Statistical Thinking in Python Part 1

2). Quantitative exploratory data analysis

a). Computing mean

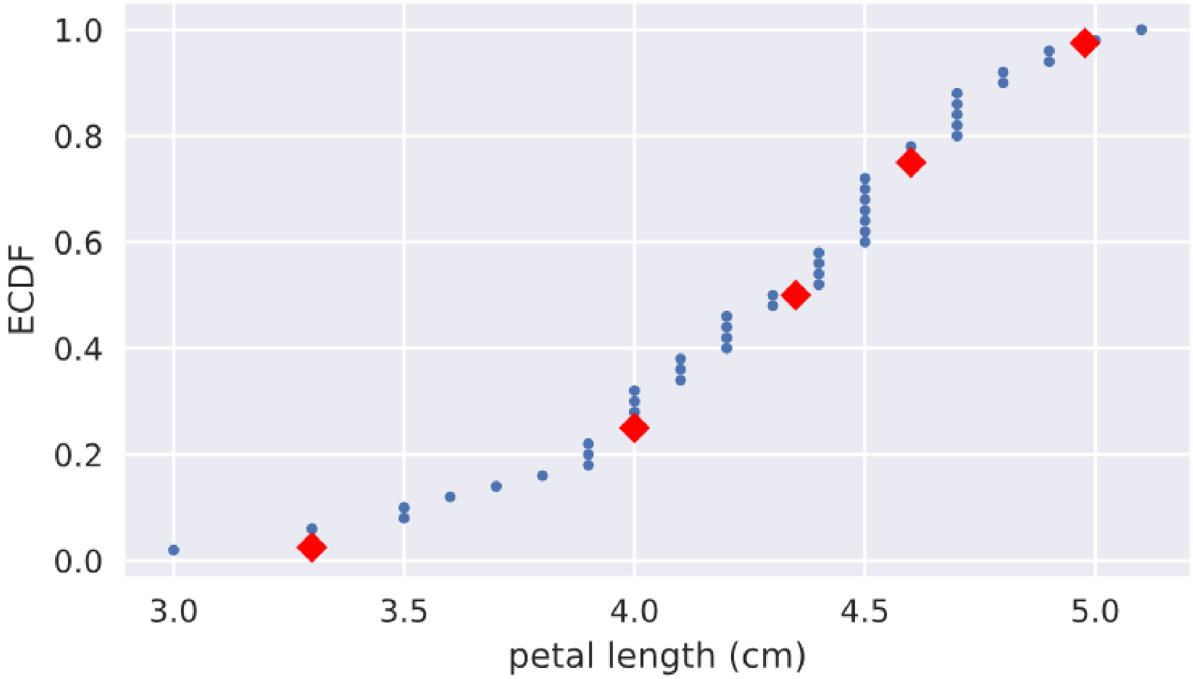
```
# Compute the mean: mean_length_vers
mean_length_vers=np.mean(versicolor_petal_length)
# Print the result with some nice formatting
print('I. versicolor:', mean_length_vers, 'cm')
```

b). Computing Percentiles

```
# Specify array of percentiles: percentiles
percentiles = np.array([2.5,25,50,75,97.5])

# Compute percentiles: ptiles_vers
ptiles_vers=np.percentile(versicolor_petal_length,percentiles)

# Print the result
print(ptiles_vers)
```

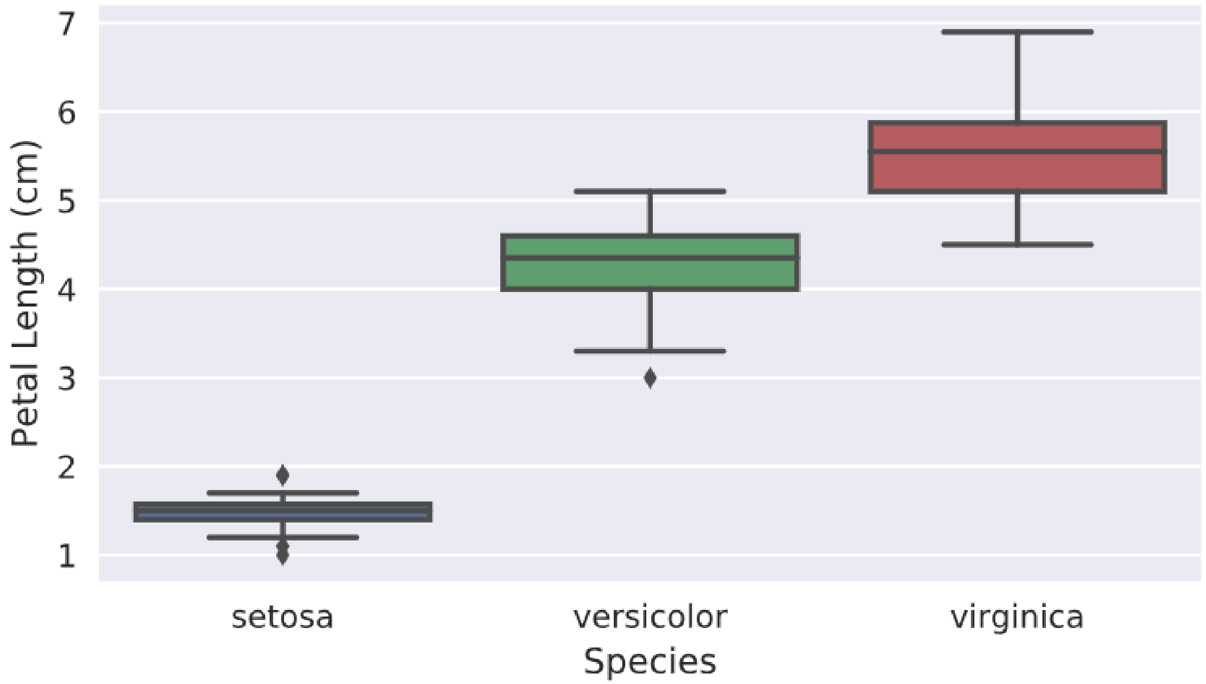
c). **Comparing Percentiles to ECDF**

```
# Plot the ECDF
_ = plt.plot(x_vers, y_vers, '.')
_ = plt.xlabel('petal length (cm)')
_ = plt.ylabel('ECDF')

# Overlay percentiles as red diamonds.
_ = plt.plot(ptiles_vers, percentiles/100, marker='D', color='red',
        linestyle='none')

# Show the plot
plt.show()
```



d). Box and Whisker Plot

```
# Create box plot with Seaborn's default settings
_=sns.boxplot(x='species', y='petal length (cm)', data =df )

# Label the axes
_=plt.xlabel('Species')
_=plt.ylabel('Petal Length (cm)')

# Show the plot
plt.show()
```



e). Computing the Variance

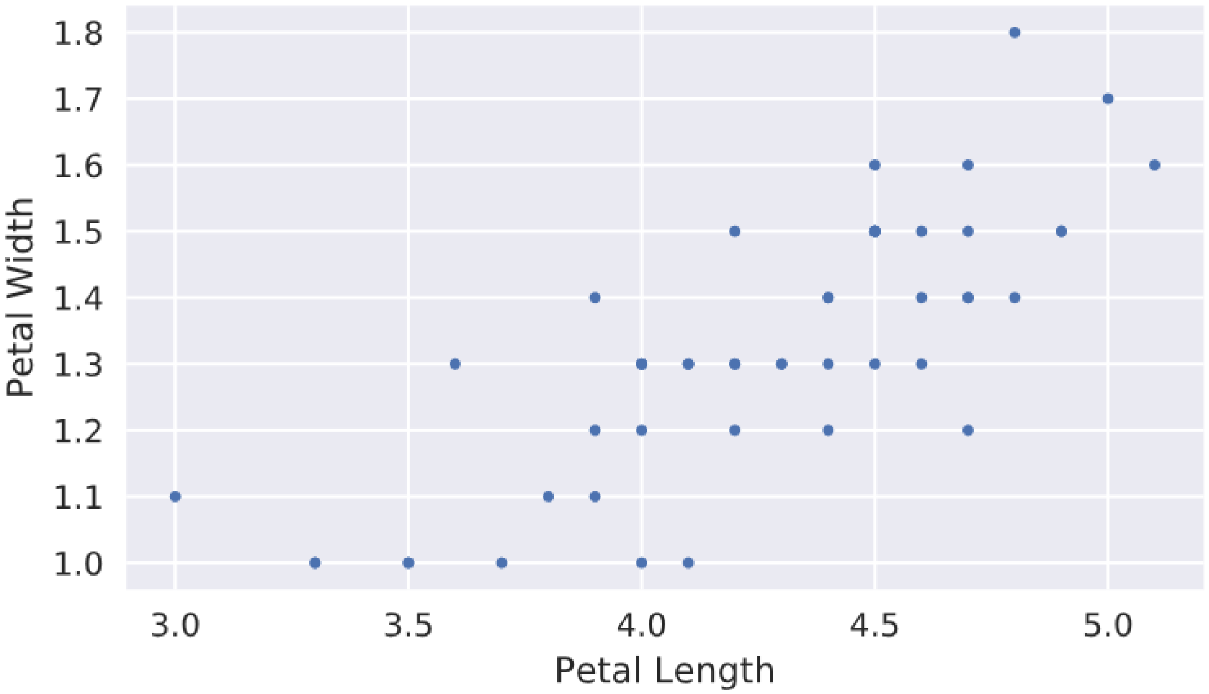
```
# Array of differences to mean: differences
differences=np.array(versicolor_petal_length-np.mean(versicolor_petal_length))

# Square the differences: diff_sq
diff_sq=differences**2

# Compute the mean square difference: variance_explicit
variance_explicit=np.mean(diff_sq)

# Compute the variance using NumPy: variance_np
variance_np=np.var(versicolor_petal_length)
# Print the results
print(variance_explicit,variance_np)
```

f). The Standard deviation and Variance

```
# Compute the variance: variance
variance=np.var(versicolor_petal_length)

# Print the square root of the variance
print(np.sqrt(variance))

# Print the standard deviation
print(np.std(versicolor_petal_length))
```

g). Scatter Plots

```
# Make a scatter plot
plt.plot(versicolor_petal_length,versicolor_petal_width,marker='.',linestyle='none')

# Label the axes
_=plt.xlabel('Petal Length')
_=plt.ylabel('Petal Width')

# Show the result
plt.show()
```

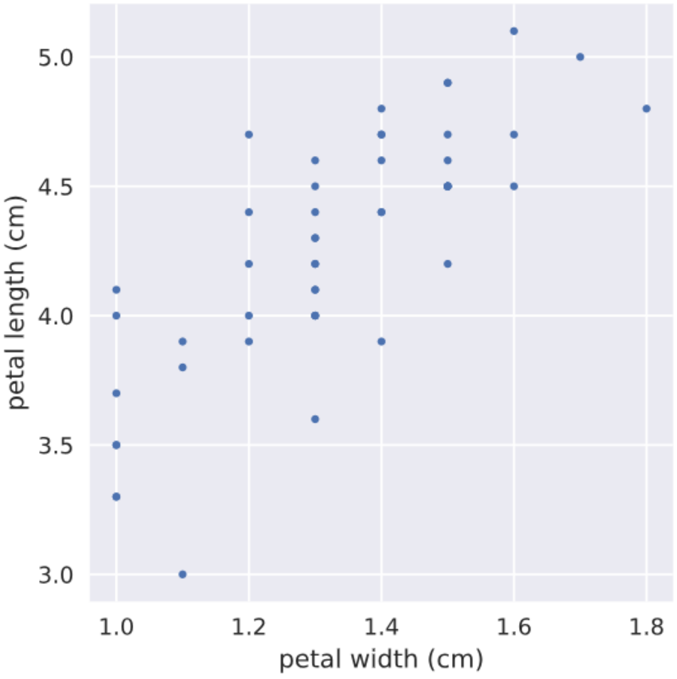


h). Computing the Covariance

```
# Compute the covariance matrix: covariance_matrix
covariance_matrix=np.cov(versicolor_petal_length,versicolor_petal_width)
# Print covariance matrix
print(covariance_matrix)

# Extract covariance of length and width of petals: petal_cov
petal_cov=covariance_matrix[0,1]

# Print the length/width covariance
print(petal_cov)
```

<script.py> output:

```
    [[0.22081633 0.07310204]
     [0.07310204 0.03910612]]
    0.07310204081632653
```



i). Computing the Pearson correlation coefficient

```
def pearson_r(x, y):
    """Compute Pearson correlation coefficient between two arrays."""
    # Compute correlation matrix: corr_mat
    corr_mat= np.corrcoef(x,y)

    # Return entry [0,1]
    return corr_mat[0,1]

# Compute Pearson correlation coefficient for I. versicolor: r
r= pearson_r(versicolor_petal_length, versicolor_petal_width)

# Print the result
print(r)
```