CHAPTER 3

Computing Principal Components with R

OBJECTIVE

The purpose of this chapter is to show you step by step how to perform Principal Component Analysis (PCA) using the R software. I will start with an R built-in dataset named `MTcars`, and use specific R functions to perform PCA. The approach adopted here is practical and allows you to learn as you do. New concepts are introduced in a given context as they are needed. This presentation is not mathematical. The only basic mathematical concepts used throughout this chapter are those of "Coordinate Systems" and "Basis Vectors." If you ever decide to take your study of principal components to the next level, I advise that you read the free eBook entitled "Matrix Algebra for Educational Scientists," which you can access with the link `https://zief0002.github.io/matrix-algebra/index.html`.

Contents

3.1 Introduction

R offers 2 functions from the built-in `stats` package, which you can use to perform Principal Component Analysis (PCA)[1]. Their simplified syntaxes are the following:

```
prcomp(x, center = TRUE, scale = FALSE)
princomp(x, cor = FALSE, scores = TRUE)
```

where the different parameters are defined as follows:

- `x`: A numeric matrix (or data frame) which provides the data for principal components analysis.
- `center`: A logical value indicating whether the variables should be zero centered. Alternatively, this argument can be a vector containing the same number of elements as the number of columns in `x`. In this case, each column of `x` will be shifted by a magnitude determined by the corresponding element of vector "`center`." I always use the default value `TRUE` as argument. This choice will likely meet the requirements of most analysis projects.
- `cor`: A logical value indicating whether PCA will be based on the correlation matrix or not. If not, then the covariance matrix calculated from matrix `x`'s columns will be used. I always set this parameter to "`TRUE`." This option is safer if the input variables are on different scales.
- `scale`: A logical value indicating whether the variables should be scaled or standardized to a unit variance before being analyzed. The default is `FALSE` for consistency with S, but in general scaling your data is recommended to control the unduly large influence of high-variability variables.
- `scores`: A logical value indicating whether or not principal component scores should be calculated. I always use the default value "`TRUE`."

Either function will produce a valid PCA. However, there are 2 reasons why the `prcomp()` function is the more popular one. First, it is based on an algorithm named "Singular Value Decomposition" (SVD), which allows you to analyze all

[1]Since both functions are provided by the built-in package `stats`, they are readily available as soon as you launch R.

datasets, including those where the number of variables exceeds the number of records[2]. Second, this algorithm is expected to have a slightly better numerical accuracy than the one the `princomp()` function is based on. I find this argument disputable. But I will come back to this issue later in section 3.3.

> *Whenever you perform a PCA task using one of the 2 functions `prcomp()` or `princomp()`, I recommend that you set all logical arguments to TRUE.*

Setting all logical arguments of the PCA functions to `TRUE` will ensure that your input data will always be centered and standardized (or scaled). Centering transforms each variable to shift its mean to 0, whereas scaling or standardization gives each variable the same standard deviation of 1.

The underlying mathematics of PCA require that the data be centered to properly quantify the amount of information contained in your dataset. Therefore, data centering is mandatory when doing PCA. But standardizing is not. However, not standardizing your data will make it more difficult to interpret the PCA results, since variables with more variability will dominate the analysis making the other variables less relevant. The `princomp()` function automatically centers your input data. Although `prcomp()` will center your data by default, you are given the option to analyze your raw un-centered data by setting its `center=` parameter to `FALSE`. Both functions allow you to decide whether you want to scale your data or not, although both will not scale your data by default. To scale your data, set `scale = TRUE` in function `prcomp()`, and set `cor = TRUE` with the `princomp()` function.

3.2 Calculating the Principal Components

The primary objective of this section is to show you by example, how to perform PCA using the R software. Then you will explore R's PCA output, and learn how it should be interpreted. What I want you to retain from this section is that PCA is about transforming your dataset that contains a set of initial variables, into a new dataset of synthetic variables also known as principal component scores (or PC scores). Both datasets will have the same number of variables and contain the same amount of information. However, most of that

[2]Although this situation is uncommon in practice, you should note that the `princomp()` function requires the number of records to be higher than the number of variables.

information will be concentrated in the first few synthetic variables of the new dataset. Therefore, your analysis can be based on 2 or 3 synthetic variables with limited loss of information. This is how your problem's dimensionality will be reduced.

3.2.1 *Problem: Analysis of the MTcars Dataset*

Consider the `mtcars` dataset, built in R and described in section 1.5. It is a small dataset of 32 car models. For each car model, this dataset contains various information about the car design and performance. Here are the first 6 records:

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

You may type the command `>?mtcars` straight on RStudio's console to see how these variables are defined. Alternatively, you can read section 1.5 to obtain the same information.

A question of interest you may ask about this dataset is whether cars showing some similarity with respect to these characteristics can be organized in groups or clusters. You may also want to know whether this dataset contains an outlier. An outlier in this context, is a car that strays away from other cars with respect to the different measurements contained in the dataset. A visual inspection of this multidimensional dataset cannot help us, our naked eyes being able to handle 2 or 3 dimensions only.

I will now show you how to address these 2 issues ("Clustering" and "Outlier Detection") with principal components. The first step is to run the following small script:

```
mtcars.df <- mtcars[,c(1:7,10,11)]
mtcars.pca <- prcomp(mtcars.df, center = TRUE, scale = TRUE)
str(mtcars.pca)
```

In line #01, I create a data frame named mtcars.df that contains only quantitative measurements, excluding the 2 categorical variables vs and am. In this line, you tell R to select columns 1 through 7, and columns 10 and 11. You need to know the variables these column numbers are associated with.

Line #02 is where I perform the PCA by calling function prcomp(), and setting the 2 parameters center= and scale= to TRUE. Therefore, all variables in the input dataset will be centered and standardized.

The prcomp() function of line #02 produces a 5-element list object named mtcars.pca. You can explore its structure by calling the str() function as shown in line #03. This produces the following structure:

```
> str(mtcars.pca)
List of 5
 $ sdev    : num [1:9] 2.378 1.443 0.71 0.515 0.428 ...
 $ rotation: num [1:9, 1:9] -0.393 0.403 0.397 0.367 -0.312 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:9] "mpg" "cyl" "disp" "hp" ...
  .. ..$ : chr [1:9] "PC1" "PC2" "PC3" "PC4" ...
 $ center  : Named num [1:9] 20.09 6.19 230.72 146.69 3.6 ...
  ..- attr(*, "names")= chr [1:9] "mpg" "cyl" "disp" "hp" ...
 $ scale   : Named num [1:9] 6.027 1.786 123.939 68.563 0.535 ...
  ..- attr(*, "names")= chr [1:9] "mpg" "cyl" "disp" "hp" ...
 $ x       : num [1:32, 1:9] -0.664 -0.637 -2.3 -0.215 1.587 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" ...
  .. ..$ : chr [1:9] "PC1" "PC2" "PC3" "PC4" ...
 - attr(*, "class")= chr "prcomp"
```

The 5 elements included in the mtcars.pca list are sdev, rotation, center, scale, and x. This is the order in which they appear on the list. However, I found it more logical to discuss these elements in the following order: center, scale, rotation, x and sdev.

- **center:** This is a vector containing the mean values of all numeric variables in your dataset[3]. For the above example, these mean values can be displayed by typing mtcars.pca$center. This yields the following values:

[3]Centering amounts to calculating $y_i - \overline{y}$ for each record i of each variable y, where $\overline{y}$ is the average y value.

```
    mpg      cyl      disp        hp    drat      wt     qsec
20.0906 6.1875 230.7219 146.6875 3.5966 3.2173  17.8488
   gear    carb
3.6875 2.8125
```

- **`scale`:** This list element is a vector containing the standard deviations of all numeric variables in your dataset[4]. For the `mtcars` example, these standard deviations can be accessed by typing `mtcars.pca$scale`, and are given by:

```
     mpg      cyl      disp        hp     drat       wt     qsec
6.02695 1.78592 123.93869 68.56287 0.53468 0.97846 1.78694
    gear     carb
0.73780 1.61520
```

  If you want to analyze data that are unscaled or non-standardized, which I do not recommend, then you must set the `scale` parameter to `FALSE` (i.e. `scale=FALSE`) when calling the `prcomp()` function. In this case, the vector `mtcars.pca$scale` will contain the logical value `FALSE`.

- **`rotation`:** This list element is a matrix containing all principal components also known as Principal Component coefficients (PC coefficients). You will access its content by typing `mtcars.pca$rotation`. The naming convention used here can be confusing. The term "rotation" is used here primarily because principal components are often regarded as a new coordinate system that is obtained by rotating the original Cartesian coordinate system (see Figure 1.2 of chapter #1). The problem with this terminilogy stems from the fact that the term rotation also refers to a special technique used in PCA to make principal components easier to interpret as discussed in (Gwet, 2020, section 3.4).

  Here is the matrix of principal components:

```
> round(mtcars.pca$rotation,4)
         PC1     PC2     PC3     PC4     PC5     PC6     PC7
mpg  -0.3931  0.0275 -0.2212 -0.0061 -0.3208  0.7202 -0.3814
cyl   0.4026  0.0157 -0.2523  0.0407  0.1171  0.2243 -0.1589
```

[4] Centering and scaling amounts to calculating $(y_i - \overline{y})/s_{\mathtt{y}}$ for each record i of each variable `y`, and $s_{\mathtt{x}}$ is the standard deviation of variable `x`.

```
disp  0.3974 -0.0889 -0.0783  0.3395 -0.4868 -0.0197 -0.1823
hp    0.3671  0.2694 -0.0172  0.0683 -0.2947  0.3539  0.6962
drat -0.3118  0.3417  0.1500  0.8457  0.1619 -0.0154  0.0477
wt    0.3735 -0.1719  0.4537  0.1913 -0.1875 -0.0838 -0.4278
qsec -0.2244 -0.4840  0.6281 -0.0303 -0.1482  0.2575  0.2762
gear -0.2095  0.5508  0.2066 -0.2824 -0.5625 -0.3230 -0.0856
carb  0.2446  0.4843  0.4641 -0.2145  0.3998  0.3571 -0.2060
         PC8     PC9
mpg  -0.1247  0.1149
cyl   0.8103  0.1627
disp -0.0642 -0.6619
hp   -0.1657  0.2518
drat  0.1351  0.0381
wt   -0.1984  0.5692
qsec  0.3561 -0.1687
gear  0.3164  0.0472
carb -0.1083 -0.3205
```

What do these numbers stand for? What are they used for? Remember that principal components are vectors that form the basis of a new corrdinate system designed to help you look at your dataset from different angles that display the data variation more effectively. These coefficients are the coordinates of the principal components in the Cartesian coordinate system. You will need them for interpreting the synthetic variables that contain PC scores.

- `x:` This element, the content of which is accessed as `mtcars.pca$x`, is the matrix of Principal Component scores (or PC scores) . It can be seen as a version of the initial dataset `mtcars` translated into the new coordinate system of principal components. In the new dataset, the original variables are replaced with the same number of "synthetic variables," which will further be analyzed. Each synthetic variable is a linear combination of all original variables. Here is an extract of the series of PC scores:

```
> round(head(mtcars.pca$x),4)
                      PC1     PC2     PC3     PC4     PC5     PC6
Mazda RX4         -0.6642  1.1734 -0.2043 - 0.1260  0.7520 -0.1251
Mazda RX4 Wag     -0.6372  0.9769  0.1108 -0.0857  0.6567 -0.0662
Datsun 710        -2.2997 -0.3266 -0.2101 -0.1086 -0.0762 -0.5669
Hornet 4 Drive    -0.2153 -1.9768 -0.3295 -0.3081 -0.2439  0.0838
```