6 - Computing Statistics and Percentiles

Computing general statistics and percentiles is a popular way to generate quick inference on a given data set. Some of these popular statistics include mean, median, mode, variance, skewness, kurtosis and central moments. This information can almost always be generated very quickly using tools in both R and Python. Percentiles of interest are usually those that correspond to quartiles, specifically the first and third from which one generate an interquartile range (IQR). This section will cover how to compute these statistics and desired percentiles in both R and python. One will notice the extreme similarity in code between R and Python for these tasks.

6.1 - Computing Basic Statistics

Computing popular statistics for data sets is something both R and Python can handle easily. For a set of data points $X = [x_1 \dots x_n]$, $x_i \in \mathbb{R}$ one can define:

The mean of X as:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The median of X as:

$$Med(X) = \begin{cases} X[\frac{n}{2}] & n\ even \\ (X[\frac{n-1}{2}] + X[\frac{n+1}{2}])/2 & n\ odd \end{cases}$$

The variance of X as:

$$Var(X) = \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X})^2$$

The standard deviation of X as:

$$sd(X) = \sqrt{Var(X)} = \sigma$$

The mode as the element in X which occurs most frequently (can be more than one).

The kurtosis as:

$$Kurt(X) = n \cdot \frac{\sum_{i=1}^{n}(x_i - \bar{X})^4}{(\sum_{i=1}^{n}(x_i - \bar{X})^2)^2}$$

The skewness as:

$$Skew(X) = n \cdot \frac{(\sum_{i=1}^{n}(x_i - \bar{X})^3)^2}{(\sum_{i=1}^{n}(x_i - \bar{X})^2)^3}$$

The k^{th} central moment as:

$$\frac{1}{n}\cdot\sum_{i=1}^{n}(x_i - \bar{X})^k$$

It should be noted that these statistics can be computed across a given dimension for data of dimension higher than 1. For example if given a data matrix, these statistics can be computed across rows or columns.

6.1.1 - Statistics in Python

The NumPy library has built-in functions for the major statistical properties such as mean, median, variance and standard deviation. One can compute the desired statistics using the NumPy functions; `mean()`, `median()`, `var()` and `std()`.

```
data = np.array([1,2,3,4,2])

# Compute using NumPy
mean = np.mean(data)
median = np.median(data)
var = np.var(data)
std = np.std(data)

# Display results:
print('Data:', data)
print('Mean:', mean)
print('Median:', median)
print('Variance:', var)
print('Standard Deviation:', round(std, 4))
```

```
Data: [1 2 3 4 2]
Mean: 2.4
Median: 2.0
Variance: 1.04
Standard Deviation: 1.0198
```

The SciPy 'stats' library is needed for more detailed properties such as mode, kurtosis (Fisher or Pearson), skewness and n^{th} moments. As per the SciPy stats documentation page [9] "This module contains a large number of probability distributions, summary and frequency statistics, correlation functions and statistical tests, masked statistics, kernel density estimation, quasi-Monte Carlo functionality, and more." The SciPy stats library is imported using `import scipy.stats` which is the same import call used throughout this document.

```
import scipy.stats as stats # statistics tools
```

One can computed desired statistics using SciPy stats functions `mode()`, `kurtosis()`, `skew()` and `moment()`.

```
# Compute using Scipy
mode = stats.mode(data)[0][0]
kurt_pear = stats.kurtosis(data, fisher=False)
skew = stats.skew(data)
moment3 = stats.moment(data, moment=3)

# Display results:
print('Data:', data)
print('Mode:', mode)
print('Pearson Kurtosis:', round(kurt_pear, 4))
print('Skewness:', round(skew, 4))
print('3rd Moment:', round(moment3, 4))
```

```
Data: [1 2 3 4 2]
Mode: 2
Pearson Kurtosis: 1.9556
Skewness: 0.2715
3rd Moment: 0.288
```

As perviously stated, these functions are built so that they can be computed across any array axis, not just for one dimensional vector arrays.

```
data = np.array([[1,2,3,4,2], [1,2,3,4,2]])
data
```

```
array([[1, 2, 3, 4, 2],
       [1, 2, 3, 4, 2]])
```

```
mean2 = np.mean(data, axis = 1)
skew2 = stats.skew(data, axis = 1)
mean2, skew2
```

```
(array([2.4, 2.4]), array([0.27154542, 0.27154542]))
```

6.1.2 - Computing Statistics in R

R was built for statistical analysis and hence has many built-in functions to compute popular statistics, these include `mean()`, `median()`, `var()` and `std()`. Note that R does not have a built-in 'mode' function but the calculation can be done fairly easily using the call `names(sort(table(data), TRUE))[[1]]` for the given data.

```
data <- c(1,2,3,4,2)

# Compute using built-in R functions
mean <- mean(data)
median <- median(data)
var <- var(data)
std <- sd(data)

# Manual computation (no built-in functions)
mode <- names(sort(table(data), TRUE))[[1]]

# Display results:
print('Data:')
print(data)
print(paste0('Mean: ', mean))
print(paste0('Median: ', median))
print(paste0('Variance: ', var))
print(paste0('Standard Deviation: ', round(std, 4)))
print(paste0('Mode: ', mode))
```

```
[1] "Data:"
[1] 1 2 3 4 2
[1] "Mean: 2.4"
[1] "Median: 2"
[1] "Variance: 1.3"
[1] "Standard Deviation: 1.1402"
[1] "Mode: 2"
```

Notice results are identical to those produced in Python. In order to compute more complex statistics such as kurtosis, skewness and moments the package 'moments' is required.

6.1.3 - R moments Package

As per the moments documentation page [13] Moments, Cumulants, Skewness, Kurtosis and Related Tests, moments can be described as "Functions to calculate: moments, Pearson's kurtosis, Geary's kurtosis and skewness; tests related to them (Anscombe-Glynn, D'Agostino, Bonett-Seier)." We will use the package to compute more complex statistics for which there is no built-in R function. The moments package can be installed in R or R Studio with the command `install.packages('moments')`. Once the package is installed it can be loaded/imported in with the command `library(moments)` after which all the package functions will be available for use. It should be noted that moments version 0.14 is being used.

```
library(moments)
packageVersion("moments")
```

```
[1] '0.14'
```

Below the moments functions `kurtosis()`, `skewness()` and `moment()` are used to compute the desired statistics.

```
# Compute using "moments" package
kurt_pear <- kurtosis(data)
skew <- skewness(data)
moment3 <- moment(data, order = 3, central = TRUE)

# Display results:
print('Data:')
print(data)
print(paste0('Pearson Kurtosis: ', round(kurt_pear, 4)))
print(paste0('Skewness: ', round(skew, 4)))
print(paste0('3rd Moment: ', round(moment3, 4)))
```

```
[1] "Data:"
[1] 1 2 3 4 2
[1] "Pearson Kurtosis: 1.9556"
[1] "Skewness: 0.2715"
[1] "3rd Moment: 0.288"
```

Notice results are identical to those produced in Python.

6.2 - Computing Percentiles

Other common measures of location are Quartiles and Percentiles. It should be noted that Quartiles are just special percentiles. The first quartile, Q_1, is the same as the 25^{th} percentile. The third quartile, Q_3 is the same as the 75^{th} percentile. The median or second quartile, Q_2 is the same as the 50^{th} percentile. A k^{th} percentile can be defined as a score below which a given percentage k of scores in its frequency distribution falls.

6.2.1 - Percentiles in Python

NumPy provides the function `percentile()` for calculating percentiles. The SciPy stats function base also provides a function for calculating the inter quartile range `iqr()` which is the difference between the third and first quartile.

```
data = np.arange(11)
data
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
# quartiles
Q50 = np.percentile(data, 50) # 2nd quartile
Q25 = np.percentile(data, 25) # 1st quartile
Q75 = np.percentile(data, 75) # 3rd quartile
Q25, Q50, Q75
```

```
(2.5, 5.0, 7.5)
```

```
IQR = stats.iqr(data)
IQR
```

```
5.0
```

```
# percentiles
P30 = np.percentile(data, 30) # 30th percentile
P60 = np.percentile(data, 60) # 60th percentile
P30, P60
```

```
(3.0, 6.0)
```

6.2.2 - Percentiles in R

R provides the function `quantile()` for calculating percentiles and also an `IQR()` function for calculating the interquartile range. These are built in functions and do not require external packages.

```
data <- c(0:10)
print(data)
```

```
 [1]  0  1  2  3  4  5  6  7  8  9 10
```

```
# quartiles
Q <- quantile(data, c(0.25, 0.5, 0.75))
print(Q)
```

```
25% 50% 75%
2.5 5.0 7.5
```

```
IQR <- IQR(data)
IQR
```

```
5
```

```
# percentiles
P <- quantile(data, c(0.3, 0.6))
print(P)
```

```
30% 60%
  3   6
```

Notice results are identical to those produced in Python.

[9] Virtanen, P. et al., 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, [link] (https://scipy.org/)
[13] Lukasz Komsta, Frederick Novomestky, 2022-05-02. Moments, Cumulants, Skewness, Kurtosis and Related Tests, [link] (https://CRAN.R-project.org/package=moments)