**Lab 9: Principal Components Analysis in R**

**Lab Objectives:**

* Gain a brief introduction to a code based statistical software: R
* Use R to code and run a PCA
* Practice running a PCA on a dataset
* Learn to read the output and interpret the results

# Principal Components Analysis Essentials

### ****The Basics****

### **Principal component analysis** (**PCA**) allows us to summarize and to visualize the information in a data set containing individuals/observations described by multiple inter-correlated quantitative variables. Each variable could be considered as a different dimension. If you have more than 3 variables in your data set, it could be very difficult to visualize a multi-dimensional hyperspace.

### Principal component analysis is used to extract the important information from a multivariate data table and to express this information as a set of few new variables called **principal components**. These new variables correspond to a linear combination of the original variables.

### The information in a given data set, corresponds to the total variation it contains. The goal of PCA is to identify directions (or principal components) along which the variation in the data is maximal. In other words, PCA reduces the dimensionality of multivariate data to two or three principal components, that can be visualized graphically, with minimal loss of information.

### http://www.sthda.com/english/sthda-upload/figures/principal-component-methods/006-principal-component-analysis-scatter-plot-data-mining-1.pnghttp://www.sthda.com/english/sthda-upload/figures/principal-component-methods/006-principal-component-analysis-scatter-plot-data-mining-2.pngUnderstanding the details of PCA requires knowledge of linear algebra. Here, we’ll explain only the basics with simple graphical representation of the data. In Plot 1A, the data are represented in the X-Y coordinate system. The dimension reduction is achieved by identifying the principal directions, called principal components, in which the data varies. PCA assumes that the directions with the largest variances are the most “important” or the “most principal”. In Plot 1A, the PC1 axis is the first principal direction along which the samples show the largest variation. The PC2 axis is the second most important direction and it is orthogonal to the PC1 axis. The dimensionality of our two-dimensional data can be reduced to a single dimension by projecting each sample onto the first principal component (Plot 1B).

### http://www.sthda.com/english/sthda-upload/figures/principal-component-methods/006-principal-component-analysis-unnamed-chunk-3-1.pnghttp://www.sthda.com/english/sthda-upload/figures/principal-component-methods/006-principal-component-analysis-unnamed-chunk-3-2.pngThe amount of variance retained by each principal component is measured by the ****eigenvalue (****special set of scalars associated with a linear system of equations) or loading scores.

### Note that, the PCA method is particularly useful when the variables within the data set are highly correlated. Correlation indicates that there is redundancy in the data. Due to this redundancy, PCA can be used to reduce the original variables into a smaller number of new variables explaining most of the variance in the original variables (i.e. principal components).

### To summarize, the main goals of a PCA are to:

* Identify hidden patterns in a data set.
* Reduce the dimensionality of the data by removing the noise and redundancy in the data.
* Identify correlated variables.

### Loading Packages

**As seen on the tutorial video link posted on sakai: “**Getting started with R and RStudio”**, we need to use several functions from different packages. We will begin by simply calling the packages we will be using to compute the PCA today. Since you are working on a shared computer, some of these packages may have already been loaded by a previous user, therefore, we will load them into R using the following codes (just type in as seen below and hit enter after each line):**

> library(ggplot2)

> library(factoextra)

> library(datasets)

### If you do not get an error code, then all of the required packages have been previously loaded! If you get an error code under the line you typed, then you know the package has not been installed on the computer and will have to do so. Go to the plot and file panel and under packages, hit install and type the missing package out exactly as above in the “packages” space:

### C:\Users\Dorina Szuroczki\Desktop\packages.png

**Note:** you only need install a library package once, then you can just load them with the “library" command like we did above.

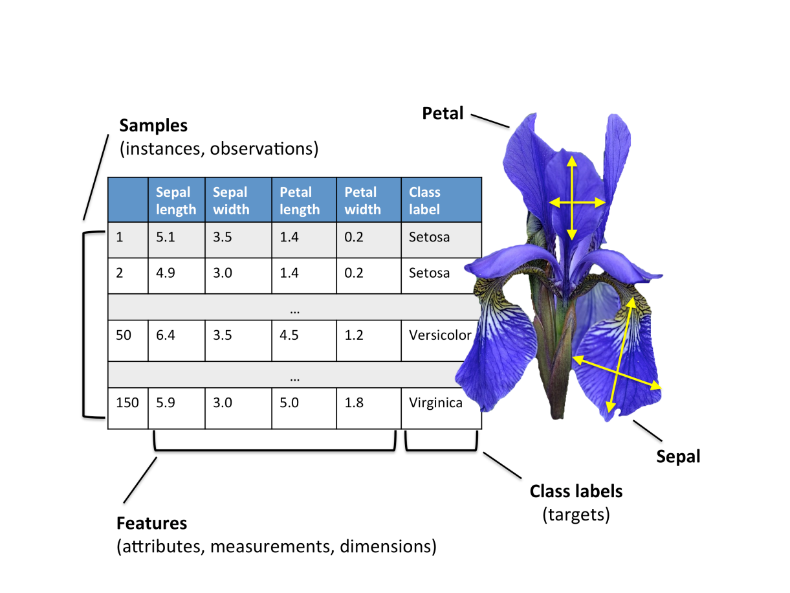
**If you end up using R in the future, you will get quite good at coding, so you can always save time by installing a package using straight code:**

> install.packages("ggplot2")

**Data format**

R has a large depository of different data sets in different libraries. For the PCA tutorial, we will use a famous data set that has been already loaded into R:

**Iris**  
This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of various structural components of 50 flowers from 3 species.

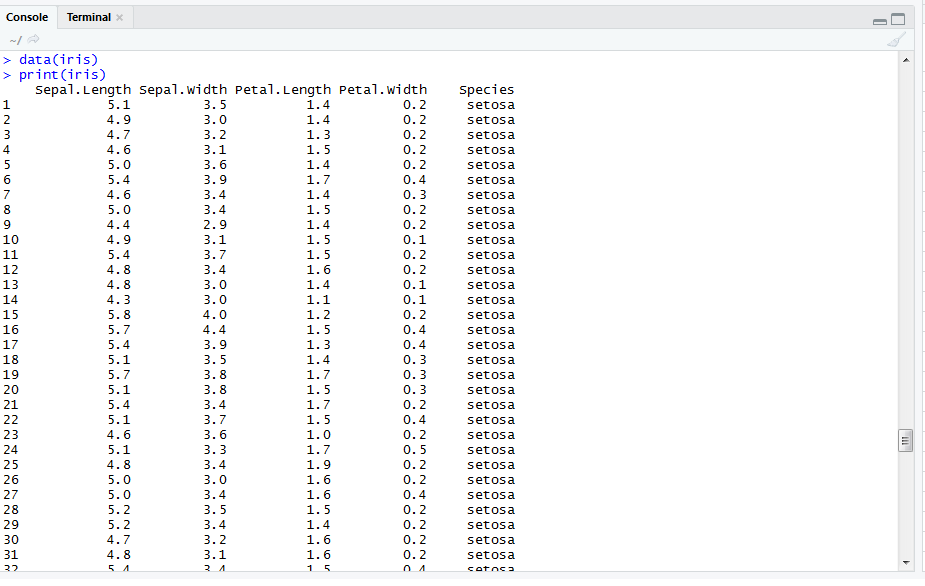
The three classes or species in the Iris dataset are:

1. *Iris setosa* (n=50)
2. *Iris versicolor* (n=50)
3. *Iris virginica* (n=50)

And the four features of the Iris dataset are:

1. *Sepal length* in cm
2. *Sepal width* in cm
3. *Petal length* in cm
4. *Petal width* in cm

Since this data is already present in the “datasets” package you previously loaded, all we need to do is call it, visualize it, and run some summary statistics:

****

> data(iris)

> print(iris)

> na.omit(iris) (see next page before running this line)

> summary(iris)

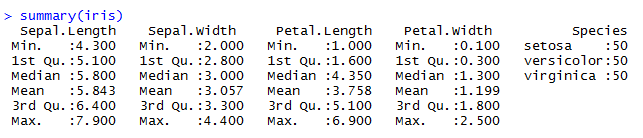
It’s a good idea to make sure your data has loaded properly. So quickly make sure that the data that R retrieved for you, match the above criteria (I’m only showing you the first 30 or so entries). In addition, another useful trick is to add in the line of code “na.omit()”, this will remove any blank cells in your data (should you have any). That being said, if say you have just a missing cell, R will delete the whole row the missing cell is found in. When you go to plot your final graph, the graphing function will try to match the data you used in the PCA to the original data and will not be able to match the two = R will have a HEART ATTACK! Basically, to use this code effectively, you need to know your data and ONLY use it when appropriate.

Another neat trick while visualizing the data, is to only print the first 6 rows or the last 6 rows using:

> head(iris)

> tail(iris)

You should notice that the “summary” command gives you a nice initial pass of the data, with key summary statistics:



**Loading your own data**

Often, you will likely want to import your own data into R. Therefore, I will provide you with

code to do so. **Note:** that R can take .txt, .csv and .xlsx files but generally, .csv files seem to be the

most stable file extension.

To import your own data:

> raw.data<-read.csv("C:/Users/Dorina Szuroczki/Desktop/dorina\_data.csv", stringsAsFactors = TRUE, header=TRUE)

The “raw.data” is the name you are giving your data set, so it can really be anything. The “<-“ symbol allows you to make an object. In this case, I’m making the .csv file "dorina\_data" an object named raw.data! “StringsAsFactors” and “header” are all lines of code that help R read the data in and are important, but I will not go into details here. In order to get the file location, its easiest to just right click on the saved file and under “properties”, you will see the location information.

**Preparing the data for PCA**

The easiest way to run a PCA, without hassle, is by subsetting the data to include only the variables that will be analyzed: exclude the factor or variable that is categorical as the PCA can only work with numeric variables; or if it is of interest, you will need to re-code it so that it is numerical (e.g.

setosa = 1, versicolor = 2 etc.). For the Iris example, we will simply exclude the categorical variable

“species” from the PCA. In addition, we need to scale or standardize the data because most often the variables are measured in different scales (e.g: kilograms, kilometers, centimeters, etc.); otherwise, the PCA outputs obtained will be severely affected. The goal is to make the variables comparable. Generally, variables are scaled to have i) standard deviation one and ii) mean zero.

> IrisPCAData <- as.data.frame(scale(iris[,1:4]))

You can see here, we are calling this new standardized data “IrisPCAData” and we are telling R to scale and to only work with the variables found in columns 1:4 (sepal length, width, petal length and width with “species” being ignored).

**Running the PCA**

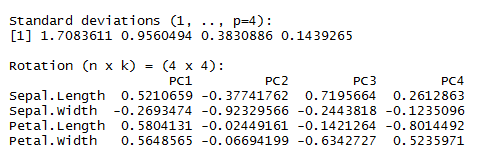
There are several functions from different packages available in the *R software* for computing PCA:

* *prcomp*() and *princomp*() [built-in R *stats* package],
* *PCA*() [*FactoMineR* package],
* *dudi.pca*() [*ade4* package],
* *epPCA*() [*ExPosition* package]

We will use the “prcomp” function:

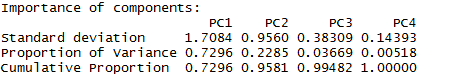
> IrisPca <- prcomp(IrisPCAData)

Here we have now run the PCA on the subsetted, standardized data, and have also given the results of the PCA its own name (“IrisPca”). This allows us to visualize different parts of the PCA easily. Next, we need to call the results of the PCA and a summary of the results:



> IrisPca

> summary(IrisPca)

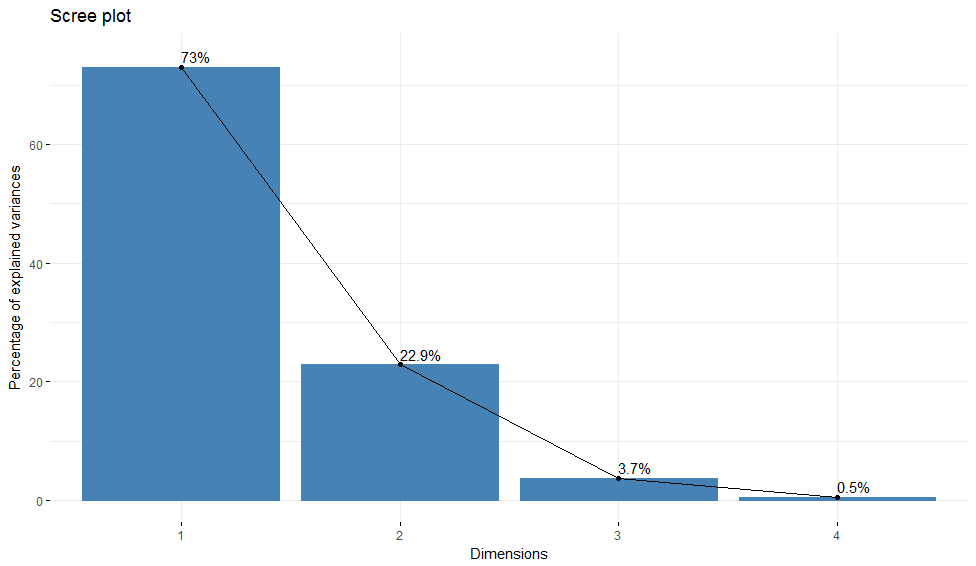


From the above results, we see in the first box, the loadings for each variable used in our PCA. “Rotation” refers to the number of principal components which are equal to the number of variables in the dataset. Considering rotations of PC1, we can conclude that sepal length, petal length and petal width are directly related, and they all are inversely related to sepal width (which has a negative value in rotation of PC1). One possible explanations for this, is that there may be a factor in plants (some chemical/physical functional system etc.) which may be affecting all these variables (sepal length, petal length and width in one direction, and sepal width in an opposite direction).

From the summary, we can understand that PC1 explains 73% of variance and PC2 explains 23% and so on. Unfortunately, there is no well-accepted objective way to decide how many principal components are enough. This will depend on the specific field of application and the specific data set. In practice, we tend to look at the first few principal components in order, to find interesting patterns in the data. In our analysis, the first two principal components explain 96% of the variation, which is great (we aim for as close to 95% as possible)! Thus, we would only retain PC1 and PC2

An alternative method to determine the number of principal components is to look at a scree plot, which is the plot of eigenvalues/loadings ordered from largest to the smallest. The number of components is determined at the point, beyond which the remaining eigenvalues are all relatively small and of comparable size.

> fviz\_eig(IrisPca, addlabels = TRUE, ylim = c(0, 75))

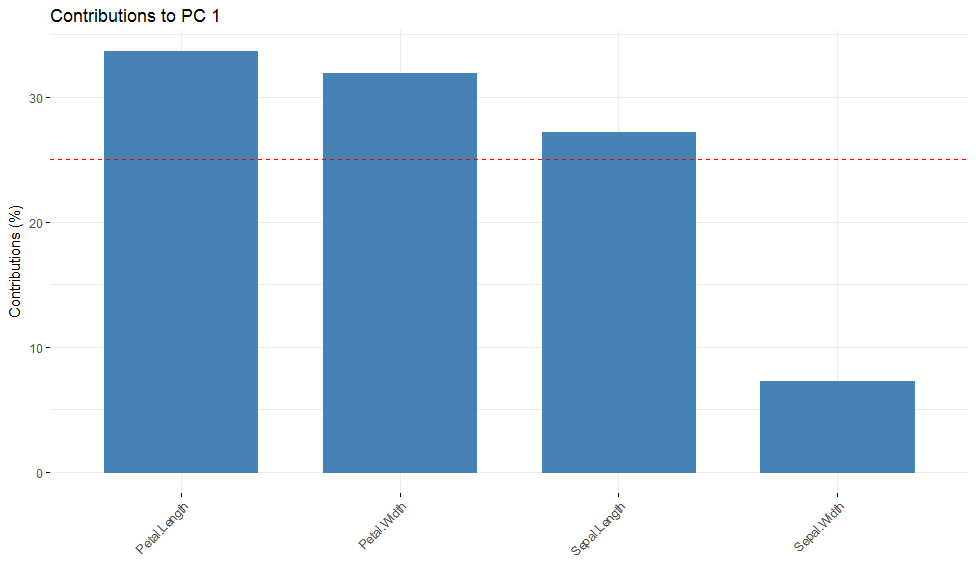
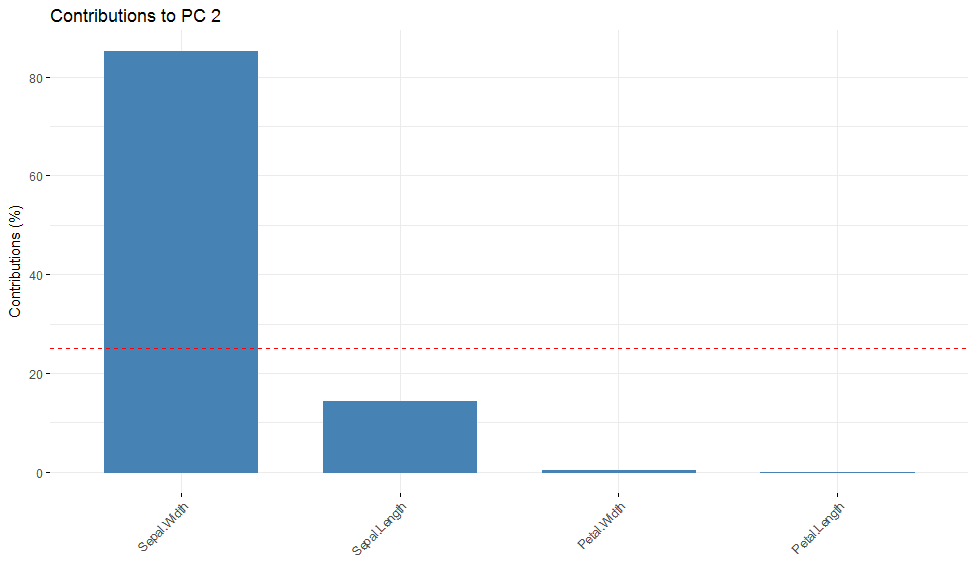
The above line of code will generate the scree plot corresponding to the IrisPca and we can easily see here that PC1 and PC2 explain most of the variance in our dataset. Note: you need to adjust the height of the Y-axis to fit the largest proportion of variance (PC1 = 73%), thus we adjust the height to 75: “c(0,75)”.

Next, we can determine how each variable contributes to the most important principal components (a.k.a as loading scores), in our case, PC1 and PC2:

> fviz\_contrib(IrisPca, choice="var", axes = 1 )+labs(title = "Contributions to PC 1")

> fviz\_contrib(IrisPca, choice="var", axes = 2 )+labs(title = "Contributions to PC 2")

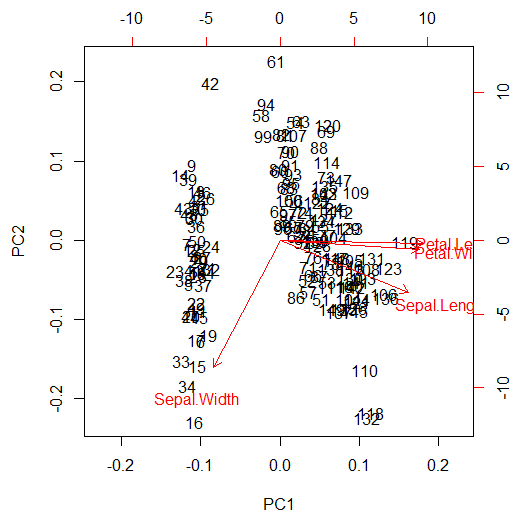
Note how the “axes = “changes to reflect each PC of interest. Also, the back half of the code is just setting up the figure header.



We can clearly see, that petal length, width and sepal length contribute the most in explaining the 73% variance on PC1 and subsequently, sepal width contributes the most in explaining the 23% variance on PC2.

# Graphing the Results of a PCA – the FUN Part!!!!

It’s often easier to draw inferences from your PCA by graphing the data. We have a few options for this. We can plot our PCA in 2-dimensions = biplot:

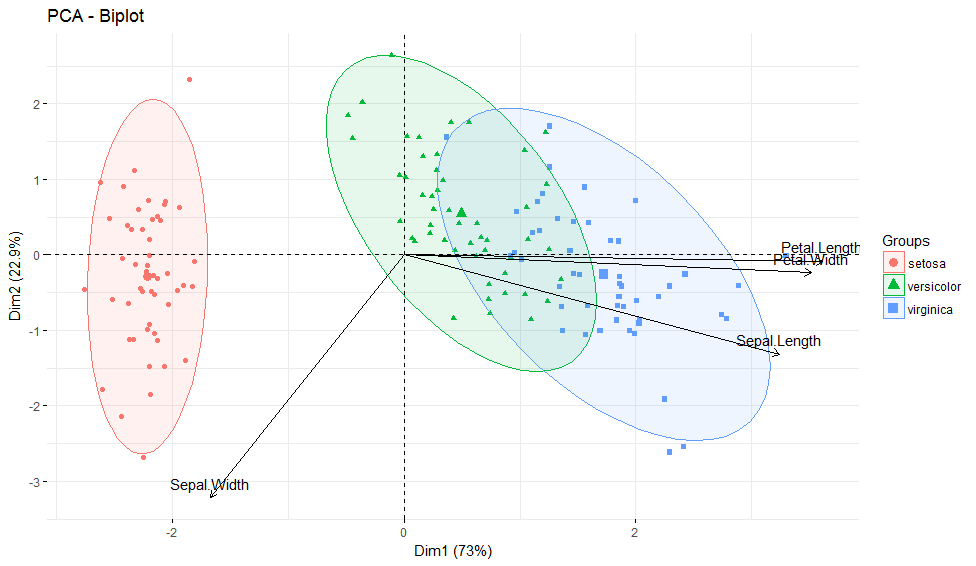


> biplot(IrisPca, col = c("black", "red"))

The biplot graph shows each individual flower (denoted by numbers) and then the 4 variables used in the PCA. The arrows are vectors and scale according to the importance of the variables. We can see some clear clustering here which is no surprise considering the scree plots we performed. This graphical representation is fine; however, we can vastly improve the clarity with just a few lines of code:

> fviz\_pca\_biplot(IrisPca, label="var", habillage=iris$Species, col.var = "black",addEllipses=TRUE, ellipse.level=0.95, labelsize=4)

Here, all we did is create a biplot that is differentiated by the categorical variable of interest= “species” (e.g. habillage=iris$species), which is color coded. We also added 95% confidence ellipses that concentrate our points. The “labelsize=4”, just makes the variable names large enough to read:



This is where the “na.omit” would cause you problems if used improperly. Remember, we ignored the “species” column (5) for the PCA but we are now calling on that data to create this plot. If we omitted data from the numerical variables only, R would not be able to match the two to execute this biplot.

On to the data…..we can now clearly see the 3-different species and how they are separated out. We still see the vectors and the variables that are important in clustering the species. This PCA is quite definitive: for *I. setosa*, the main variable that explains the greatest variation, is sepal width (as seen on PC2). For PC1 or *I. versicolor* and *I. virginica*, we can see that there is overlap of the ellipses, so its harder to isolate the species. We can see that 3 variables still dominate in describing both species: petal length, width and sepal length. That being said, visually, petal length has a slightly greater impact for both *I. virginica* and *I. versicolor*.

**Note on confidence ellipses:**

The variance of the underlying population relates to the confidence ellipse. High variance will mean that the data are all over the place, so the mean is not well estimated, so the confidence ellipse will be larger than if the variance within the population were smaller.

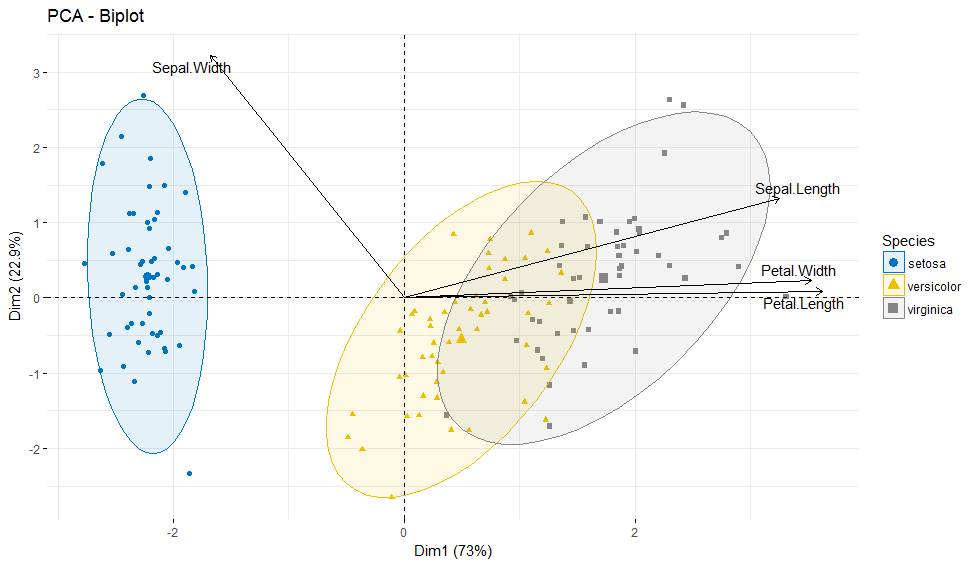
This is a great spot to end, you have all you need to run and interpret a PCA for any data set you choose. However, below, I’ll include some ways to make your plots look “cleaner”. We can change the colours of the species by telling R the plot is an object:

> p<-fviz\_pca\_biplot(IrisPca, label="var", habillage=iris$Species, col.var = "black",addEllipses=TRUE, ellipse.level=0.95, labelsize=4)

Then you can manually change the colours to almost anything you want:

> p + scale\_color\_manual(values=c("green", "blue", "red"))+scale\_fill\_manual(values=c("green", "blue", "red"))

Finally, I took things one step further and flipped my vectors, changed the colours and added a title to the legend to make things as nice as I possibly could:



In order to do this, I re-ran my PCA using another package: “FactoMineR”:

> library("FactoMineR")

> library("factoextra")

> iris.pca<-PCA(iris[,1:4], graph=FALSE)

> fviz\_pca\_biplot(iris.pca, habillage = iris$Species, palette = "jco", addEllipses = TRUE, label = "var",col.var = "black", repel = TRUE,legend.title = "Species")

# This really illustrates the power of R. There is always many ways to achieve one end goal, meaning R gives you the power to customize your outputs and statistics in anyway you choose. This however, can be a double-edged sword if you do not know what you are doing or how to interpret your statistic!

**Final Note for the R beginner**

R can be quite challenging but once you get the hang of coding, clear patterns start to emerge that help you anticipate how to code future commands. In addition, since R is a free program that has

thousands of “beautiful minds” contributing packages, there is TONS of help and codes online.

Often times, if you run into a coding error you can always find a way to overcome it by googling it.

In the same way, if you have no idea how to code a command of interest, you can always find it online with examples!

# Exercise

* 1. Download the excel file on sakai in the “Lab 9” folder.
  2. Convert the file to a .csv file extension and save it somewhere you can easily access.
  3. Try going through the above steps to run and plot a PCA for the data provided.
  4. Answer the questions on sakai “Lab 9 – PCA” under the test and quizzes tab.

Reference:

Kassambara, A. 2017. Practical guide to Principal components methods in R (multivariate analysis), Vol. 2. STHDA, USA.