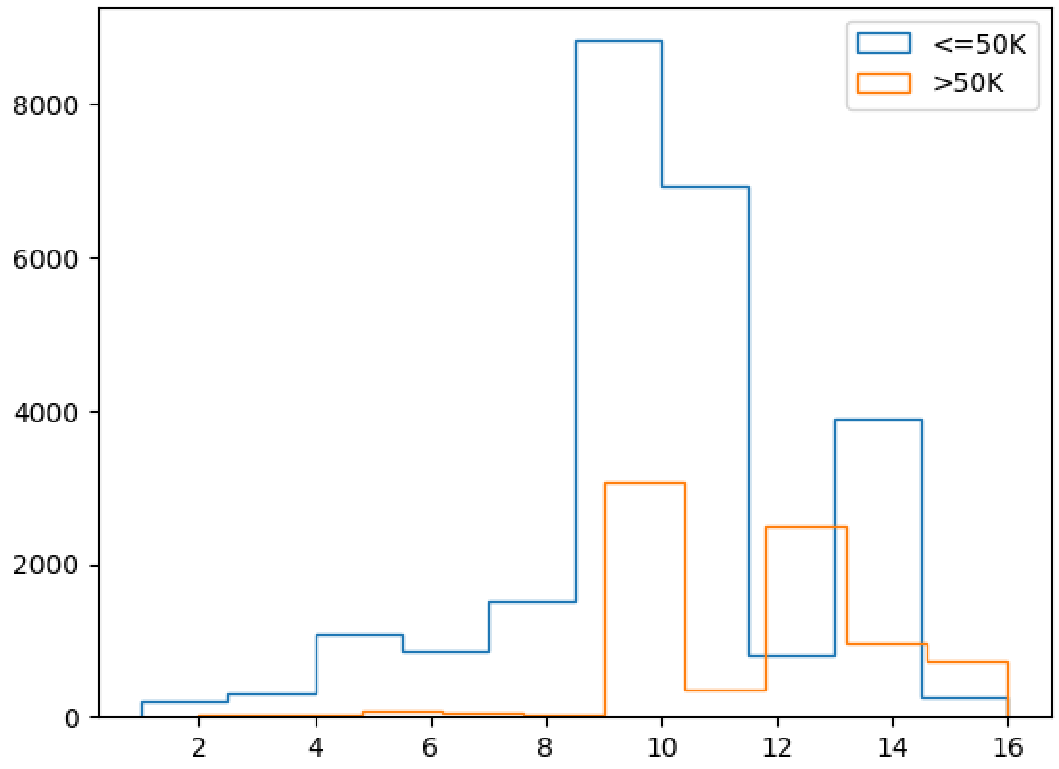
[8]:

```
income_possibilities = adult_df.income.unique()

dataForBox_dic= {}

for poss in income_possibilities:
    BM = adult_df.income == poss
    dataForBox_dic[poss] = adult_df[BM]['education-num']

plt.boxplot(dataForBox_dic.values(),vert=False)
plt.yticks([1,2],income_possibilities)
plt.show()
```

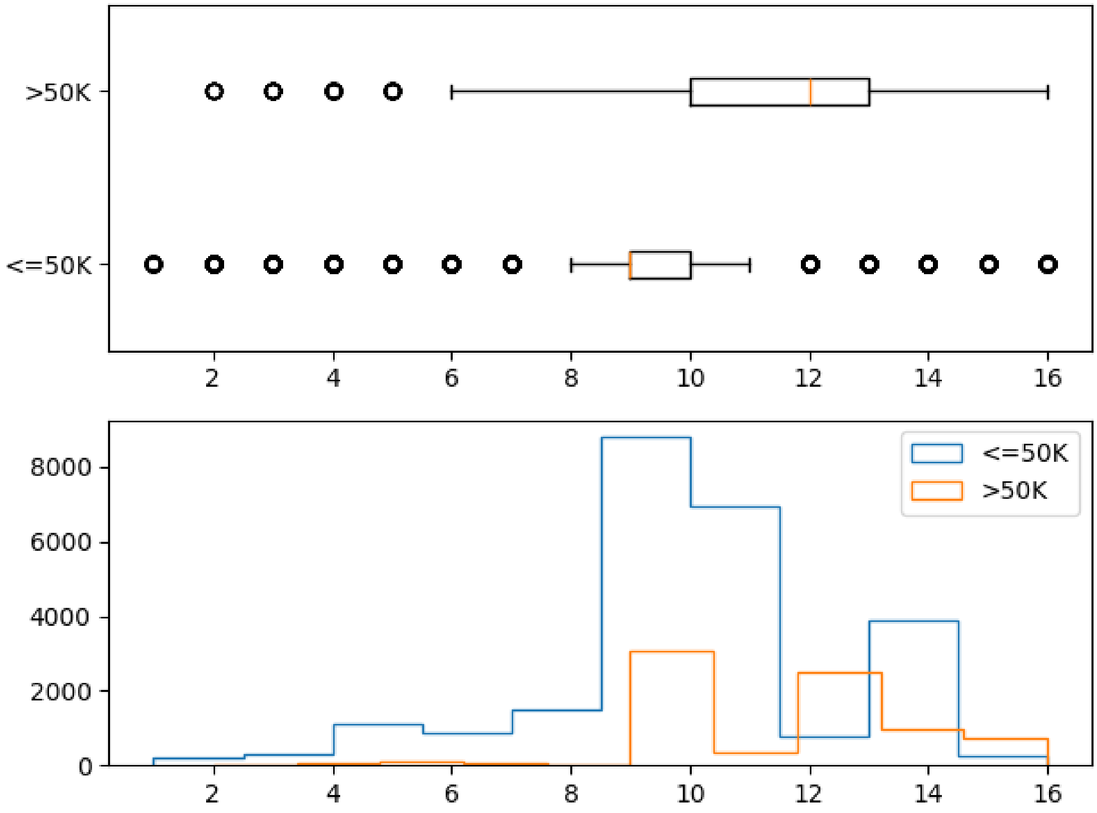


2.2.2 Example of comparing populations using histograms

[9]:
```
income_possibilities = adult_df.income.unique()

for poss in income_possibilities:
    BM = adult_df.income == poss
    plt.hist(adult_df[BM]['education-num'],
                histtype='step', label=poss)
plt.legend()
plt.show()
```



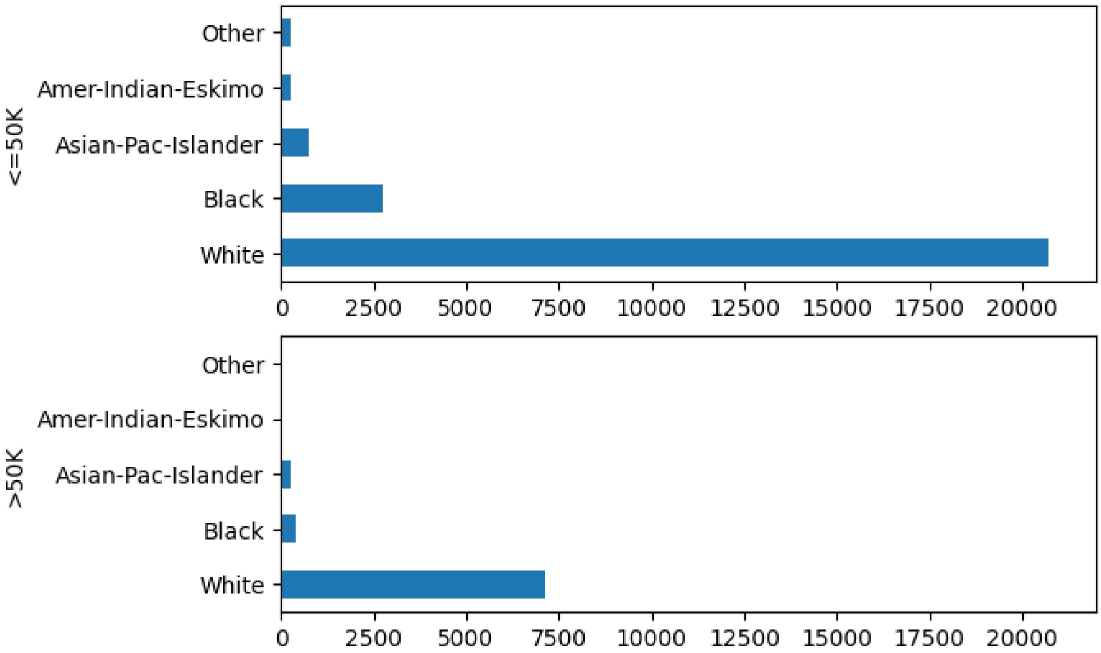
[10]:
```
income_possibilities = adult_df.income.unique()

dataForBox_dic= {}

for poss in income_possibilities:
    BM = adult_df.income == poss
    dataForBox_dic[poss] = adult_df[BM]['education-num']

plt.subplot(2,1,1)
plt.boxplot(dataForBox_dic.values(),vert=False)
plt.yticks([1,2],income_possibilities)

plt.subplot(2,1,2)
for poss in income_possibilities:
    BM = adult_df.income == poss
    plt.hist(adult_df[BM]['education-num'],
            histtype='step',label=poss)
plt.legend()
plt.tight_layout()
plt.show()
```



2.2.3 Example of comparing populations using bar charts

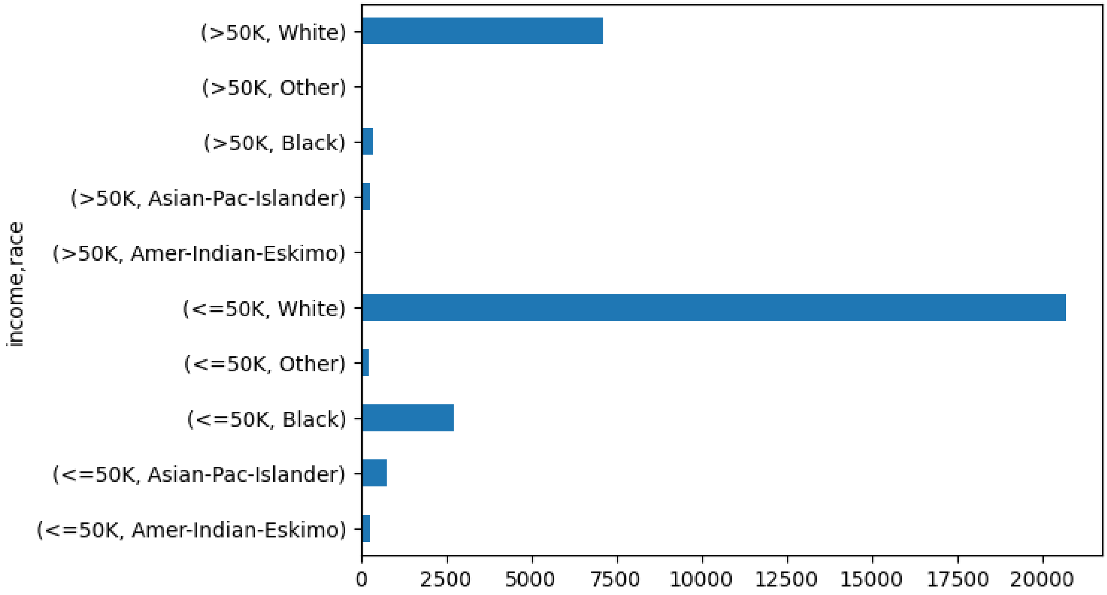
The first way of solving

[11]:
```
income_possibilities = adult_df.income.unique()

for i,poss in enumerate(income_possibilities):
    plt.subplot(2,1,i+1)
    BM = adult_df.income == poss
    adult_df[BM].race.value_counts().plot.barh()
    plt.xlim([0,22000])
    plt.ylabel(poss)
```



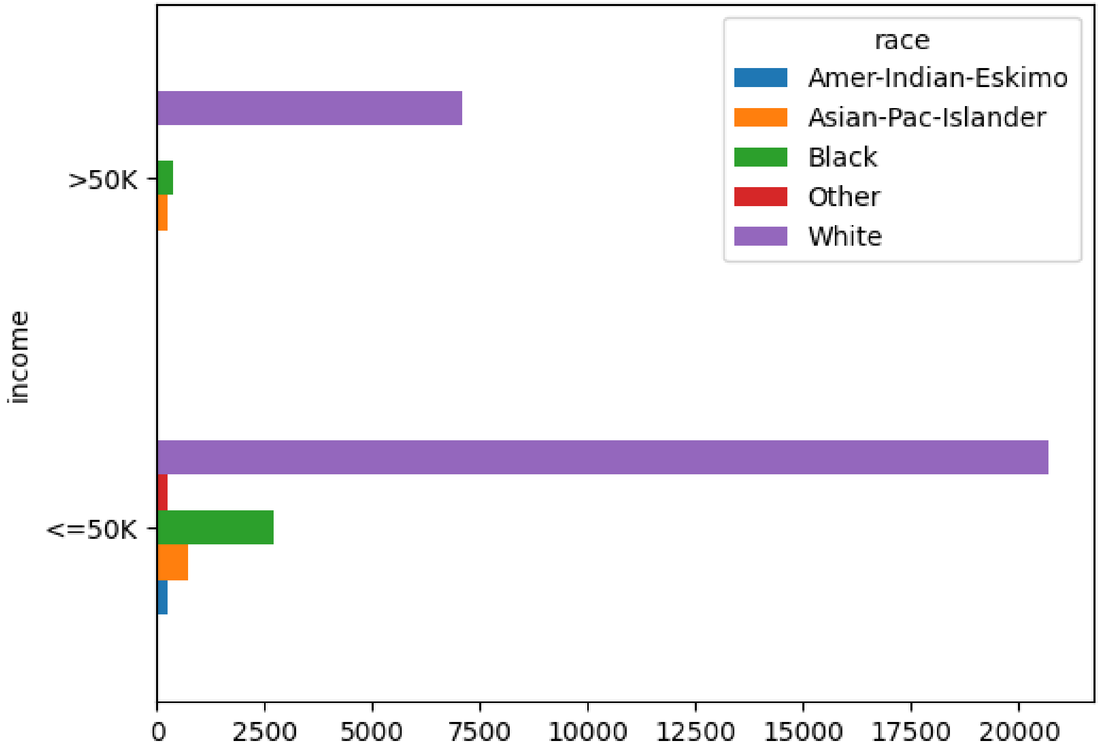
The second way of solving

```
[12]: adult_df.groupby(['income','race']).size().plot.barh()
      plt.show()
```



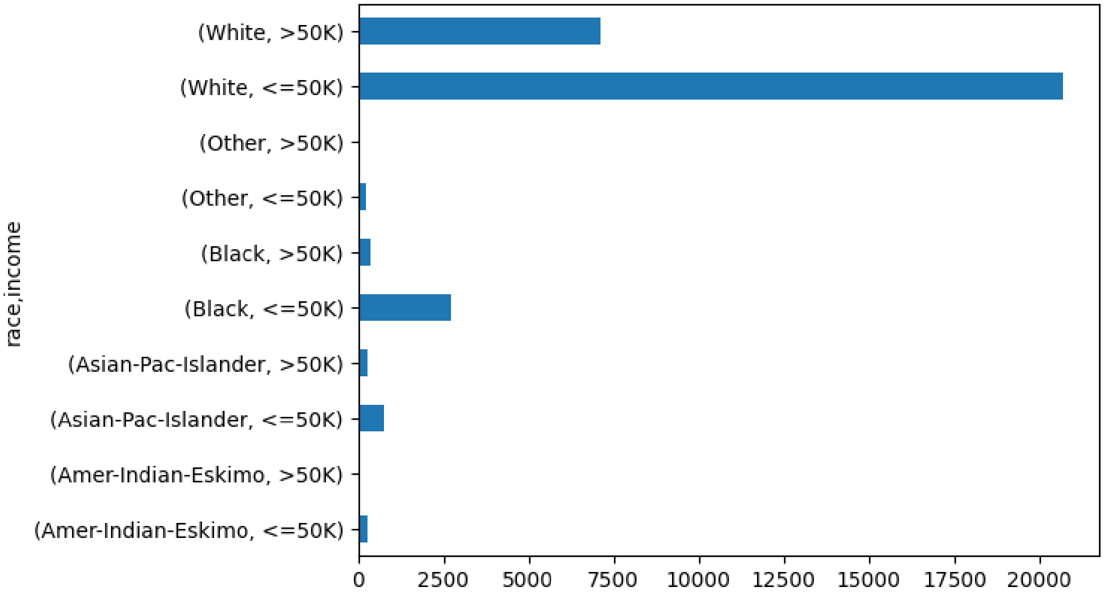
The third way of solving

[13]:
```
adult_df.groupby(['income','race']).size().unstack().plot.barh()
plt.show()
```



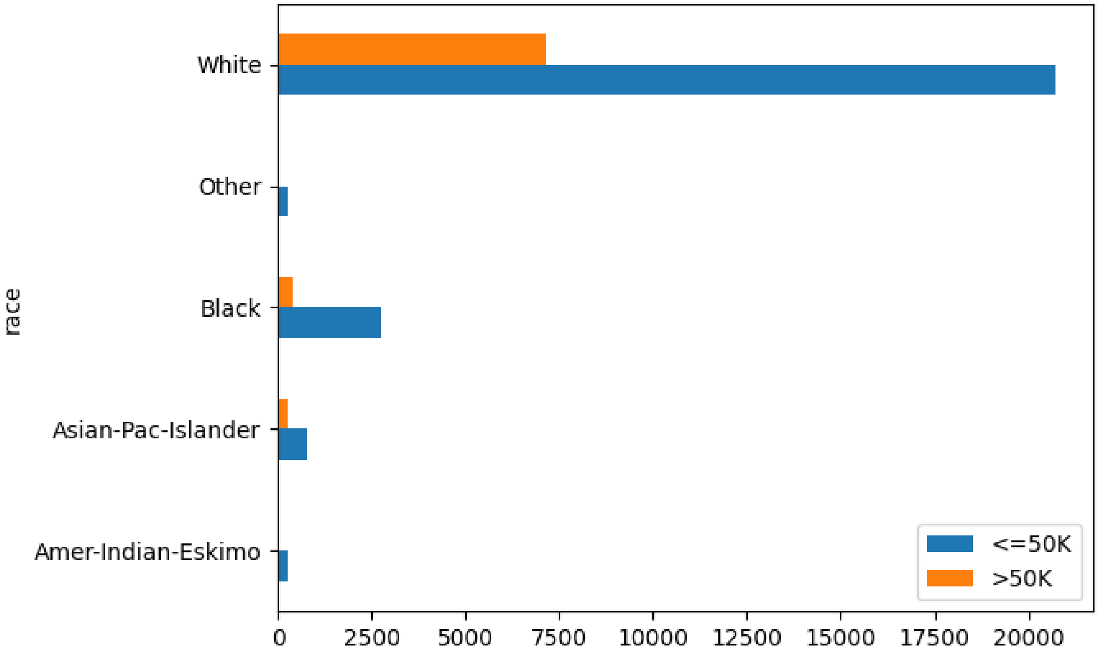
The fourth way of solving

[14]:
```
adult_df.groupby(['race','income']).size().plot.barh()
plt.show()
```



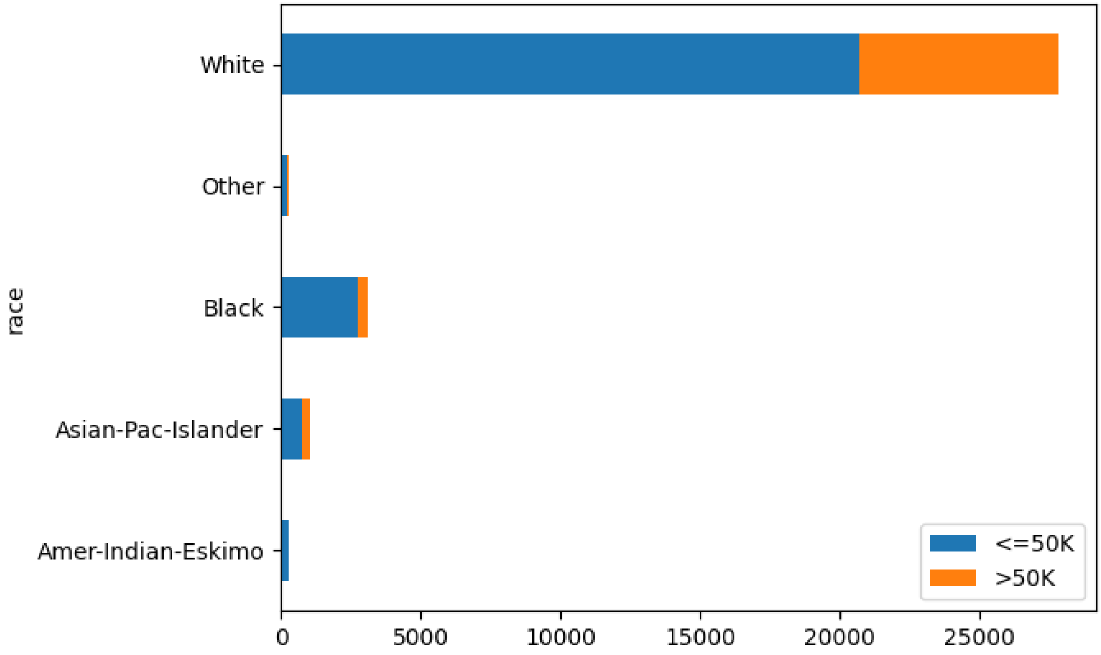
The fifth way of solving

[15]:
```
adult_df.groupby(['race','income']).size().unstack().plot.barh()
plt.legend(loc=4)
plt.show()
```



The sixth way of solving

```
[16]: adult_df.groupby(['race','income']).size().unstack().plot.barh(stacked=True)
plt.legend(loc=4)
plt.show()
```

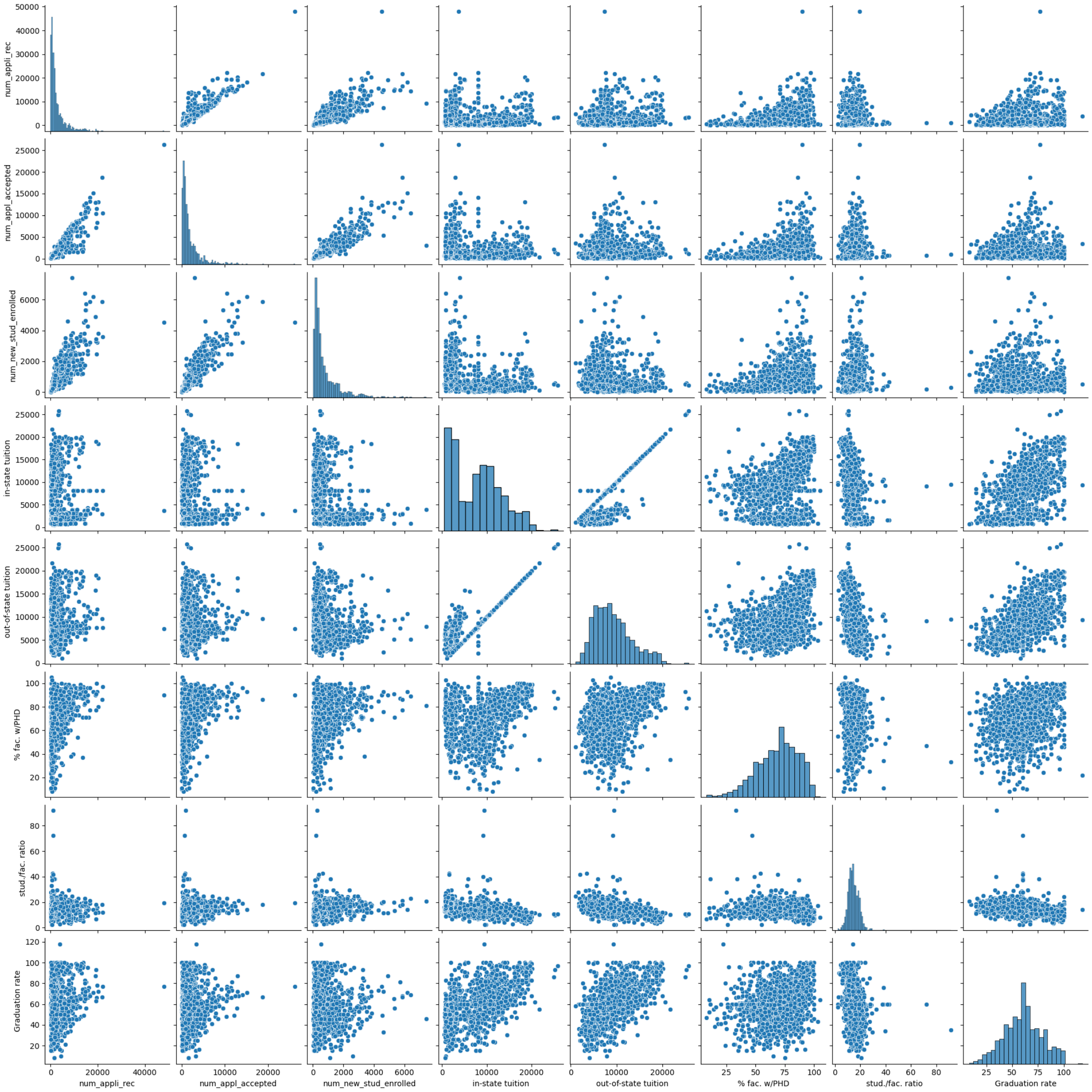


2.3 Investigating the relationship between two attributes

2.3.1 Visualizing the relationship between two numerical attributes

Example of using scatterplots to investigate between the numerical attributes

```
[17]: import seaborn as sns
uni_df = pd.read_csv('Universities_imputed_reduced.csv')
sns.pairplot(uni_df)
plt.show()
```



2.3.2 Visualizing the relationship between two categorical attributes

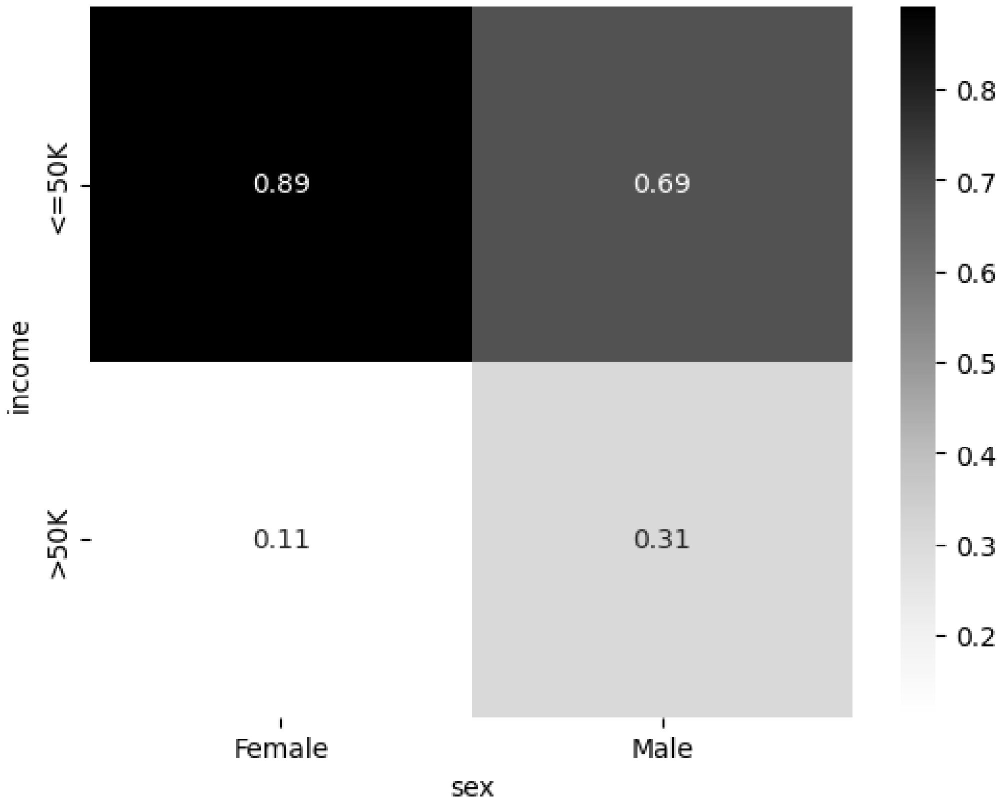
Example of using contingency table to examine the relationship between two categorical (binary) attributes

[18]:
```
contingency_tbl = pd.crosstab(adult_df.income,adult_df.sex)
contingency_tbl
```

[18]:
```
sex     Female   Male
income
<=50K     9592  15128
>50K      1179   6662
```

[19]:

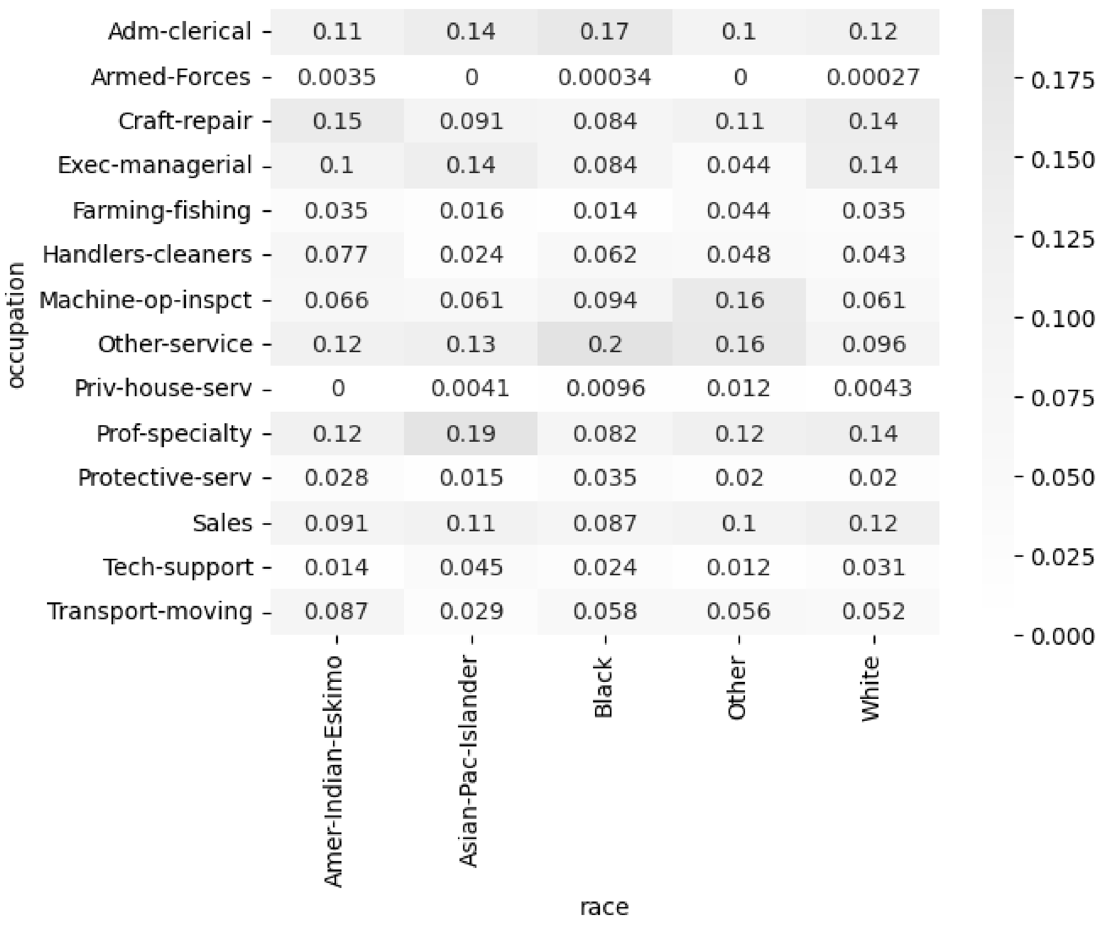
```
probablity_tbl = contingency_tbl/ contingency_tbl.sum()
sns.heatmap(probablity_tbl, annot=True, center=0.5 ,cmap="Greys")
plt.show()
```



Example of using contingency table to relationship between two categorical (non-binary) attributes

[20]:

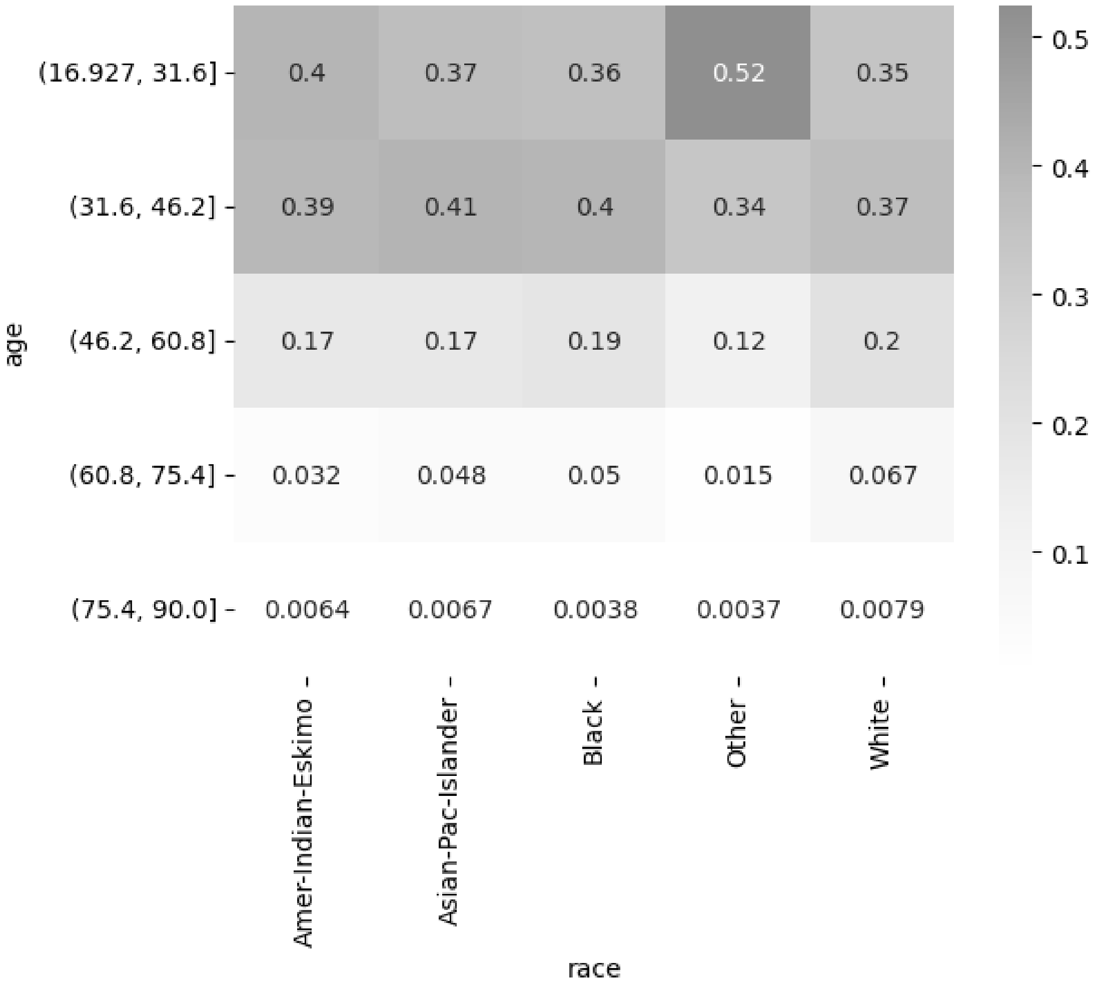
```
contingency_tbl = pd.crosstab(adult_df.occupation,adult_df.race)
probablity_tbl = contingency_tbl/ contingency_tbl.sum()
sns.heatmap(probablity_tbl, annot=True, center=0.5 ,cmap="Greys")
plt.show()
```



2.3.3 Visualizing the relationship between a numerical attribute and a categorical attribute

Visualizing the relationship between a numerical attribute and a categorical attribute

[21]:
```
age_discretized = pd.cut(adult_df.age, bins = 5)
contingency_tbl = pd.crosstab(age_discretized,adult_df.race)
probablity_tbl = contingency_tbl/ contingency_tbl.sum()
sns.heatmap(probablity_tbl, annot=True, center=0.5 ,cmap="Greys")
plt.show()
```



Another example of examining the relationship between a categorical attribute and a numerical attribute

[22]:
```
pd.DataFrame(adult_df.groupby(['education','education-num']).size()).
 ↪drop(columns=[0]).reset_index().sort_values('education-num').
 ↪reset_index(drop=True).transpose()
```

[22]:
```
                      0         1         2         3    4     5     6     7  \
education     Preschool   1st-4th   5th-6th   7th-8th  9th  10th  11th  12th
education-num         1         2         3         4    5     6     7     8

                    8             9         10          11         12  \
education     HS-grad  Some-college  Assoc-voc  Assoc-acdm  Bachelors
education-num       9            10         11          12         13

                    13           14         15
education     Masters  Prof-school  Doctorate
```

```
education-num       14          15          16
```

[23]:

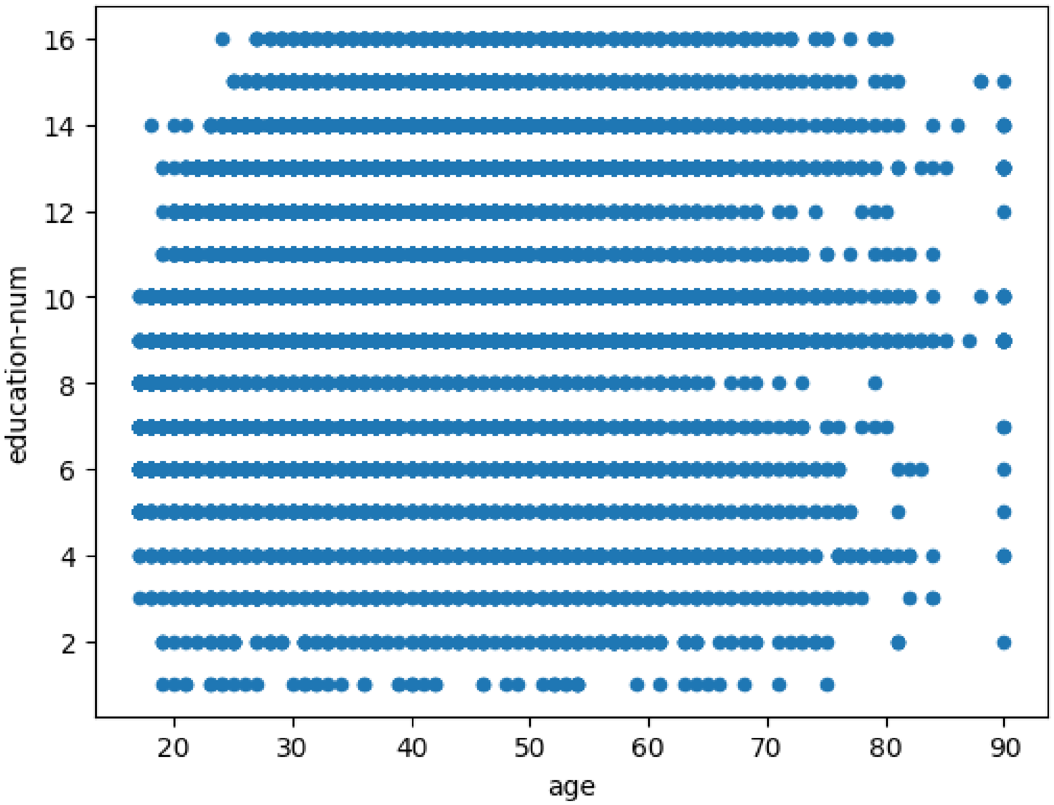
```
adult_df.groupby(['education','education-num']).size()
```

[23]:

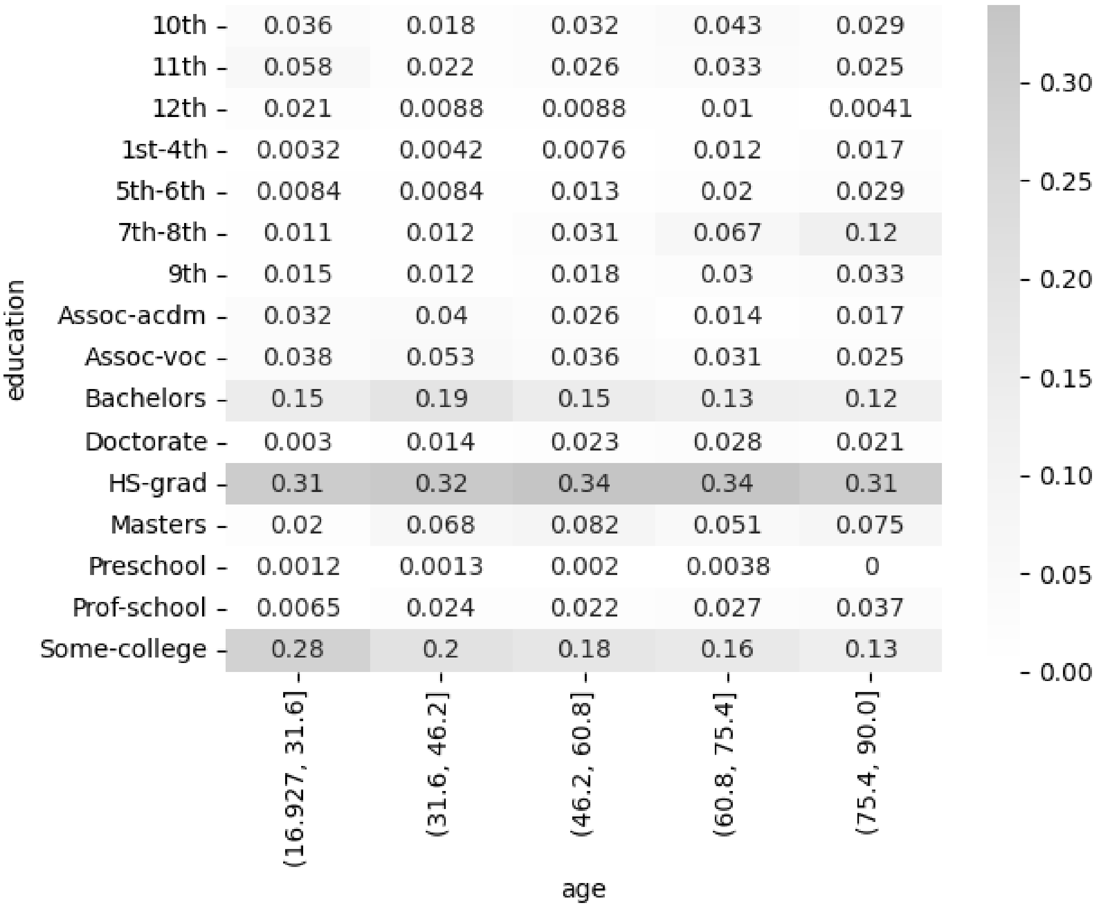
```
education     education-num
10th          6                  933
11th          7                 1175
12th          8                  433
1st-4th       2                  168
5th-6th       3                  333
7th-8th       4                  646
9th           5                  514
Assoc-acdm    12                1067
Assoc-voc     11                1382
Bachelors     13                5355
Doctorate     16                 413
HS-grad       9                10501
Masters       14                1723
Preschool     1                   51
Prof-school   15                 576
Some-college  10                7291
dtype: int64
```

[24]:

```
adult_df.plot.scatter(x='age',y='education-num')
plt.show()
```



[25]:
```
age_discretized = pd.cut(adult_df['age'], bins = 5)
contingency_tbl = pd.crosstab(adult_df.education,age_discretized)
probablity_tbl = contingency_tbl/ contingency_tbl.sum()
sns.heatmap(probablity_tbl, annot=True, center=0.5 ,cmap="Greys")
plt.show()
```

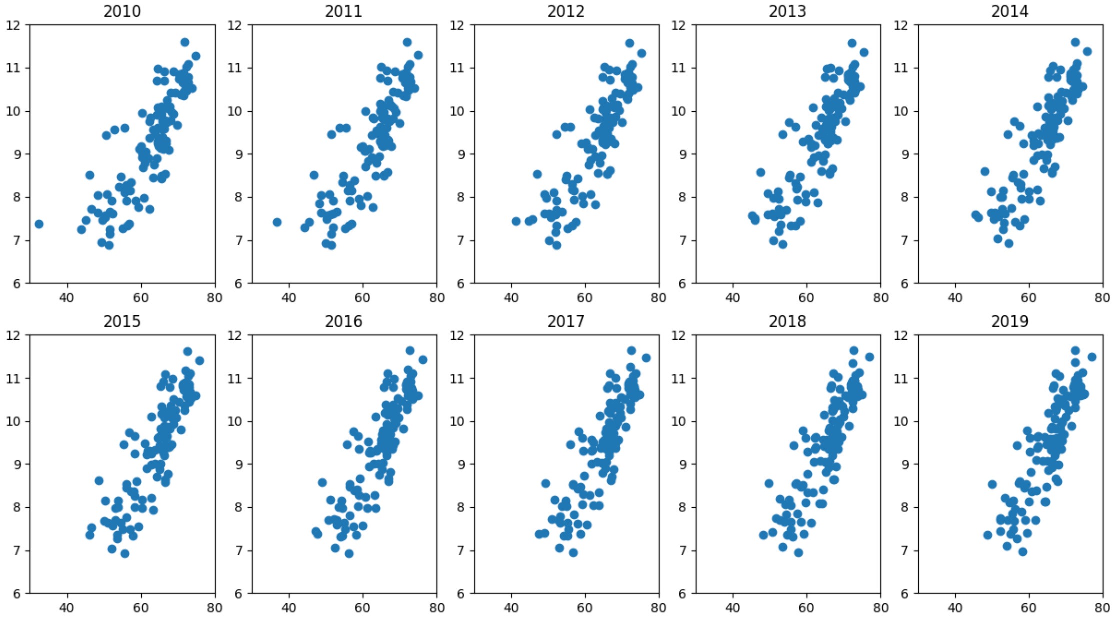


education	(16.927, 31.6]	(31.6, 46.2]	(46.2, 60.8]	(60.8, 75.4]	(75.4, 90.0]
10th	0.036	0.018	0.032	0.043	0.029
11th	0.058	0.022	0.026	0.033	0.025
12th	0.021	0.0088	0.0088	0.01	0.0041
1st-4th	0.0032	0.0042	0.0076	0.012	0.017
5th-6th	0.0084	0.0084	0.013	0.02	0.029
7th-8th	0.011	0.012	0.031	0.067	0.12
9th	0.015	0.012	0.018	0.03	0.033
Assoc-acdm	0.032	0.04	0.026	0.014	0.017
Assoc-voc	0.038	0.053	0.036	0.031	0.025
Bachelors	0.15	0.19	0.15	0.13	0.12
Doctorate	0.003	0.014	0.023	0.028	0.021
HS-grad	0.31	0.32	0.34	0.34	0.31
Masters	0.02	0.068	0.082	0.051	0.075
Preschool	0.0012	0.0013	0.002	0.0038	0
Prof-school	0.0065	0.024	0.022	0.027	0.037
Some-college	0.28	0.2	0.18	0.16	0.13

2.4 Adding visual dimensions

2.4.1 Example of a 5-dimensional scatterplot

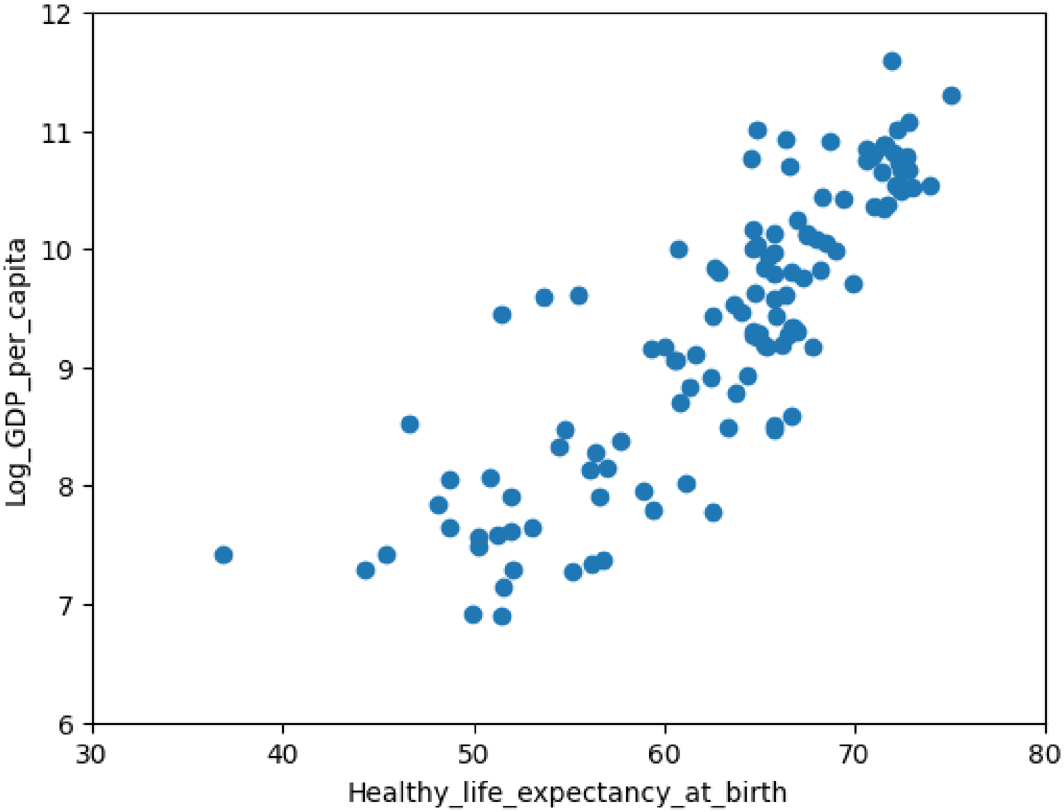
[26]:
```
country_df = pd.read_csv('WH Report_preprocessed.csv')
plt.figure(figsize=(15,8))

year_poss = country_df.year.unique()

for i,yr in enumerate(year_poss):
    BM = country_df.year == yr
    X= country_df[BM].Healthy_life_expectancy_at_birth
    Y= country_df[BM].Log_GDP_per_capita

    plt.subplot(2,5,i+1)
    plt.scatter(X,Y)
    plt.title(yr)
    plt.xlim([30,80])
    plt.ylim([6,12])

plt.show()
plt.tight_layout()
```



```
<Figure size 640x480 with 0 Axes>
```

[27]:
```
def plotyear(year):
    BM = country_df.year == year
    X= country_df[BM].Healthy_life_expectancy_at_birth
    Y= country_df[BM].Log_GDP_per_capita

    plt.scatter(X,Y)
    plt.xlabel('Healthy_life_expectancy_at_birth')
    plt.ylabel('Log_GDP_per_capita')
    plt.xlim([30,80])
    plt.ylim([6,12])
    plt.show()
```

[28]:
```
plotyear(2011)
```



[30]:
```
from ipywidgets import interact, widgets

interact(plotyear,year=widgets.IntSlider(min=2010,max=2019,step=1,value=2010))
```

```
interactive(children=(IntSlider(value=2010, description='year', max=2019,␣
↪min=2010), Output()), _dom_classes=(…
```

[30]:
```
<function __main__.plotyear(year)>
```

The fourth dimension

[31]:
```
Continent_poss = country_df.Continent.unique()
colors_dic={'Asia':'b', 'Europe':'g', 'Africa':'r', 'South America':'c',
            'Oceania':'m', 'North America':'y', 'Antarctica':'k'}

def plotyear(year):
    for cotinent in Continent_poss:
        BM1 = (country_df.year == year)
        BM2 = (country_df.Continent ==cotinent)
        BM = BM1 & BM2
        X = country_df[BM].Healthy_life_expectancy_at_birth
        Y = country_df[BM].Log_GDP_per_capita
        plt.scatter(X,Y,c=colors_dic[cotinent], marker='o',
                    linewidths=0.5,edgecolors='w',label=cotinent)

    plt.xlabel('Healthy_life_expectancy_at_birth')
    plt.ylabel('Log_GDP_per_capita')
    plt.xlim([30,80])
    plt.ylim([6,12])
    plt.legend(ncol=1)
    plt.show()

interact(plotyear,year=widgets.IntSlider(min=2010,max=2019,step=1,value=2010))
```

```
interactive(children=(IntSlider(value=2010, description='year', max=2019,␣
 ↪min=2010), Output()), _dom_classes=(…
```

```
[31]: <function __main__.plotyear(year)>
```

The fifth dimension

```
[33]: Continent_poss = country_df.Continent.unique()
colors_dic={'Asia':'b', 'Europe':'g', 'Africa':'r', 'South America':'c',
            'Oceania':'m', 'North America':'y', 'Antarctica':'k'}
country_df.sort_values(['population'],inplace = True, ascending=False)

def plotyear(year):
    for cotinent in Continent_poss:
        BM1 = (country_df.year == year)
        BM2 = (country_df.Continent ==cotinent)
        BM = BM1 & BM2
        size = country_df[BM].population/200000
        X = country_df[BM].Healthy_life_expectancy_at_birth
        Y= country_df[BM].Log_GDP_per_capita
        plt.scatter(X,Y,s=size,c=colors_dic[cotinent], marker='o',
                    linewidths=0.5,edgecolors='w',label=cotinent)

    plt.xlabel('Healthy_life_expectancy_at_birth')
    plt.ylabel('Log_GDP_per_capita')
    plt.xlim([30,80])
    plt.ylim([6,12])
    plt.legend(markerscale=0.5)
    plt.show()

interact(plotyear,year=widgets.IntSlider(min=2010,max=2019,step=1,value=2010))
```

```
interactive(children=(IntSlider(value=2010, description='year', max=2019,␣
 ↪min=2010), Output()), _dom_classes=(…
```

```
[33]: <function __main__.plotyear(year)>
```

2.5 Showing and comparing Trends

[34]:

```
amazon_df = pd.read_csv('Amazon Stock.csv')
apple_df = pd.read_csv('Apple Stock.csv')
show_table = amazon_df.iloc[5031:5041][['Date','Close']]
show_table.columns = ['Date','Amazon']
show_table = show_table.join(apple_df.iloc[5031:5041]['Close'])
show_table.columns = ['Date','Amazon','Apple']
show_table = show_table.transpose()
show_table.columns = show_table.loc['Date']
show_table.drop(index=['Date'])
```

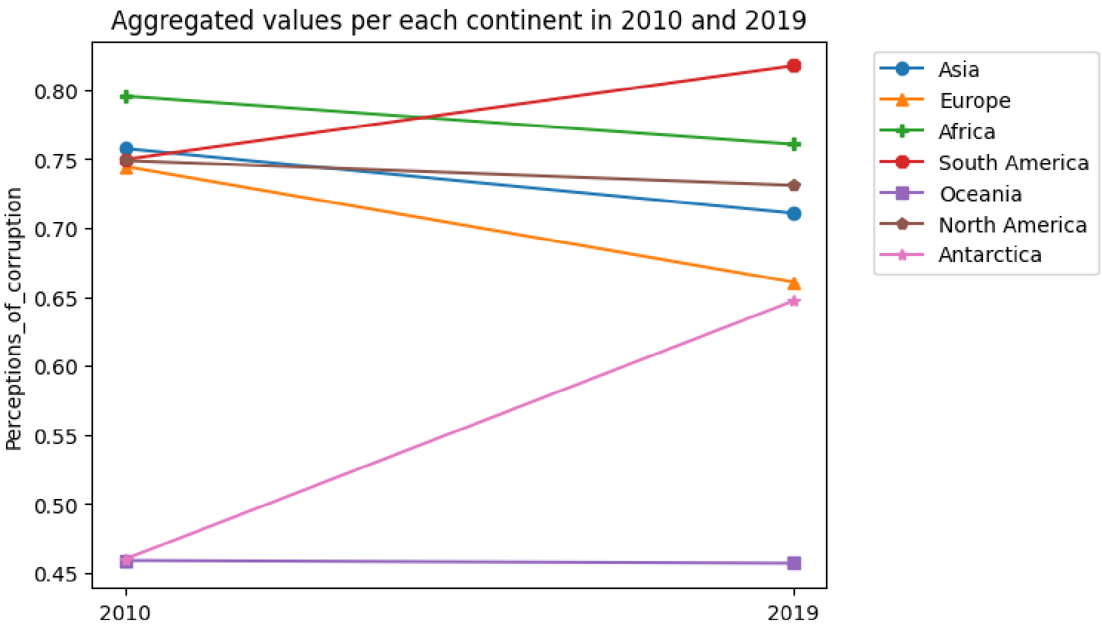
[34]:

```
Date       1/2/2020     1/3/2020     1/6/2020     1/7/2020     1/8/2020  \
Amazon   1898.01001  1874.969971  1902.880005  1906.859985  1891.969971
Apple     74.333511     73.61084    74.197395    73.848442    75.036385

Date         1/9/2020    1/10/2020    1/13/2020    1/14/2020    1/15/2020
Amazon   1901.050049  1883.160034  1891.300049  1869.439941   1862.02002
Apple      76.630219    76.803459    78.444321    77.385063    77.053429
```

2.5.1 Example of visualizing and comparing trends

[35]:

```
country_df = pd.read_csv('WH Report_preprocessed.csv')
continent_poss = country_df.Continent.unique()
byContinentYear_df = country_df.groupby(['Continent','year']).
  ↪Perceptions_of_corruption.mean()
Markers_options = ['o', '^','P', '8', 's', 'p', '*']

for i,c in enumerate(continent_poss):
    plt.plot([2010,2019],byContinentYear_df.loc[c,[2010,2019]],
            label=c,marker=Markers_options[i])
plt.xticks([2010,2019])
plt.legend(bbox_to_anchor=(1.05, 1))
plt.title('Aggregated values per each continent in 2010 and 2019')
plt.ylabel('Perceptions_of_corruption')
plt.show()
```


Aggregated values per each continent in 2010 and 2019
Perceptions_of_corruption
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
2010
2019
Asia
Europe
Africa
South America
Oceania
North America
Antarctica