Correlations

Introducing an additional library: seaborn - for statistical data visualization

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

We'll work with the California Housing data

In [2]:
```
url = "https://raw.githubusercontent.com/ulhlin/data-analytics/main/datasets/housing.csv"
house_df = pd.read_csv(url)
house_df.head()
```

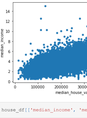
Out[2]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	

1. Correlation computation and scatterplots

In [3]:
```
house_df.plot.scatter(x = 'median_house_value', y = 'median_income')
```

Out[3]:
```
<AxesSubplot:xlabel='median_house_value', ylabel='median_income'>
```



In [4]:
```
house_df[['median_income', 'median_house_value']].corr(method='pearson')
```

Out[4]:

	median_income	median_house_value
median_income	1.000000	0.688075
median_house_value	0.688075	1.000000

In [5]:
```
house_df.corr(method='pearson')
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median
longitude	1.000000	-0.924664	-0.108197	0.044568	0.069608	0.099773	0.055310	-0.015176	
latitude	-0.924664	1.000000	0.011173	-0.036100	-0.066983	-0.108785	-0.071035	-0.079809	
housing_median_age	-0.108197	0.011173	1.000000	-0.361262	-0.320451	-0.296244	-0.302916	-0.119034	
total_rooms	0.044568	-0.036100	-0.361262	1.000000	0.930380	0.857126	0.918484	0.198050	
total_bedrooms	0.069608	-0.066983	-0.320451	0.930380	1.000000	0.877747	0.979728	-0.007723	
population	0.099773	-0.108785	-0.296244	0.857126	0.877747	1.000000	0.907222	0.004834	
households	0.055310	-0.071035	-0.302916	0.918484	0.979728	0.907222	1.000000	0.013033	
median_income	-0.015176	-0.079809	-0.119034	0.198050	-0.007723	0.004834	0.013033	1.000000	
median_house_value	-0.045967	-0.144160	0.105623	0.134153	0.049686	-0.024650	0.065843	0.688075	

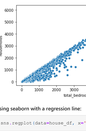
In [6]:
```
house_df.plot.scatter(x = 'total_bedrooms', y = 'households')
```

Out[6]:
```
<AxesSubplot:xlabel='total_bedrooms', ylabel='households'>
```

Almost similar - using matplotlib.pyplot as pic function:

In [7]:
```
import matplotlib.pyplot as plt
plt.scatter(house_df['total_bedrooms'], house_df['households'])
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x26d72f20430>
```



Using seaborn:

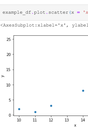
In [8]:
```
sns.scatterplot(data=house_df, x='total_bedrooms', y='households')
```

Out[8]:
```
<AxesSubplot:xlabel='total_bedrooms', ylabel='households'>
```

using seaborn with a regression line

In [9]:
```
sns.regplot(data=house_df, x='total_bedrooms', y='households')
```

Out[9]:
```
<AxesSubplot:xlabel='total_bedrooms', ylabel='households'>
```



Scatterplots work when there are missing data

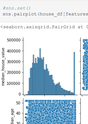
In [10]:
```
x = pd.Series(range(15, 20))
y = pd.Series([2, 1, 3, np.nan, 8, 7, 18, 25])
example_df = pd.DataFrame({'x': x, 'y': y})
example_df
```

Out[10]:

	x	y
0	10	2.0
1	11	1.0
2	12	3.0
3	13	NaN
4	14	8.0
5	15	7.0
6	16	18.0
7	17	25.0
8	18	NaN
9	19	NaN

In [11]:
```
example_df.plot.scatter(x = 'x', y = 'y')
```

Out[11]:
```
<AxesSubplot:xlabel='x', ylabel='y'>
```



In [12]:
```
example_df.corr(method='pearson')
```

Out[12]:

	x	y
x	1.000000	0.831833
y	0.831833	1.000000

In [13]:
```
example_df.corr(method='spearman')
```

Out[13]:

	x	y
x	1.000000	0.900937
y	0.900937	1.000000

In [14]:
```
example_df.corr(method='kendall')
```

Out[14]:

	x	y
x	1.00000	0.78072
y	0.78072	1.00000

2. Scatterplot matrix

The diagonal shows the distribution of the three numeric variables.

In the other cells of the plot matrix, we have the scatterplots of each variable combination in the dataframe.

In [15]:
```
features = ['median_house_value', 'housing_median_age',
            'median_income']
pd.plotting.scatter_matrix(house_df[features])
```

Out[15]:
```
array([[<AxesSubplot:xlabel='median_house_value', ylabel='median_house_value'>,
        <AxesSubplot:xlabel='housing_median_age', ylabel='median_house_value'>,
        <AxesSubplot:xlabel='median_income', ylabel='median_house_value'>],
       [<AxesSubplot:xlabel='median_house_value', ylabel='housing_median_age'>,
        <AxesSubplot:xlabel='housing_median_age', ylabel='housing_median_age'>,
        <AxesSubplot:xlabel='median_income', ylabel='housing_median_age'>],
       [<AxesSubplot:xlabel='median_house_value', ylabel='median_income'>,
        <AxesSubplot:xlabel='housing_median_age', ylabel='median_income'>,
        <AxesSubplot:xlabel='median_income', ylabel='median_income'>]],
      dtype=object)
```

In [16]:
```
#enx.set()
sns.pairplot(house_df[features], height = 2.5)
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x26d7d0b89>
```

3. Heatmaps

Pandas doesn't contain a built-in heatmap function. We can try and create one by adding color to corr:

In [17]:
```
correlation_matrix = house_df[features].corr()
correlation_matrix.style.background_gradient(cmap='coolwarm')
```

Out[17]:

	median_house_value	housing_median_age	median_income
median_house_value	1.000000	0.105623	0.688075
housing_median_age	0.105623	1.000000	-0.119034
median_income	0.688075	-0.119034	1.000000

In [18]:
```
correlation_matrix.style.background_gradient(cmap='Blues')
```

Out[18]:

	median_house_value	housing_median_age	median_income
median_house_value	1.000000	0.105623	0.688075
housing_median_age	0.105623	1.000000	-0.119034
median_income	0.688075	-0.119034	1.000000

Or we can use seaborn

In [19]:
```
features = ['median_house_value', 'housing_median_age','median_income','total_bedrooms','population']
correlation_matrix = house_df[features].corr().round(2)
sns.heatmap(data=correlation_matrix,cmap='Greens', annot=True)
```

Out[19]:
```
<AxesSubplot:>
```


Summary

- `.corr` - compute pairwise correlation of columns, excluding NA/null values. documentation
- `.corr.style.background_gradient` - change the background color. various options
- `.plotting.scatter_matrix` - draw a matrix of scatter plots, documentation
- `.plot.scatter` - plot a scatter plot

Seaborn package:

- `sns.scatterplot` - a scatter plot
- `sns.regplot` - a scatter plot with a regression line
- `sns.pairplot` - scatter plot matrix
- `sns.heatmap` - a heatmap. @annot = True to print the values inside the square

In [ ]: