univariate_data

March 4, 2021

[1]:
```
# Import the pandas and numpy library
import pandas as pd
import numpy as np

# Load the iris.csv dataset
iris = pd.read_csv('assets/iris.csv')

# Look at the first 5 rows
print(iris.head())

#info method to print information about the data frame including the index␣
 ↪dtype and column dtypes,
#and non-null values
iris.info()
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[2]:
```
# Just the mean for one variable
iris['sepal_length'].mean()
```

[2]: 5.843333333333335

[3]:
```
# Just the standard deviation for one variable
iris['sepal_length'].std()
```

[3]:
```
0.8280661279778629
```

[4]:
```
#However, when exploring our data we usually want to know a bit more about it.␣
 ↪So, let's use Pandas describe function
#to calculate  the mean, standard deviation and interquartile range IQR values␣
 ↪for sepal_length.

iris['sepal_length'].describe()
```

[4]:
```
count    150.000000
mean       5.843333
std        0.828066
min        4.300000
25%        5.100000
50%        5.800000
75%        6.400000
max        7.900000
Name: sepal_length, dtype: float64
```

[5]:
```
#We can call the .describe function on the iris data, and it will exclude the␣
 ↪character columns and provide summary
#statistics of numeric columns.

iris.describe()
```

[5]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

[6]:
```
# summary statistics of character column

iris.describe(include='all')
```

[6]:

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	virginica
freq	NaN	NaN	NaN	NaN	50

```
mean        5.843333     3.057333      3.758000     1.199333        NaN
std         0.828066     0.435866      1.765298     0.762238        NaN
min         4.300000     2.000000      1.000000     0.100000        NaN
25%         5.100000     2.800000      1.600000     0.300000        NaN
50%         5.800000     3.000000      4.350000     1.300000        NaN
75%         6.400000     3.300000      5.100000     1.800000        NaN
max         7.900000     4.400000      6.900000     2.500000        NaN
```

[7]:
```
# Get the mean of the petal length per group
iris.groupby('species')['petal_length'].mean()
```

[7]:
```
species
setosa        1.462
versicolor    4.260
virginica     5.552
Name: petal_length, dtype: float64
```

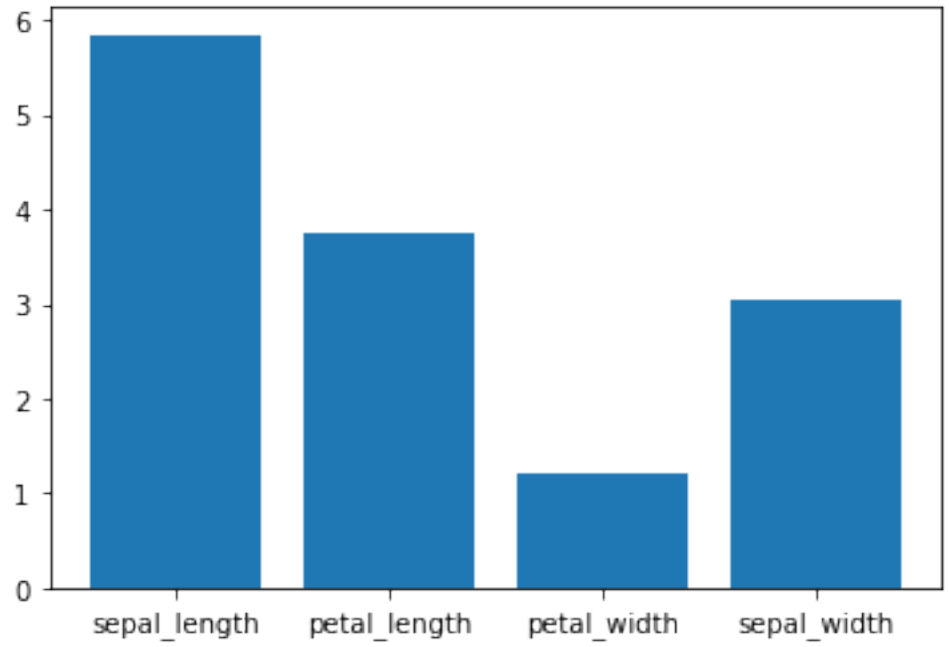
[8]:
```
#explore summary statistics by each group
iris.groupby('species').describe()
```

[8]:
```
           sepal_length                                                \
                  count   mean       std  min    25%  50%  75%  max
species
setosa             50.0  5.006  0.352490  4.3  4.800  5.0  5.2  5.8
versicolor         50.0  5.936  0.516171  4.9  5.600  5.9  6.3  7.0
virginica          50.0  6.588  0.635880  4.9  6.225  6.5  6.9  7.9

           sepal_width         … petal_length      petal_width         \
                 count   mean  …          75%  max       count   mean
species                        …
setosa            50.0  3.428  …        1.575  1.9        50.0  0.246
versicolor        50.0  2.770  …        4.600  5.1        50.0  1.326
virginica         50.0  2.974  …        5.875  6.9        50.0  2.026


                 std  min  25%  50%  75%  max
species
setosa      0.105386  0.1  0.2  0.2  0.3  0.6
versicolor  0.197753  1.0  1.2  1.3  1.5  1.8
virginica   0.274650  1.4  1.8  2.0  2.3  2.5

[3 rows x 32 columns]
```

[9]:
```
import matplotlib as mpl
mpl.get_backend()

import matplotlib.pyplot as plt
```

[10]:
```
# Calculate the mean
sepal_length_mean = iris['sepal_length'].mean()
petal_length_mean = iris['petal_length'].mean()
petal_width_mean = iris['petal_width'].mean()
sepal_width_mean = iris['sepal_width'].mean()

# Calculate the STD
sepal_lengthstd = iris['sepal_length'].std()
petal_lengthstd = iris['petal_length'].std()
petal_widthstd = iris['petal_width'].std()
sepal_widthstd = iris['sepal_width'].std()
```

[11]:
```
# Build a bar plot
plt.bar(['sepal_length', 'petal_length', 'petal_width',␣
 ↪'sepal_width'],[sepal_length_mean, petal_length_mean,petal_width_mean␣
 ↪,sepal_width_mean])
plt.show()
```



[12]:
```
iris_mean =iris.groupby('species').mean()
iris_mean
```

[12]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326

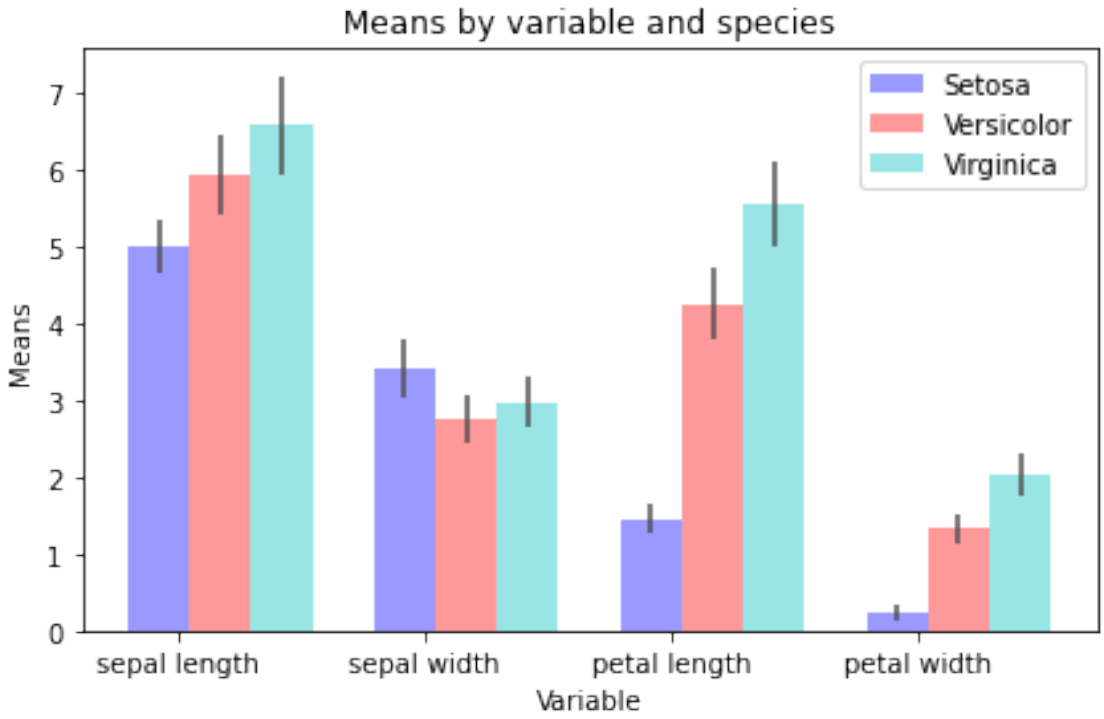
```
virginica          6.588        2.974         5.552        2.026
```

[13]:
```
iris_std =iris.groupby('species').std()
iris_std
```

[13]:
```
            sepal_length  sepal_width  petal_length  petal_width
species
setosa          0.352490     0.379064      0.173664     0.105386
versicolor      0.516171     0.313798      0.469911     0.197753
virginica       0.635880     0.322497      0.551895     0.274650
```

[14]:
```
n_groups = 4

means_setosa = (5.006, 3.428, 1.462, 0.246)
std_setosa= (0.352490, 0.379064, 0.173664, 0.10538)

means_versicolor = (5.936, 2.770,4.260,1.326)
std_versicolor= (0.516171, 0.313798,0.469911,0.197753)

means_virginica = (6.588, 2.974, 5.552, 2.026)
std_virginica= (0.635880, 0.322497, 0.551895, 0.274650)
# Create a figure and a set of subplots.
fig, ax = plt.subplots()

index = np.arange(n_groups) # the x locations for the groups
bar_width = 0.25 # the width of the bars

opacity = 0.4
error_config = {'ecolor': '0.3'} # and error bars

# ax.bar is to make a bar plot.
# The bars are positioned at x with the given alignment. Their dimensions are␣
 ↪given by height and width. The vertical baseline is bottom (default 0).
# Many parameters can take either a single value applying to all bars or a␣
 ↪sequence of values, one for each bar.
rects1 = ax.bar(index, means_setosa, bar_width,
                alpha=opacity, color='b',
                yerr=std_setosa, error_kw=error_config,
                label='Setosa')

rects2 = ax.bar(index + bar_width, means_versicolor, bar_width,
                alpha=opacity, color='r',
                yerr=std_versicolor, error_kw=error_config,
                label='Versicolor')

rects3 = ax.bar(index + bar_width + bar_width, means_virginica, bar_width,
                alpha=opacity, color='c',
```

```
                yerr=std_virginica, error_kw=error_config,
                label='Virginica')

#After creating the plot we can add information to plot to make it more␣
 ↪readable.
#So, our final step is to add some text for labels, title and axes ticks.
# add usful information
ax.set_xlabel('Variable')
ax.set_ylabel('Means')
ax.set_title('Means by variable and species')
ax.set_xticks(index + bar_width / 3)
ax.set_xticklabels(('sepal length', 'sepal width', 'petal length', 'petal␣
 ↪width'))
ax.legend()

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



[ ]: