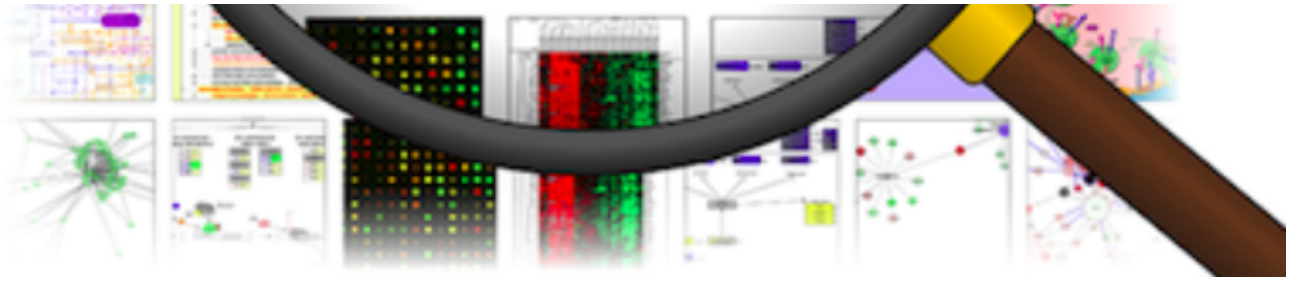
Baffled by Covariance and Correlation??? Get the Math and the Application in Analytics for both the terms..

Srishti Saha
Oct 5, 2018 · 12 min read

Covariance and correlation are two significantly used terms in the field of statistics and probability theory. Most articles and reading material on probability and statistics presume a basic understanding of terms like means, standard deviation, correlations, sample sizes and covariance. Let us demystify a couple of these terms today so that we can move ahead with the rest. The aim of the article is to define the terms: correlation and covariance matrices, differentiate between the two and understand the application of the two in the field of analytics and datasets.



I am creating an index for easy reference to the topics:

1. Demystifying the terms
2. Defining the Terms Mathematically
3. Covariance versus Correlation
4. Application in Analytics

Demystifying the terms

In simple words, both the terms measure the relationship and the dependency between two variables. “Covariance” indicates the direction of the linear relationship between variables. “Correlation” on the other hand measures both the strength and direction of the linear relationship between two variables. Correlation is a function of the covariance. **What sets them apart is the fact that correlation values are standardized whereas, covariance values are not.** You can obtain the correlation coefficient of two variables by dividing the covariance of these variables by the product of the standard deviations of the same values. If we revisit the definition of Standard Deviation, it essentially measures the absolute variability of a datasets’ distribution. When you divide the covariance values by the standard deviation, it essentially scales the value down to a limited range of **-1 to +1**. This is precisely the range of the correlation values.

Defining the Terms Mathematically

Let us now visit the mathematical definitions of these terms.

Covariance

The covariance of two variables (x and y) can be represented as cov(x,y). If E[x] is the expected value or mean of a sample ‘x’, then cov(x,y) can be represented in the following way:

$$cov(x,y) = E[(x-\mu_x)(y-\mu_y)]$$
$$= E[xy] - E[x]E[y]$$
$$= E[xy] - \mu_x\mu_y$$
$\forall\ \mu_x$ & μ_y = $E[x]$ & $E[y]$ respectively.

If we look at a single variable, say 'y', cov(y,y), the expression can be written in the following way:

$$cov(y,y) = Var(y) = \sigma^2(y) = \sigma_y^2$$
Here, Var → variance of variable y
$$\boxed{Var(y) = E[(y-\mu_y)^2]}$$

Also; $Var(y) = s^2$
where $s^2 \Rightarrow$ sample variance

Now as we see, in the image above, 's^2' or sampled variance is basically the covariance of a variable with itself. This term can also be defined in the following manner:

$$s^2 = cov(x,x) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}$$
$$= \frac{\sum_{i=1}^{n}(x_i-\bar{x})(x_i-\bar{x})}{n-1} \quad \text{——— (A)}$$

for 2 variables:

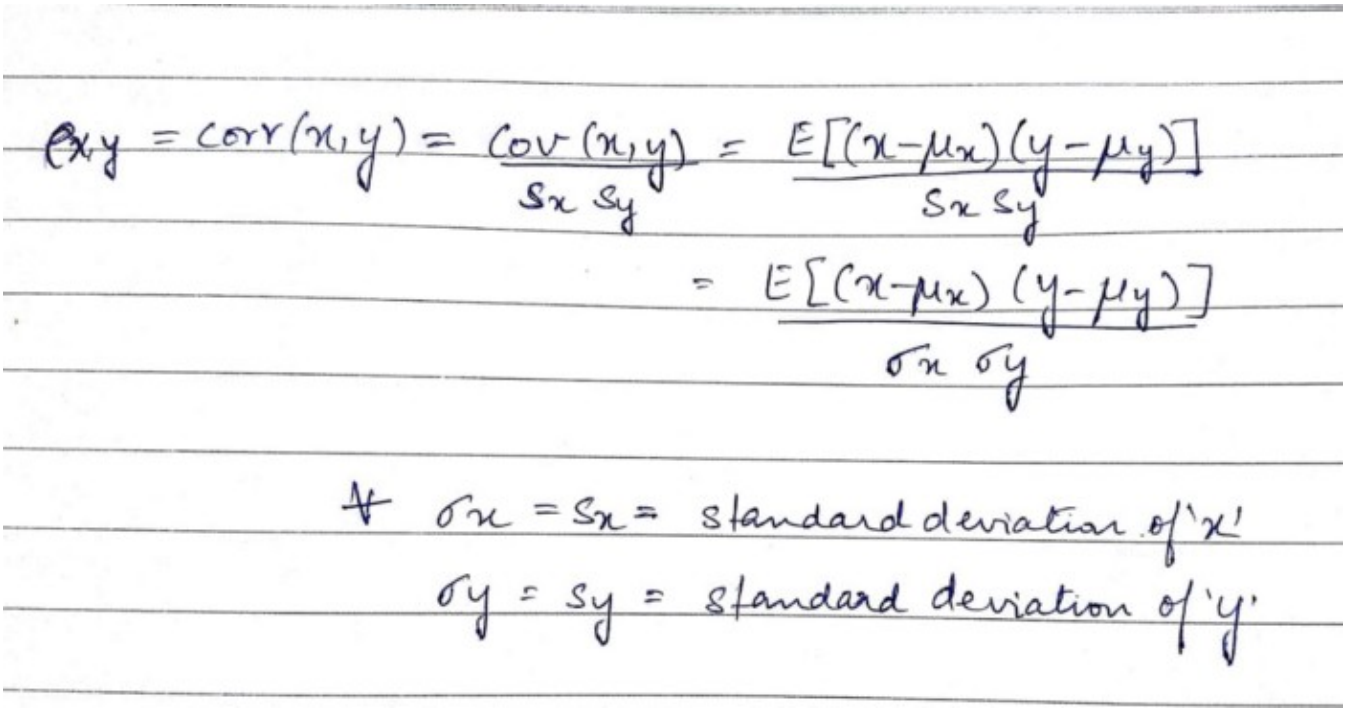
$$cov(x,y) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{n-1} \quad \text{——— (B)}$$

In the above formula, the numerator of the equation(A) is called the sum of squared deviations. In equation(B) with two variables x and y, it is called the sum of cross products. In the above formula, n is the number of samples in the data set. The value (n-1) indicates the degrees of freedom.

> *To explain what degrees of freedom are, let us just take an example. In a set of 3 numbers with the mean as 10 and two out of three variables as 5 and 15, there is only one possibility of the value that the third number can take up i.e. 10. With any set of 3 numbers with the same mean, for example: 12,8 and 10 or say 9,10 and 11, there is only one value for any 2 given values in the set. You can basically change the two values here and the third value fixes itself. The degree of freedom here is 2. Essentially, degrees of freedom is the number of independent data points that went into calculating the estimate. As we see in the example above, it is not necessarily equal to the number of items in the sample (n).*

Correlation

The correlation coefficient is also known as the Pearson product-moment correlation coefficient, or Pearson's correlation coefficient. As mentioned earlier, it is obtained by dividing the covariance of the two variables by the product of their standard deviations. The mathematical representation of the same can be shown in the following manner:

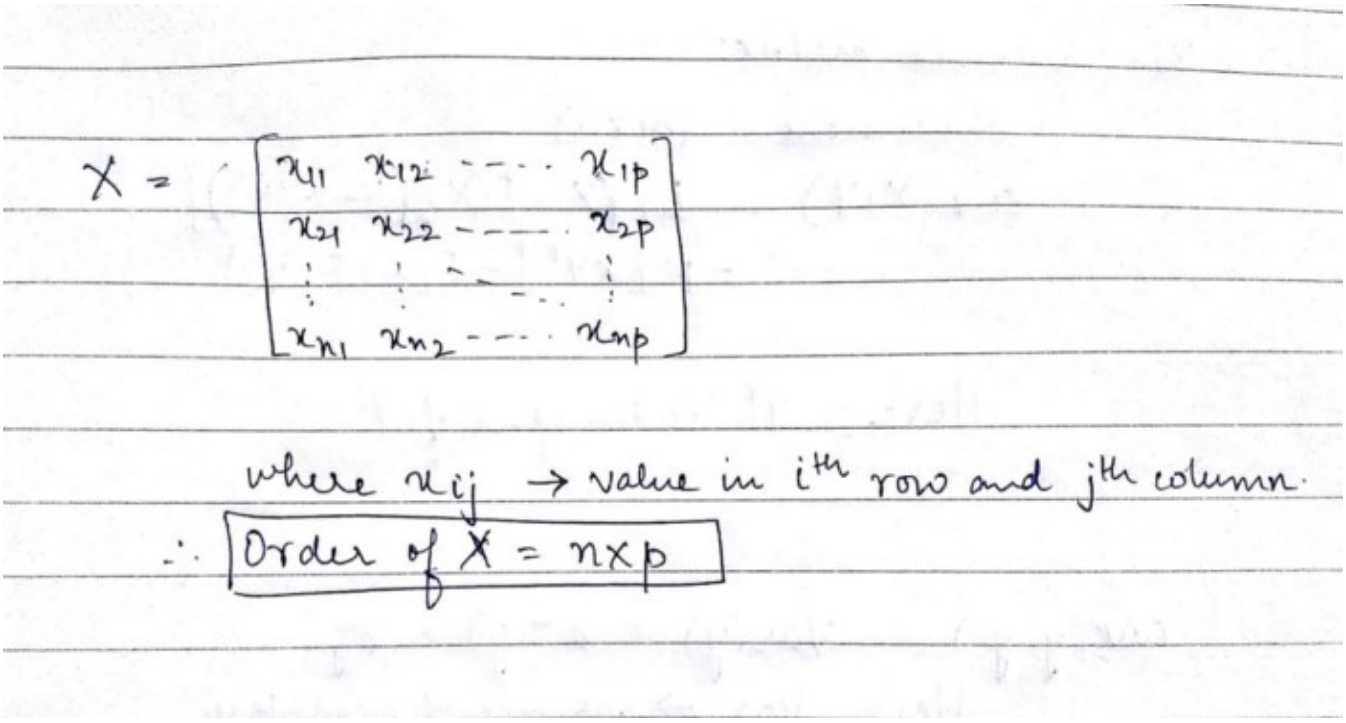


The values of the correlation coefficient can range from -1 to +1. The closer it is to +1 or -1, the more closely are the two variables are related. The positive sign signifies the direction of the correlation i.e. if one of the variables increases, the other variable is also supposed to increase.

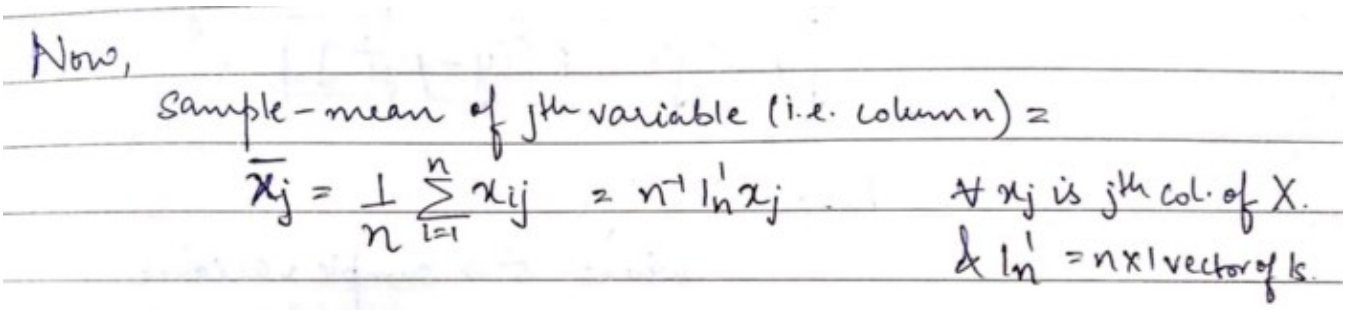
Data-matrix representation of Covariance and Correlation

Let us also delve a little deeper and look at the matrix-representation of covariance.

For a data matrix, X where X can be represented in the following manner:



a vector 'xj' would basically imply a (n × 1) vector extracted from the j-th column of X where j belongs to the set (1,2,….,p). Similarly 'xi`' represents the (1 × p) vector from the i-th row of X. Here 'i' can take a value from the set (1,2,…,n). You can also interpret X as a matrix of variables where 'xij' is the j-th variable (column) collected from the i-th item (row). For ease of reference, let us call rows as items/subjects and columns as variables. Let us now look at the mean of a column of the above data-matrix:



Using the above concept, let us now define the row-mean. It is basically the average of the elements present in the specified row.

Similarly, row-mean:

$$\bar{x_i} = \frac{1}{p}\sum_{j=1}^{p} x_{ij} = p^{-1} x_i' 1_p$$

$\forall\ 1_p = p \times 1$ vector of 1s

& $x_i' = i^{th}$ row of X

Now, that we have the above metrics, it shall be easier to define the covariance matrix (S):

$$S = \begin{bmatrix} s_1^2 & s_{12} & s_{13} & \cdots & s_{1p} \\ s_{21} & s_2^2 & s_{23} & \cdots & s_{2p} \\ s_{31} & s_{32} & s_3^2 & \cdots & s_{3p} \\ \vdots & \vdots & \vdots & & \vdots \\ s_{p1} & s_{p2} & s_{p3} & \cdots & s_p^2 \end{bmatrix}$$

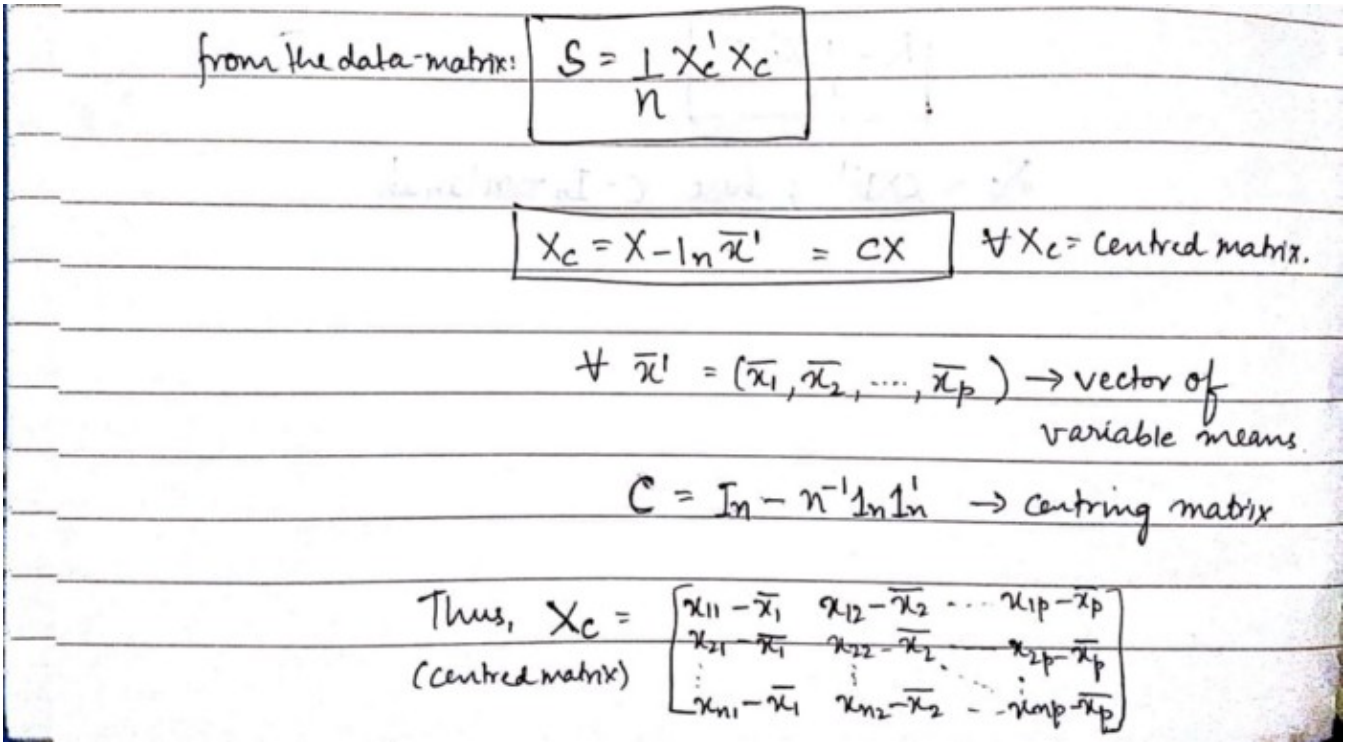
where $s_j^2 = \left(\frac{1}{n}\right)\sum_{i=1}^{n}(x_{ij} - \bar{x_j})^2$: variance of j^{th} variable

$$s_{jk} = (1/n)\sum_{i=1}^{n}(x_{ij} - \bar{x_j})(x_{ik} - \bar{x_k})$$: covariance b/w j^{th} & k^{th} variables

$$\bar{x_j} = (1/n)\sum_{i=1}^{n} x_{ij}$$: mean of j^{th} variable

In the above matrix, we see that the dimension of the covariance matrix is p × p. This is basically a symmetric matrix i.e. a square matrix that is equal to its transpose (S`). The terms building the covariance matrix are called the variances of a given variable, forming the diagonal of the matrix or the covariance of 2 variables filling up the rest of the space. The covariance of the j-th variable with the k-th variable is equivalent to the covariance of the k-th variable with the j-th variable i.e. 'sjk'= 'skj'.

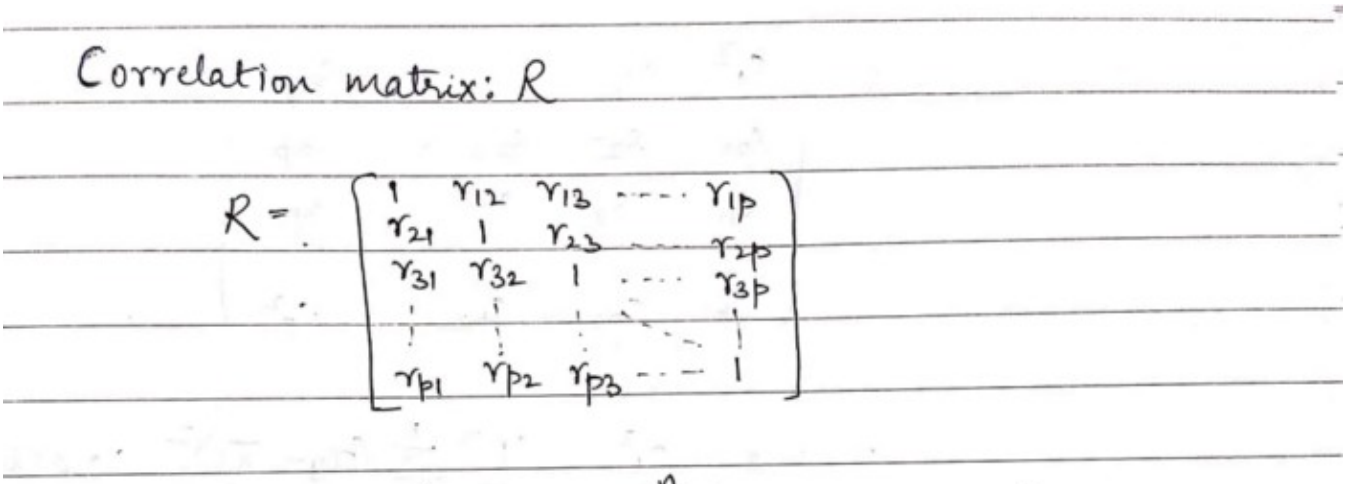
We can create the covariance matrix from the data matrix in the following way:



Here, 'Xc' is a centered-matrix that has the respective column means subtracted from each element. Using that as the central component, the covariance matrix 'S' is the product of the transpose of 'Xc`' and 'Xc' itself, which is then divided by the number of items or rows ('n') in the data-matrix.

> *Before, we move further ahead, let us revisit the concept of sample variance or s-squared (s^2). We can derive the standard deviation of a data-set from this value. Mathematics defines the value 's' as the standard deviation of the data-set. It basically indicates the degree of dispersion or spread of data around the its average.*

Similarly, using the same data-matrix and the covariance matrix, let us define the correlation matrix (R):



$$\forall \quad r_{jk} = \frac{S_{jk}}{S_j S_k} = \frac{\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)(x_{ik}-\bar{x}_k)}{\sqrt{\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2 \sum_{i=1}^{n}(x_{ik}-\bar{x}_k)^2}}$$

r_{jk} = correlation coeff. b/w x_j and x_k

As we see here, the dimension of the correlation matrix is again p × p. Now, if we look at the individual elements of the correlation matrix, the main diagonal all comprises of 1. This indicates that the correlation of an element with itself is 1, or the highest value possible. This makes complete sense logically and intuitively. The other elements 'rjk' is the Pearson's correlation coefficient between two values: 'xj' and 'xk'. As we saw earlier, 'xj' denotes the j-th column of the data-matrix, X. Moving on to how the correlation matrix can be obtained from the data-matrix:

from the Data-Matrix:

$$R = \frac{1}{n} X_S' X_S$$

$X_S = CXD^{-1}$; here: $C = I_n - n^{-1} 1_n 1_n'$

$D = \text{diag}(s_1, s_2, \ldots, s_p)$: a diagonal scaling matrix

$$\therefore X_S = \begin{bmatrix} (x_{11}-\bar{x}_1)/s_1 & (x_{12}-\bar{x}_2)/s_2 & \cdots & (x_{1p}-\bar{x}_p)/s_p \\ (x_{21}-\bar{x}_1)/s_1 & (x_{22}-\bar{x}_2)/s_2 & \cdots & (x_{2p}-\bar{x}_p)/s_p \\ \vdots & \vdots & \ddots & \vdots \\ (x_{n1}-\bar{x}_1)/s_1 & (x_{n2}-\bar{x}_2)/s_2 & \cdots & (x_{np}-\bar{x}_p)/s_p \end{bmatrix}$$

'Xs' in the above definition is called the scaled or the standardized matrix. Here we see that the correlation matrix can be defined as the product of the transpose of the scaled matrix with itself, divided by 'n'. On revisiting the definition of standard deviation from

above, we see that every element (similar to the covariance matrix above) of the standardized matrix 'Xs' is divided by the respective column's standard deviation. This reinforces our understanding of the correlation matrix being a standardized or scaled derivative of the covariance matrix.

(more elaborate working and mathematical operations on the matrices can be found here: http://users.stat.umn.edu/~helwig/notes/datamat-Notes.pdf)

Covariance versus Correlation

As we see from the formula of covariance, it assumes the units from the product of the units of the two variables. On the other hand, correlation is dimensionless. It is a unit-free measure of the relationship between variables. This is because we divide the value of covariance by the product of standard deviations which have the same units. The value of covariance is affected by the change in scale of the variables. If all the values of the given variable are multiplied by a constant and all the values of another variable are multiplied, by a similar or different constant, then the value of covariance also changes. However, on doing the same, the value of correlation is not influenced by the change in scale of the values. Another difference between covariance and correlation is the range of values that they can assume. While correlation coefficients lie between -1 and +1, covariance can take any value between -∞ and +∞.

Application in Analytics

Now that we are done with mathematical theory, let us explore how and where it can be applied in the field of data analytics. Correlation analysis, as a lot of analysts would know is a vital tool for feature selection and multivariate analysis in data preprocessing and exploration. Correlation helps us investigate and establish relationships between variables. This is employed in feature selection before any kind of statistical modelling or data analysis.

PCA or Principal Component Analysis is one significant application of the same. So how do we decide what to use? Correlation matrix or the covariance matrix? In simple words, you are advised to **use the covariance matrix when the variable are on similar scales and the correlation matrix when the scales of the variables differ.**

Now let's understand this with the help of examples. To help you with implementation if needed, I shall be covering examples in both R and Python. Let us see the first example where we see how PCA results differ when computed with the correlation

matrix and the covariance matrix respectively. **For the first example here, we will consider the 'mtcars' data-set in R.**

```r
# Loading dataset in local R environment
data(mtcars)
# Print the first 10 rows of the dataset
head(mtcars, 10)
```

mtcars_load.R hosted with ❤ by GitHub view raw

```
> head(mtcars, 10)
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

From the above image, we see that all the columns are numerical and hence, we can go ahead with the analysis. We shall use the prcomp() function from the 'stats' package for the same.

PCA with covariance matrix

We will first conduct the PCA with the covariance matrix. For that, we set the 'scale' option as 'FALSE':

```r
# Setting the scale=FALSE option will use the covariance matrix to get the PCs
cars.PC.cov = prcomp(mtcars[,-11], scale=FALSE) # sellecting the 11 columns by indexing
```

mtcars_cov_PCA.R hosted with ❤ by GitHub view raw

Name	Type	Value
cars.PC.cov	list [5] (S3: prcomp)	List of length 5
sdev	double [10]	136.531 38.133 3.056 1.279 0.694 0.378 ...
rotation	double [10 x 10]	-0.038118 0.012035 0.899589 0.434779 -0.002660 0.006240 0.009178 -0.003374 ...
center	double [10]	20.09 6.19 230.72 146.69 3.60 3.22 ...
scale	logical	FALSE
x	double [32 x 10]	-7.96e+01 -7.96e+01 -1.34e+02 8.53e+00 1.29e+02 -2.32e+01 2.18e+00 2...

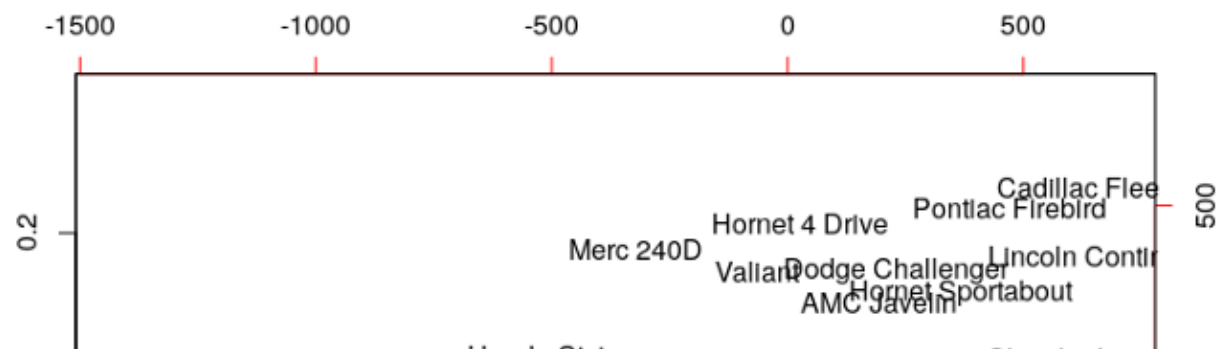
Here, cars.PC.cov is the result of the Principal Component Analysis on the mtcars data-set using the covariance matrix. So, prcomp() returns 5 key measures: sdev, rotation, center, scale and x. Let us briefly go through all the measures here:

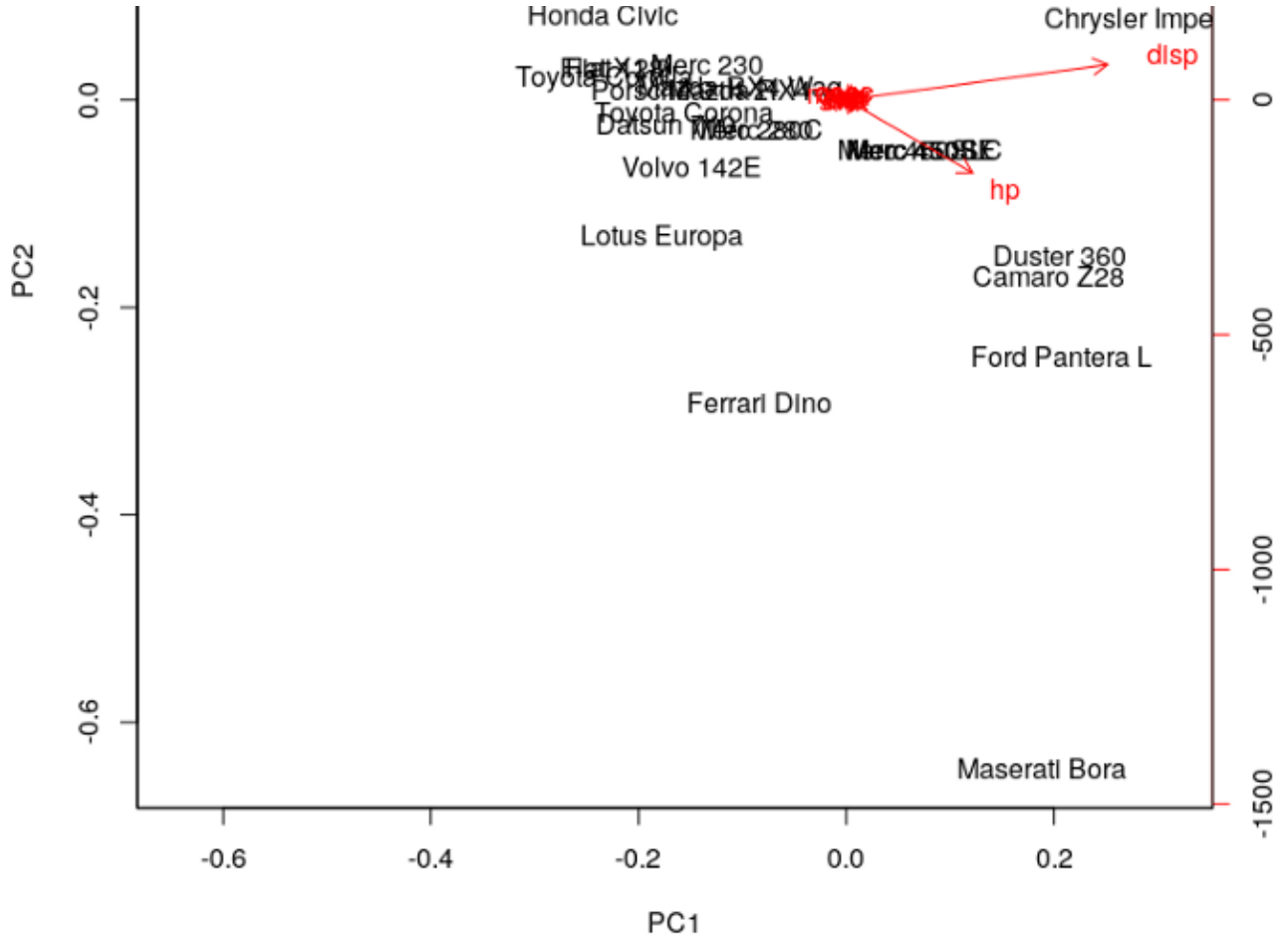
1. The *center* and *scale* provide the respective means and standard deviation of the variables that were used for normalization before implementing PCA.
2. *sdev* shows the standard deviation of principal components. In other words, it shows the square roots of the eigenvalues.
3. The *rotation* matrix contains the principal component loading. This is the most important result of the function. Each column of the rotation matrix contains the principal component loading vector. The component loading can be represented as the correlation of a particular variable on the respective PC(principal component). It can assume both positive or negative. Higher the loading value, higher is the correlation. With this information of component loading, you can easily interpret the 'key variable' in the PC.
4. the matrix *x* has the principal component score vectors.

Now, if you see the *scale* measure in the list of result 'cars.PC.cov', we see it is set as FALSE as specified by us in the input arguments. Let us now look at the principal component loading vectors:

```
Rotation (n x k) = (10 x 10):
             PC1         PC2         PC3          PC4          PC5         PC6          PC7         PC8          PC9         PC10
mpg  -0.3672381  0.01530638 -0.30848700  0.001284545 -0.005948307  0.44699836 -0.669888354  0.01103342  0.03960931  0.34561836
cyl   0.3790139 -0.10981644 -0.14363013 -0.009098330  0.181288155  0.03929748 -0.139644638 -0.11113166  0.85986903 -0.13394430
disp  0.3789329 -0.01210717  0.10192205 -0.154961862 -0.072150842  0.46484868 -0.250188807  0.38330827 -0.24024723 -0.57735049
hp    0.3210587 -0.30547412  0.35036433  0.188044745  0.326867091  0.28845701 -0.088053415 -0.58822934 -0.26475606  0.17770321
drat -0.3148186 -0.24709638  0.30051940 -0.817556496  0.266783043 -0.02250972 -0.084950285 -0.03602607  0.08239551 -0.03648035
wt    0.3563205  0.13231424  0.42095135 -0.192484984 -0.441213799  0.11227212  0.012636346  0.23939134  0.14199143  0.59697069
qsec -0.1823700  0.55702222  0.27439872 -0.066364455 -0.365590711  0.07491028 -0.127896911 -0.54884095  0.12851715 -0.32155833
vs   -0.3024000  0.32005883  0.41920364  0.322330480  0.466485972  0.33250985  0.224890149  0.31862307  0.21434487  0.01661447
am   -0.2597800 -0.44834501 -0.09242449  0.069764748 -0.426471262  0.50799168  0.485424101 -0.09920299  0.16311735 -0.07793174
gear -0.2378036 -0.44939593  0.47326769  0.349176422 -0.236780718 -0.33468615 -0.398305392  0.15295312  0.12338396 -0.17389906
```

To help with the interpretation, let us plot these results.





To read this chart, one has to look at the extreme ends (top, down, left and right). The first principal component here (on the left and right) corresponds to the measure of ‘disp’ and ‘hp’. The second principal component (PC2) does not seem to have a strong measure.

We can finish this analysis with a summary of the PCA with the covariance matrix:

```
1  summary(cars.PC.cov)
```

mtcars_cov_PCA_summ.R hosted with ❤ by **GitHub** **view raw**

```
> summary(cars.PC.cov)
Importance of components:
                          PC1      PC2     PC3     PC4     PC5     PC6   PC7    PC8    PC9  PC10
Standard deviation     136.5308 38.13336 3.05607 1.27948 0.69381 0.37784 0.286 0.2795 0.2388 0.199
Proportion of Variance   0.9271  0.07232 0.00046 0.00008 0.00002 0.00001 0.000 0.0000 0.0000 0.000
Cumulative Proportion    0.9271  0.99941 0.99987 0.99996 0.99998 0.99999 1.000 1.0000 1.0000 1.000
```

From this table, we see that the maximum contribution to variation caused is caused by PC1 (~92.7%) and all other principal components have progressively lower contribution. In simpler terms, it means that almost 93% of the variance in the data-set can be explained using the first principal component with measures of ‘disp’ and ‘hp’.

This is in line with our observations from the rotation matrix and the plot above. As a conclusion, not a lot of significant insights can be driven from the Principal Component Analysis on the basis of the covariance matrix.

PCA with correlation matrix

Now, let us shift our focus to PCA with the correlation matrix. For this, all we need to do is, set the 'scale' argument as TRUE.

```
# scale=TRUE performs the PCA using the correlation matrix
cars.PC.cor = prcomp(mtcars[,-11], scale=TRUE)
```

mtcars_cor_PCA.R hosted with ❤ by **GitHub** **view raw**

Name	Type	Value
cars.PC.cor	list [5] (S3: prcomp)	List of length 5
sdev	double [10]	2.520 1.493 0.673 0.516 0.463 0.420 ...
rotation	double [10 x 10]	-0.36724 0.37901 0.37893 0.32106 -0.31482 0.35632 0.01531 -0.10982 -0.01211 ...
center	double [10]	20.09 6.19 230.72 146.69 3.60 3.22 ...
scale	double [10]	6.027 1.786 123.939 68.563 0.535 0.978 ...
mpg	double [1]	6.026948
cyl	double [1]	1.785922
disp	double [1]	123.9387
hp	double [1]	68.56287
drat	double [1]	0.5346787
wt	double [1]	0.9784574
qsec	double [1]	1.786943
vs	double [1]	0.5040161
am	double [1]	0.4989909
gear	double [1]	0.7378041
x	double [32 x 10]	-0.884999 -0.849289 -2.556929 0.028562 2.121438 0.303364 -1.471339 -1.262294 ...

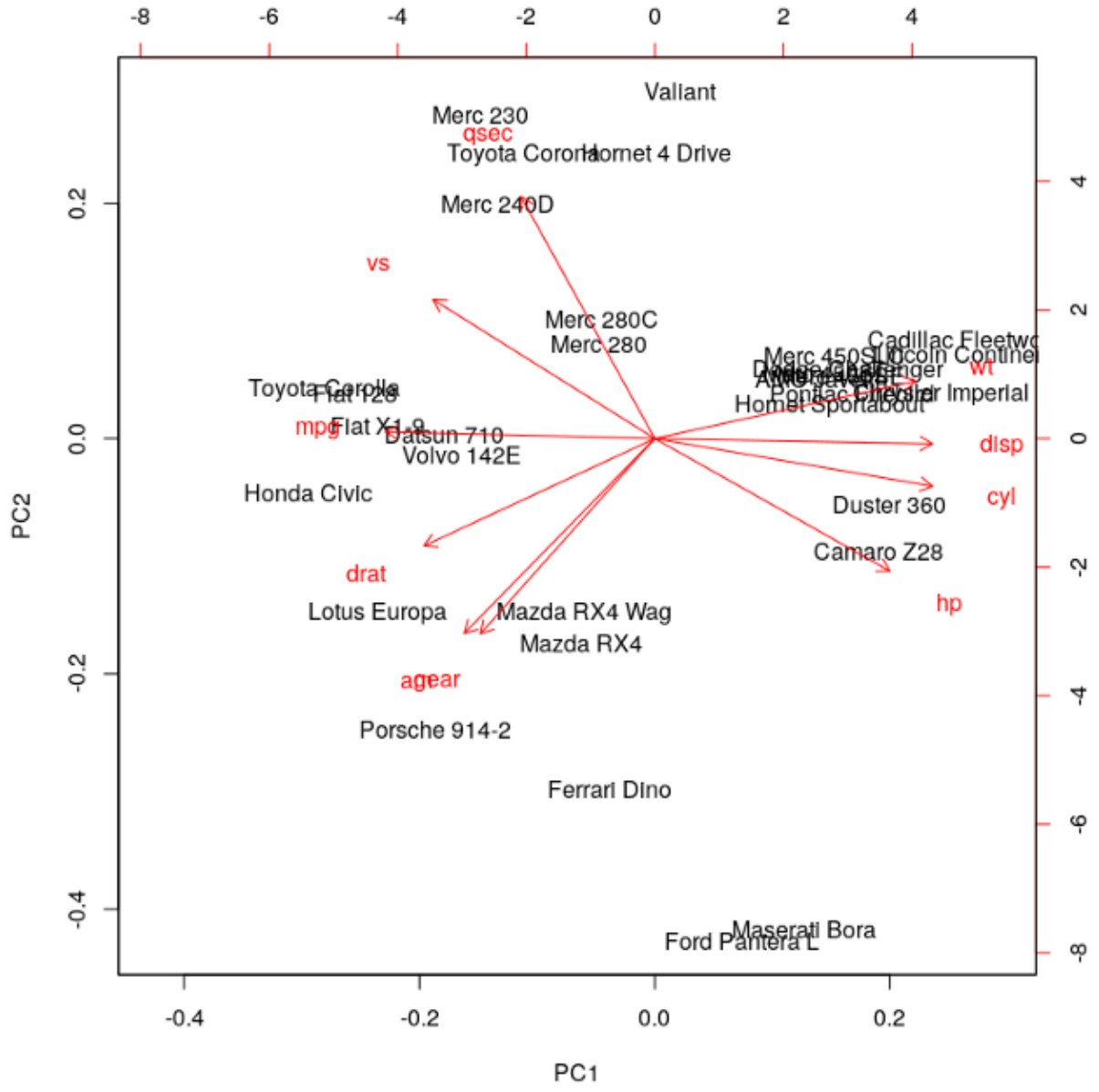
With the same definitions of all the measures above, we now see that the *scale* measure has values corresponding to each variable. The rotation matrix can be observed in a similar way along with the plot.

```
# obtain the rootation matrix
cars.PC.cor$rotation
# obtain the biplot
biplot(cars.PC.cor)
```

mtcars_cor_rotation.R hosted with ❤ by **GitHub** **view raw**

```
Rotation (n x k) = (10 x 10):
            PC1         PC2         PC3          PC4          PC5         PC6          PC7         PC8         PC9         PC10
mpg  -0.3672381  0.01530638 -0.30848780  0.001284545 -0.005948307  0.44699836 -0.669888354  0.01103342  0.03960931  0.34561836
cyl   0.3790139 -0.10981644 -0.14363013 -0.009098330  0.181288155  0.03929748 -0.139644638 -0.11113166  0.85986903 -0.13394430
disp  0.3789329 -0.01210717  0.10192285 -0.154961862 -0.072150842  0.46484868 -0.258188807  0.38330827 -0.24024723 -0.57735049
```

```
hp    0.3210587 -0.30547412  0.35036433  0.188044745  0.326867091  0.28845701 -0.888053415 -0.58822934 -0.26475606  0.17770321
drat -0.3148186 -0.24709638  0.30051940 -0.817556496  0.266783043 -0.02250972 -0.804950285 -0.03602607  0.08239551 -0.03648035
wt    0.3563205  0.13231424  0.42095135 -0.192484984 -0.441213799  0.11227212  0.012636346  0.23939134  0.14199143  0.59697069
qsec -0.1823700  0.55702222  0.27439872 -0.066364455 -0.365590711  0.07491028 -0.127896911 -0.54884095  0.12851715 -0.32155833
vs   -0.3024000  0.32005883  0.41920364  0.322330480  0.466485972  0.33250985  0.224890149  0.31862307  0.21434487  0.01661447
am   -0.2597800 -0.44834501 -0.09242449  0.069764748 -0.426471262  0.50799168  0.485424101 -0.09920299  0.16311735 -0.07793174
gear -0.2378036 -0.44939593  0.47326769  0.349176422 -0.236780718 -0.33468615 -0.398305392  0.15295312  0.12338396 -0.17389906
```



This plot looks more informative. It says that variables like ‘disp’, ‘cyl’, ‘hp’, ‘wt’, ‘mpg’, ‘drat’ and ‘vs’ contribute significantly to PC1 (first principal component). ‘qsec, ‘gear’ and ‘am’ have a significant contribution to PC2. Let us try to look at the summary of this analysis.

```
> summary(cars.PC.cor)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
Standard deviation     2.5196 1.4928 0.67327 0.51595 0.46293 0.42045 0.36485 0.28878 0.23287 0.20472
Proportion of Variance 0.6348 0.2228 0.04533 0.02662 0.02143 0.01768 0.01331 0.00834 0.00542 0.00419
Cumulative Proportion  0.6348 0.8577 0.90301 0.92963 0.95106 0.96874 0.98205 0.99039 0.99581 1.00000
```

One significant change we see is the drop in the contribution of PC1 to the total variation. It has dropped from 92.7% to 63.5%. On the other hand, the contribution of PC2 has increased from 7% to 22%. Furthermore, the component loading values show that the relationship between the variables in the data-set is way more structured and distributed. Another significant difference can be observed if you look at the standard deviation values in both the results above. The values from PCA done using the correlation matrix are closer to each other and more uniform as compared to the analysis done using the covariance matrix.

This analysis with the correlation matrix definitely, uncovers some better structure in the data and relationships between variables. The above example can be used to conclude that the results significantly differ when one tries to define variable relationships using covariance and correlation. This in turn, affects the importance of the variables computed for any further analyses. Selection of predictors and independent variables is one prominent application of such exercises.

Now, let us take another example to check if standardizing the data-set before performing PCA actually gives us the same results. To showcase agility of implementation across technologies, I shall execute this example in Python. **We will consider the 'iris' data-set for the same.**

```
from sklearn import datasets
import pandas as pd

# load iris dataset
iris = datasets.load_iris()
# Since this is a bunch, create a dataframe
iris_df=pd.DataFrame(iris.data)
iris_df['class']=iris.target

iris_df.columns=['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid', 'class']
iris_df.dropna(how="all", inplace=True) # remove any empty lines

#selecting only first 4 columns as they are the independent(X) variable
# any kind of feature selection or correlation analysis should be first done on these
iris_X=iris_df.iloc[:,[0,1,2,3]]
```

This data-set will now be standardized using the inbuilt function.

```
from sklearn.preprocessing import StandardScale
# let us now standardize the dataset
iris_X_std = StandardScaler().fit_transform(iris_X)
```

scale_iris_X.py hosted with ❤ by **GitHub** **view raw**

I have then computed 3 matrices:

1. Covariance matrix on standardized data
2. Correlation matrix on standardized data
3. Correlation matrix on unstandardized data

```
# covariance matrix on standardized data
mean_vec = np.mean(iris_X_std, axis=0)
cov_matrix = (iris_X_std - mean_vec).T.dot((iris_X_std - mean_vec)) / (iris_X_std.shape
print('Covariance matrix \n%s' %cov_matrix)

# correlation matrix:
cor_matrix = np.corrcoef(iris_X_std.T)
print('Correlation matrix using standardized data\n%s' %cor_matrix)


cor_matrix2 = np.corrcoef(iris_X.T)
print('Correlation matrix using base unstandardized data \n%s' %cor_matrix2)
```

corr_cov_iris.py hosted with ❤ by **GitHub** **view raw**

Let us look at the results:

```
Correlation matrix using standardized data
[[ 1.         -0.10936925  0.87175416  0.81795363]
 [-0.10936925  1.         -0.4205161  -0.35654409]
 [ 0.87175416 -0.4205161   1.          0.9627571 ]
 [ 0.81795363 -0.35654409  0.9627571   1.        ]]
Correlation matrix using base unstandardized data
[[ 1.         -0.10936925  0.87175416  0.81795363]
 [-0.10936925  1.         -0.4205161  -0.35654409]
 [ 0.87175416 -0.4205161   1.          0.9627571 ]
 [ 0.81795363 -0.35654409  0.9627571   1.        ]]
```

```
Covariance matrix
[[ 1.00671141 -0.11010327  0.87760486  0.82344326]
 [-0.11010327  1.00671141 -0.42333835 -0.358937  ]
 [ 0.87760486 -0.42333835  1.00671141  0.96921855]
 [ 0.82344326 -0.358937    0.96921855  1.00671141]]
```

Here, it looks like the results are similar. The similarity of results (fractional differences) reinforces the understanding that correlation matrix is just a scaled derivative of the covariance matrix. Any computation on these matrices should now yield the same or similar results.

To perform a PCA on these matrices from scratch, we will have to compute the eigenvectors and eigenvalues:

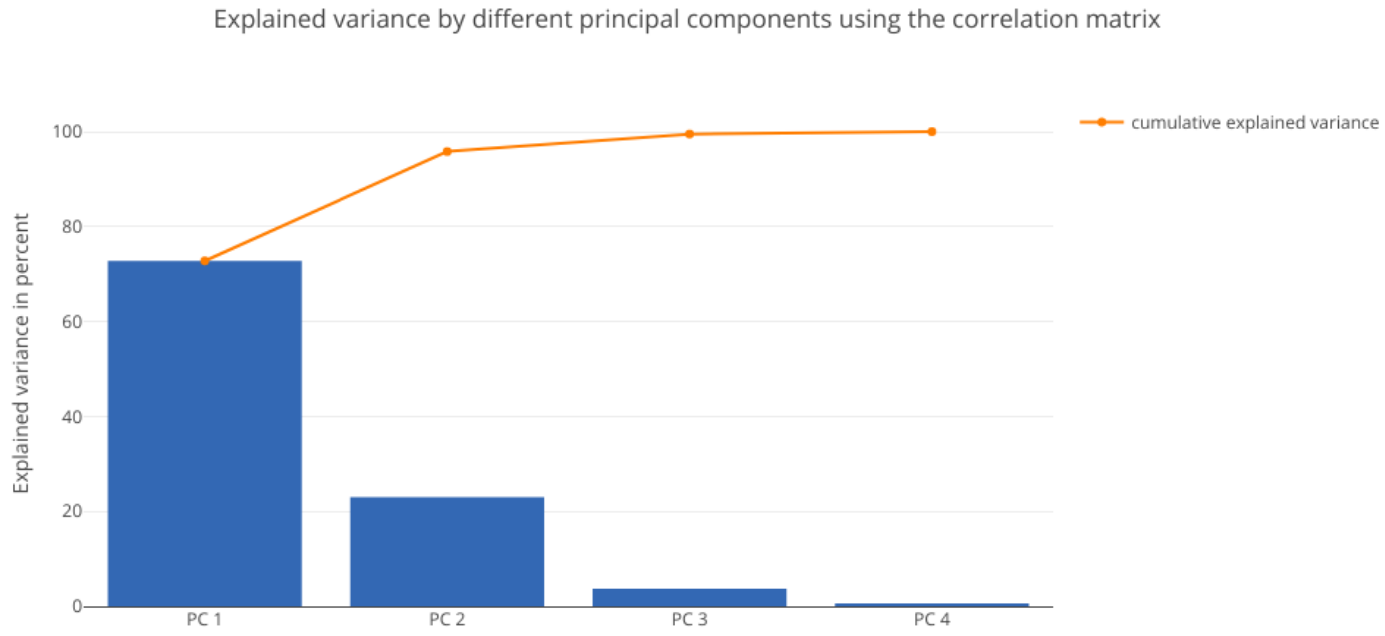
```
eig_vals1, eig_vecs1 = np.linalg.eig(cov_matrix)
print('Eigenvectors using covariance matrix \n%s' %eig_vecs1)
print('\nEigenvalues using covariance matrix \n%s' %eig_vals1)

eig_vals2, eig_vecs2 = np.linalg.eig(cor_matrix)
print('Eigenvectors using correlation matrix of standardized data \n%s' %eig_vecs2)
print('\nEigenvalues using correlation matrix of standardized data \n%s' %eig_vals2)

eig_vals3, eig_vecs3 = np.linalg.eig(cor_matrix2)
print('Eigenvectors using correlation matrix of unstandardized data \n%s' %eig_vecs3)
print('\nEigenvalues using correlation matrix of unstandardized data \n%s' %eig_vals3)
```

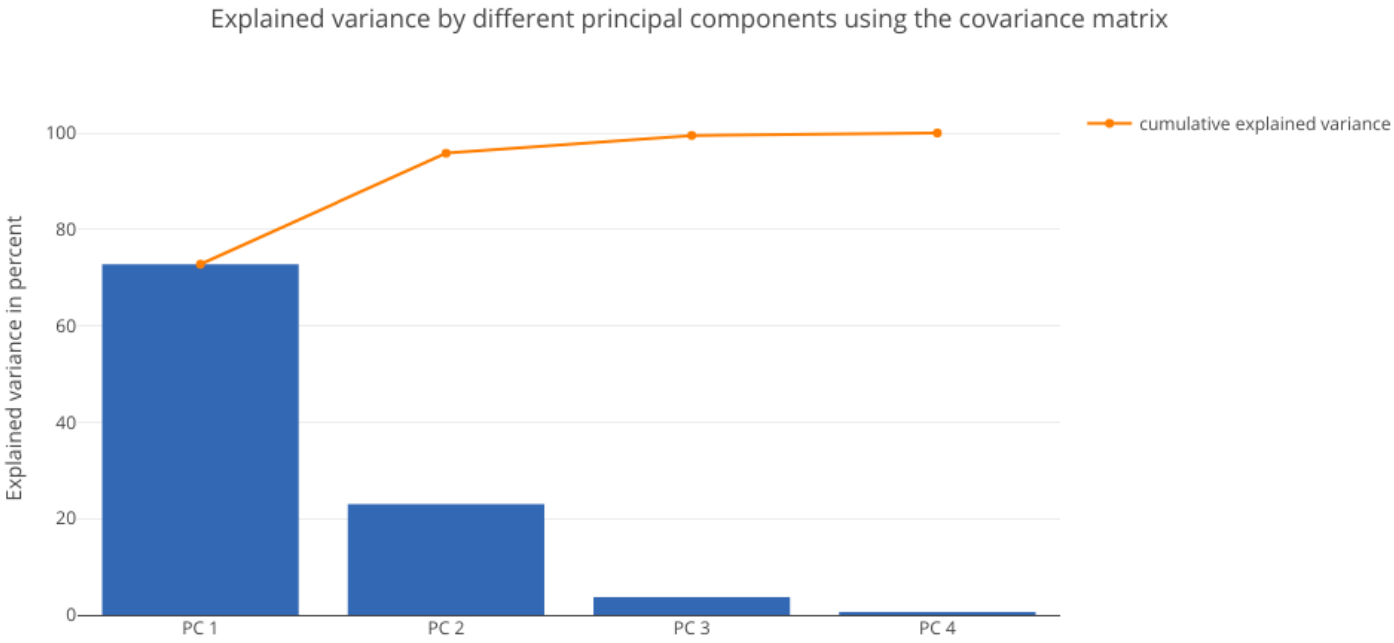
```
Eigenvectors using covariance matrix
[[ 0.52237162 -0.37231836 -0.72101681  0.26199559]
 [-0.26335492 -0.92555649  0.24203288 -0.12413481]
 [ 0.58125401 -0.02109478  0.14089226 -0.80115427]
 [ 0.56561105 -0.06541577  0.6338014   0.52354627]]

Eigenvalues using covariance matrix
[2.93035378 0.92740362 0.14834223 0.02074601]
Eigenvectors using correlation matrix of standardized data
[[ 0.52237162 -0.37231836 -0.72101681  0.26199559]
 [-0.26335492 -0.92555649  0.24203288 -0.12413481]
 [ 0.58125401 -0.02109478  0.14089226 -0.80115427]
 [ 0.56561105 -0.06541577  0.6338014   0.52354627]]

Eigenvalues using correlation matrix of standardized data
[2.91081808 0.92122093 0.14735328 0.02060771]
Eigenvectors using correlation matrix of unstandardized data
[[ 0.52237162 -0.37231836 -0.72101681  0.26199559]
 [-0.26335492 -0.92555649  0.24203288 -0.12413481]
 [ 0.58125401 -0.02109478  0.14089226 -0.80115427]
 [ 0.56561105 -0.06541577  0.6338014   0.52354627]]

Eigenvalues using correlation matrix of unstandardized data
[2.91081808 0.92122093 0.14735328 0.02060771]
```

As expected, the results from all three relationship matrices are the same. We shall now plot the explained variance charts from the eigenvalues so obtained:

```
# PLOTTING THE EXPLAINED VARIANCE CHARTS

###################################
#USING THE COVARIANCE MATRIX
###################################
tot = sum(eig_vals1)
explained_variance1 = [(i / tot)*100 for i in sorted(eig_vals1, reverse=True)]
cum_explained_variance1 = np.cumsum(explained_variance1)

```

```
trace3 = Bar(
        x=['PC %s' %i for i in range(1,5)],
        y=explained_variance1,
        showlegend=False)

trace4 = Scatter(
        x=['PC %s' %i for i in range(1,5)],
        y=cum_explained_variance1,
        name='cumulative explained variance')

data2 = Data([trace3, trace4])

layout2=Layout(
        yaxis=YAxis(title='Explained variance in percent'),
        title='Explained variance by different principal components using the covarianc

fig2 = Figure(data=data2, layout=layout2)
py.iplot(fig2)

###################################
#USING CORRELATION MATRIX
###################################
tot = sum(eig_vals3)
explained_variance = [(i / tot)*100 for i in sorted(eig_vals3, reverse=True)]
cum_explained_variance = np.cumsum(explained_variance)

trace1 = Bar(
        x=['PC %s' %i for i in range(1,5)],
        y=explained_variance,
        showlegend=False)

trace2 = Scatter(
        x=['PC %s' %i for i in range(1,5)],
        y=cum_explained_variance,
        name='cumulative explained variance')

data = Data([trace1, trace2])

layout=Layout(
        yaxis=YAxis(title='Explained variance in percent'),
        title='Explained variance by different principal components using the correlati

fig = Figure(data=data, layout=layout)
py.iplot(fig)
```





The charts resemble each other. Both the charts show that PC1 has the maximum contribution of around 71%. This analysis establishes the fact that standardizing the data-set and then computing the covariance and correlation matrices will yield the same results.

One can use this learning to judiciously apply the concept of variable relationships before using any predictive algorithm.

Data Science Mathematics Pca Python R