**An Intuitive Guide to Principal Component Analysis (PCA) in R: A Step-by-Step Tutorial with Beautiful Visualization Examples**

[[Allan victor](https://medium.com/@victorallan?source=post_page-----73bce5ee02e5--------------------------------)](https://medium.com/@victorallan?source=post_page-----73bce5ee02e5--------------------------------)

[Allan victor](https://medium.com/@victorallan?source=post_page-----73bce5ee02e5--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Ff740847e5509&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40victorallan%2Fan-intuitive-guide-to-pca-in-r-a-step-by-step-tutorial-with-beautiful-visualization-examples-73bce5ee02e5&user=Allan+victor&userId=f740847e5509&source=post_page-f740847e5509----73bce5ee02e5---------------------post_header-----------)

14 min read

·

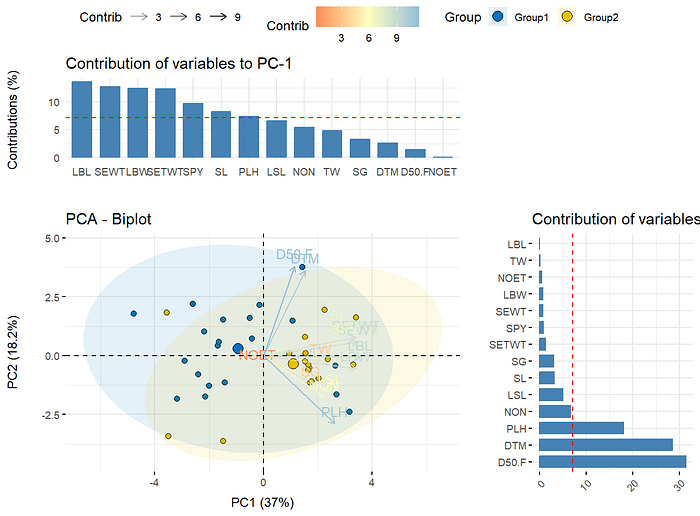
Nov 20

7

2

“*Don’t give up seeing the exhaustive lines of code. It’s just copy and paste then Run!!*”. Stay with me, and I will show you how to generate the plot below.

*C****omment what R tutorial should I do next, and I will try to do within 2 days and follow me to know latest updates.***



**Introduction:**

Principal component analysis (PCA) is a common technique for performing dimensionality reduction on multivariate data. By transforming the data into principal components, PCA allows visualization and analysis of patterns in a dataset, aiding interpretation and feature selection. In this article, I will provide an intuitive guide to conducting PCA in R, including a step-by-step walkthrough using the powerful [*FactoMineR*](https://cran.r-project.org/web/packages/FactoMineR/index.html)package.

**Note: You can visit the R shiny package “**[***PBperfect***](https://allanbiotools.shinyapps.io/pbperfect/)**” to do the PCA analysis and obtain the same results and plots as shown in this page in a single click.**

Specifically, I will demonstrate using four custom functions that I created myself for PCA analysis and visualization in R:

1. eigen\_table() - Computes eigenvalues and eigenvectors from PCA, outputting results as formatted tables
2. var\_stats() - Provides statistics on variable contributions, correlations and cos2 values for principal components.
3. varplot() - Generates various plots visualizing variable quality of representations and contributions in the PCA space.
4. biplot\_allan() - Creates an insightful biplot overlaying variables and observations with customization options.

By working through examples of implementing these four functions on a sample dataset, readers will develop an intuition for conducting end-to-end PCA in R, from analysis through interpretation and visualization of components and loadings. Clear explanations combined with beautiful data visualizations will provide an intuitive, practical introduction to harnessing the power of PCA using R.

The article will assume basic familiarity with R syntax but include code comments and explanation for those newer to R and PCA analysis. Now, let’s get started with the first step — loading the required R packages.

**Required R packages:**

* We will utilize several powerful R packages for conducting the PCA analysis and visualizing the outputs.

Install the packages if it's not installed previously using the following code,

install.packages("factoextra") #Easy PCA plots and visualization  
install.packages("FactoMineR") #PCA and factor analysis functions   
install.packages("ggplot2") #Enhanced data visualization  
install.packages("dplyr") #Data manipulation  
install.packages("gridExtra") #Arranging multiple plot grids  
install.packages("gt") # For creating beautiful tables

Load the following packages by running the below code,

library(factoextra) #Easy PCA plots and visualization  
library(FactoMineR) #PCA and factor analysis functions   
library(ggplot2) #Enhanced data visualization  
library(dplyr) #Data manipulation  
library(gridExtra) #Arranging multiple plot grids  
library(gt) # For creating beautiful tables

The key package is FactoMineR, which contains the main functions for efficiently running PCA in R and computing the resulting variable statistics.

factoextra builds on this by enabling quick construction of enhanced PCA plots. And ggplot2 facilitates customization of publication-quality graphics of the PCA results.

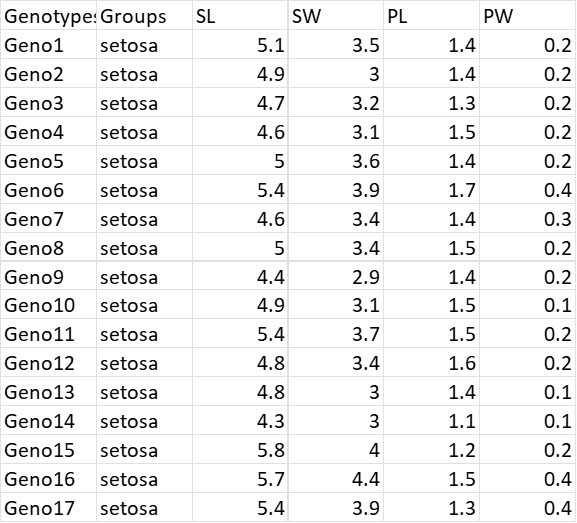
The dplyr and gridExtra packages supplement these core packages with handy data wrangling and multiple plot arrangement capabilities.

With these powerful data analysis and visualization packages loaded, we are now ready to introduce our example dataset and walk through applying PCA from start to finish using the four functions outlined earlier.

**Sample data:**

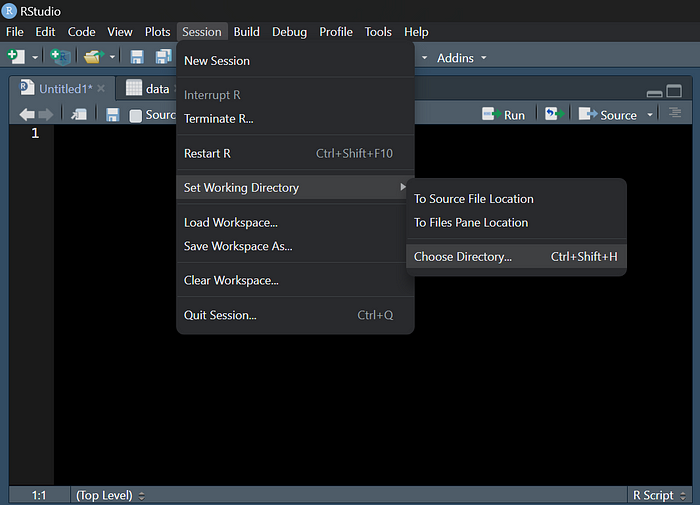
I modified the classic *iris* dataset from default R to better suit our PCA analysis. The example data can be downloaded from [*here*](https://mega.nz/file/ZTwBCKJS#EVdl4RxNUMZ9u8slmqxOY0rHTqa5VwBiELIQcxouvR4). The data is arranged with the following columns:

* **Genotypes**: Identifier for each observation
* **Groups**: Category grouping (setosa, versicolor, virginica). *You can give all the rows as a “****Group1****” or some other names, if there are no groups in the data.*
* **SL**: Sepal Length
* **SW**: Sepal Width
* **PL**: Petal Length



Data format

To import the data, first we have to set the directory of the session to the folder that contains the data. Don't worry, it can be easily done using the following step,



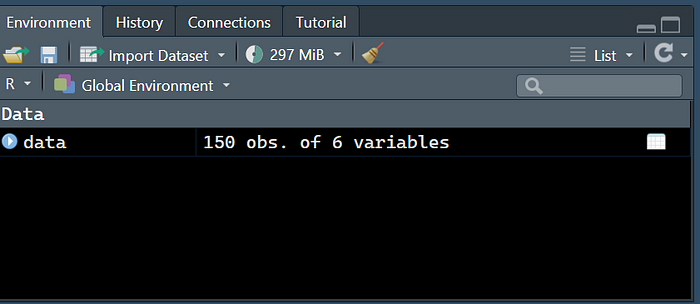
Now the data can be imported into R using the following code,

You can put you data name instead of the *PCA\_example.csv*. For example, “*PCAdata.csv*”

data= read.csv("PCA\_example.csv")

After running the code, you should see the data in the environment tab like below,

Make sure that the data is save in the name of “*data*”.



Now we can see the structure of the data using the code,

str(data)

The following code block shows the output for the example data. This output is the result of running the str() function, so kindly do not copy and paste it.

> str(data)  
'data.frame': 150 obs. of 6 variables:  
 $ Genotypes: chr "Geno1" "Geno2" "Geno3" "Geno4" ...  
 $ Groups : chr "setosa" "setosa" "setosa" "setosa" ...  
 $ SL : num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
 $ SW : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
 $ PL : num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
 $ PW : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...

**Data checks:**

1. The data should be saved in the name of “***data***” as shown in the previous parts.
2. The **first column** of the data should be the **names of the treatments without duplications.**
3. The **second column** should be the **grouping column**. Again, if there are no groups you can give “Group1” for all the treatments.
4. The data should start from the **third column** and all the data should be in **numerical**type.

**PCA analysis walkthrough:**

1. **Correlation and Covariance matrices:**

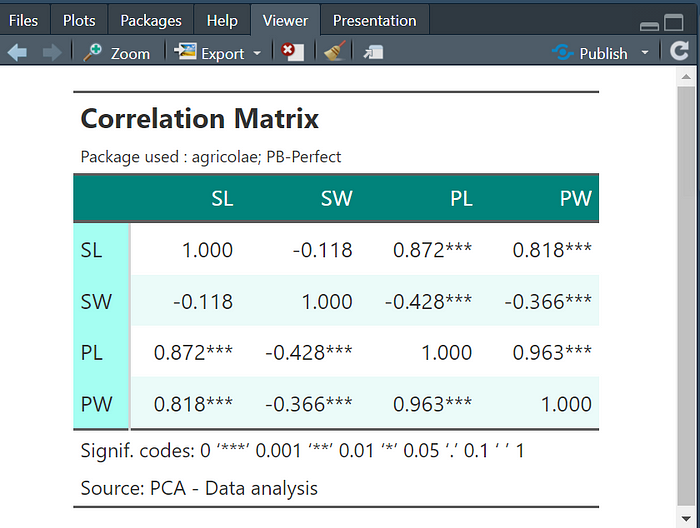
When conducting PCA, it is insightful first examine the correlation and covariance structure between the variables in our dataset.

The **correlation matrix** shows the linear correlation coefficients between each pair of variables. We can compute and visualize this for the example data using the ready-made function. **Kindly don’t be scared seeing the lengthy code block. It's just copy paste and run**.

**After the data is imported and passed the necessary checks**you are ready to run the code. Just copy everything from the below code block, **paste in R and run.**

##Function for correlation matrix with stars.   
cor\_mat\_allan <-function(df,  
 type = "pearson",  
 digits = 3,  
 decimal.mark = ".",  
 use = "all",  
 show\_significance = TRUE,  
 replace\_diagonal = FALSE,  
 replacement = ""){  
   
 # check arguments  
 stopifnot({  
 is.numeric(digits)  
 digits >= 0  
 use %in% c("all", "upper", "lower")  
 is.logical(replace\_diagonal)  
 is.logical(show\_significance)  
 is.character(replacement)  
 })  
 # we need the Hmisc package for this  
 require(Hmisc)  
   
 # retain only numeric and boolean columns  
 isNumericOrBoolean = vapply(df, function(x) is.numeric(x) | is.logical(x), logical(1))  
 if (sum(!isNumericOrBoolean) > 0) {  
 cat('Dropping non-numeric/-boolean column(s):', paste(names(isNumericOrBoolean)[!isNumericOrBoolean], collapse = ', '), '\n\n')  
 }  
 df = df[isNumericOrBoolean]  
   
 # transform input data frame to matrix  
 x <- as.matrix(df)  
   
 # run correlation analysis using Hmisc package  
 correlation\_matrix <- Hmisc::rcorr(x, type = )  
 R <- correlation\_matrix$r # Matrix of correlation coeficients  
 p <- correlation\_matrix$P # Matrix of p-value  
   
 # transform correlations to specific character format  
 Rformatted = formatC(R, format = 'f', digits = digits, decimal.mark = decimal.mark)  
   
 # if there are any negative numbers, we want to put a space before the positives to align all  
 if (sum(R < 0) > 0) {  
 Rformatted = ifelse(R > 0, paste0(' ', Rformatted), Rformatted)  
 }  
   
 # add significance levels if desired  
 if (show\_significance) {  
 # define notions for significance levels; spacing is important.  
 stars <- ifelse(is.na(p), " ", ifelse(p < .001, "\*\*\*", ifelse(p < .01, "\*\* ", ifelse(p < .05, "\* ", " "))))  
 Rformatted = paste0(Rformatted, stars)  
 }  
 # build a new matrix that includes the formatted correlations and their significance stars  
 Rnew <- matrix(Rformatted, ncol = ncol(x))  
 rownames(Rnew) <- colnames(x)  
 colnames(Rnew) <- paste(colnames(x), "", sep =" ")  
   
 # replace undesired values  
 if (use == 'upper') {  
 Rnew[lower.tri(Rnew, diag = replace\_diagonal)] <- replacement  
 } else if (use == 'lower') {  
 Rnew[upper.tri(Rnew, diag = replace\_diagonal)] <- replacement  
 } else if (replace\_diagonal) {  
 diag(Rnew) <- replacement  
 }  
   
 return(Rnew)  
}  
  
  
correlation\_matrix= cor\_mat\_allan(data[,-c(1,2)])%>%as.data.frame()%>%gt( rownames\_to\_stub = TRUE) %>%tab\_header(  
   
 title = md("\*\*Correlation Matrix\*\*"),  
 subtitle = "Package used : agricolae; PB-Perfect"  
)%>%tab\_source\_note(  
 source\_note = "Source: PCA - Data analysis"  
)%>%  
 tab\_options(  
 heading.subtitle.font.size = 12,  
 heading.align = "left",  
 table.border.top.color = "red",  
 column\_labels.border.bottom.color = "red",  
 column\_labels.border.bottom.width= px(3)  
 )%>%opt\_stylize(style = 6, color = "cyan")%>%  
 tab\_options(table.width = pct(80))%>%  
 tab\_footnote(  
 footnote = "Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1")  
  
correlation\_matrix

If everything goes smoothly you should see the results as shown below.

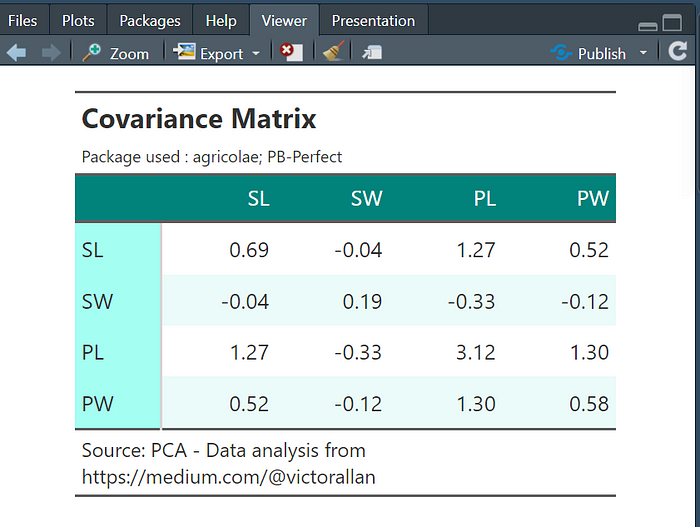


**We will see how to export everything to presentation quality tables and figures in the later parts of the article. For now, we will continue to do the analysis.**

The **covariance matrix** displays the degree to which variables vary together. We obtain this matrix via the same process —**copy, paste and run the below code.**

# Function for covariance matrix  
covariance\_table=cov(data[,-c(1,2)])%>%round(2)%>%as.data.frame()%>%gt( rownames\_to\_stub = TRUE) %>%tab\_header(  
 title = md("\*\*Covariance Matrix\*\*"),  
 subtitle = "Package used : agricolae; PB-Perfect"  
)%>%tab\_source\_note(  
 source\_note = "Source: https://medium.com/@victorallan"  
)%>%  
 tab\_options(  
 heading.subtitle.font.size = 12,  
 heading.align = "left",  
 table.border.top.color = "red",  
 column\_labels.border.bottom.color = "red",  
 column\_labels.border.bottom.width= px(3)  
 )%>%opt\_stylize(style = 6, color = "cyan")%>%  
 tab\_options(table.width = pct(80))  
  
covariance\_table

If the code successfully runs, you should see the table as shown below,



The diagonals correspond to variances of each variable, while the off-diagonals are the covariances between variables. These correlation and covariance insights motivate the use of PCA — since some variables are highly related, PCA can identify the key underlying components that explain the overall structure in the data.

Let’s now move on to extracting and interpreting those principal components.

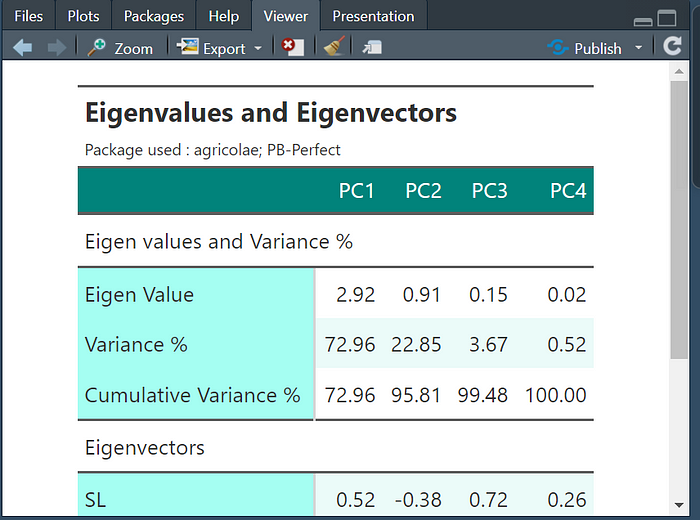
**2. Eigen values:**

The eigenvalues from PCA indicate the amount of total variance explained by each principal component. We can extract the eigenvalues from our PCA model using our ready-made function.

Do the same. **Copy the code, paste in R and Run!!** Always make sure the data format is correct as shown in the previous parts of the article.

eigen\_table=function(data){  
 library(factoextra)  
 library(FactoMineR)  
 library(gt)  
 data=data[,-c(1,2)]  
 data=na.omit(data)  
   
 pc=PCA(data,scale.unit = T, graph = F)  
 x=get\_eigenvalue(pc)  
 x=t(x)  
 rownames(x)=c("Eigen Value","Variance %", "Cumulative Variance %")  
 colnames(x)=paste("PC",1:ncol(x),sep = "")  
 x=round(x,2)  
   
 y=eigen(cor(data))$vectors  
 colnames(y)=paste("PC",1:ncol(x),sep = "")  
 rownames(y)=colnames(data)  
   
 x=rbind(x,y)  
   
 x=as.data.frame(x)  
 x=round(x,2)  
   
   
 grp=c(rep("Eigen values and Variance %",3),rep("Eigenvectors",nrow(y)))  
   
 x\_out=cbind(grp,x)  
 x=cbind(x,grp)  
   
   
 x=x%>%gt(groupname\_col = "grp", rownames\_to\_stub = TRUE)%>%  
 tab\_header(  
 title = md("\*\*Eigenvalues and Eigenvectors\*\*"),  
 subtitle = "Package used : agricolae; PB-Perfect")%>%  
 tab\_source\_note(source\_note = "Source: https://medium.com/@victorallan")%>%  
 tab\_options(  
 heading.subtitle.font.size = 12,  
 heading.align = "left",  
 table.border.top.color = "red",  
 column\_labels.border.bottom.color = "red",  
 column\_labels.border.bottom.width= px(3)  
 )%>%opt\_stylize(style = 6, color = "cyan")%>%  
 tab\_options(table.width = pct(80))  
 return(x)  
   
}  
  
  
eig\_table=eigen\_table(data)  
eig\_table

If the code runs perfectly, you should see the results as shown below,

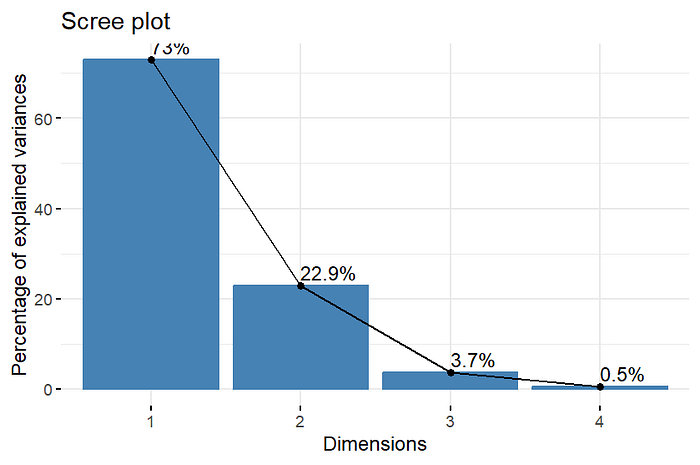


**3. Scree plot:**

A scree plot visually displays the variance explained by each principal component from the PCA. We can easily generate a scree plot using our ready-made function as below. Same process again. **Copy the code, paste in R and Run!!**

pca\_res=PCA(data[,-c(1,2)], graph = F)  
screeplot=fviz\_eig(pca\_res, addlabels = TRUE,ncp = 200)  
plot(screeplot)

A successful execution of the code will produce the following plot,



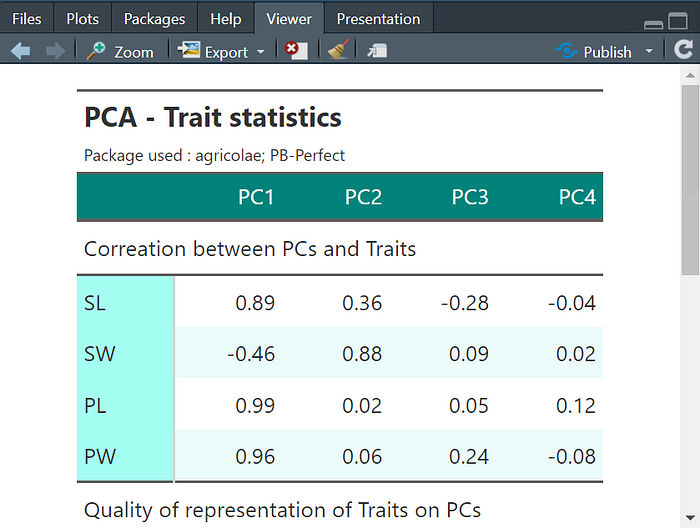
**4. Statistics for the variables:**

Understanding the variable contributions, correlations, and quality of representations within the principal component space provides further insights from PCA.

The var\_stats() function computes and neatly formats key variable statistics into an analysis table. Same process again. **Copy the code, paste in R and Run!!**

var\_stats=function(data){  
 library(FactoMineR)  
 library(dplyr)  
 data=data[,-c(1,2)]  
  
  
 res.pca=get\_pca\_var(PCA(data))  
 res.pca=lapply(res.pca, as.data.frame)  
 res.pca=lapply(res.pca, function(x){colnames(x)=c(paste("PC",1:ncol(x),sep = ""));x})  
 res.pca=lapply(res.pca, round,2)  
  
 cor\_pca=res.pca$cor  
 cos\_pca=res.pca$cos2  
 contrib\_pca=res.pca$contrib  
  
 grp=("Correation between PCs and Traits")  
 cor\_pca=cbind(cor\_pca,grp)  
 grp=("Quality of representation of Traits on PCs")  
 cos\_pca=cbind(cos\_pca,grp)  
 grp=c("Contributions(%) of the Traits to the Principal Components")  
 contrib\_pca=cbind(contrib\_pca,grp)  
  
 res=rbind(cor\_pca, cos\_pca, contrib\_pca)  
 x=res%>%gt(groupname\_col = "grp", rownames\_to\_stub = TRUE)%>%  
 tab\_header(  
 title = md("\*\*PCA - Trait statistics\*\*"),  
 subtitle = "Package used : agricolae; PB-Perfect")%>%  
 tab\_source\_note(source\_note = "Source: https://medium.com/@victorallan")%>%  
 tab\_options(  
 heading.subtitle.font.size = 12,  
 heading.align = "left",  
 table.border.top.color = "red",  
 column\_labels.border.bottom.color = "red",  
 column\_labels.border.bottom.width= px(3)  
 )%>%opt\_stylize(style = 6, color = "cyan")%>%  
 tab\_options(table.width = pct(80))  
  
 return(x)  
  
}  
  
variable\_stats = var\_stats(data)  
variable\_stats

If everything goes in line, you should see the results as shown below,



The output table has the following sections:

**Correlation between PCs and Traits** — The Pearson correlation coefficients between original variables and principal components. Gives a sense of association.

**Quality of representation of Traits on PCs** — The cos2 value represents quality of variable representation on the PC axes. Higher values indicate variables well represented by the components.

**Contributions (%) of the Traits to the Principal Components** — Percentage of contribution for each variable to the total variance of each PC. Larger values have higher contributions.

The table allows quick insights into:

* Which variables are *correlated* with the major PCs
* The variables *best represented* through the new PC axes
* Variables with the highest *contributions* to the components

Inspecting variable statistics supplements the eigenvalue and graphical analyses when interpreting your PCA solutions. The var\_stats() function provides an easy way to generate this in R.

**5. Correlation circle, Cos2 and Contribution of different traits:**

Visualizing variable statistics from PCA models provides further insight that supplements the numeric tables. We can create a variety of plots to understand relationships between the original variables and extracted principal components. You can copy paste the function from the code block below. Again, the same process, **Copy the code, paste in R and Run!!**

varplot=function(data){  
 library(FactoMineR)  
 require(factoextra)  
 library(dplyr)  
 require(stringi)  
 require(cowplot)  
 require(gridExtra)  
   
   
 data=data[,-c(1,2)]  
   
 res=PCA(data,graph = F)  
 var=get\_pca\_var(PCA(data, graph = F))  
 res.pca=get\_pca\_var(PCA(data, graph = F))  
 res.pca=lapply(res.pca, as.data.frame)  
 res.pca=lapply(res.pca, function(x){colnames(x)=c(paste("PC",1:ncol(x),sep = ""));x})  
 res.pca=lapply(res.pca, round,2)  
   
 cor\_pca=res.pca$cor  
 cos\_pca=res.pca$cos2  
 contrib\_pca=res.pca$contrib  
   
 cos\_contrib=fviz\_pca\_var(res, col.var = "contrib",  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")  
 )  
   
 cos\_contrib$labels$x=gsub("Dim","PC",cos\_contrib$labels$x)  
 cos\_contrib$labels$y=gsub("Dim","PC",cos\_contrib$labels$y)  
   
   
 cos\_cor=fviz\_pca\_var(res, col.var = "cos2",  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 repel = TRUE # Avoid text overlapping  
 )  
   
 cos\_cor$labels$x=gsub("Dim","PC",cos\_contrib$labels$x)  
 cos\_cor$labels$y=gsub("Dim","PC",cos\_contrib$labels$y)  
   
   
   
 temp1=fviz\_cos2(res, choice = "var", axes = 1)  
 temp2=fviz\_cos2(res, choice = "var", axes = 2)  
 temp3=fviz\_cos2(res, choice = "var", axes = 3)  
   
 # Contributions of variables to PC1  
 temp4=fviz\_contrib(res, choice = "var", axes = 1)  
 # Contributions of variables to PC2  
 temp5=fviz\_contrib(res, choice = "var", axes = 2)  
 temp6=fviz\_contrib(res, choice = "var", axes = 3)  
   
 temp1$labels$title="Cos2 of variables to PC-1"  
 temp2$labels$title="Cos2 of variables to PC-2"  
 temp3$labels$title="Cos2 of variables to PC-3"  
   
 temp4$labels$title="Contribution of variables to PC-1"  
 temp5$labels$title="Contribution of variables to PC-2"  
 temp6$labels$title="Contribution of variables to PC-2"  
   
   
   
   
 first\_row=plot\_grid(cos\_contrib,cos\_cor, nrow = 1)  
 second\_row=plot\_grid(temp1,temp2,temp3,nrow = 1)  
 third\_row=plot\_grid(temp4,temp5,temp6,nrow = 1)  
   
 return(list(first\_row,second\_row,third\_row,temp4,temp5))  
   
}  
  
  
variable\_plot=varplot(data)

The varplot() function generates various plots visualizing key statistics for interpreting variables within the principal component space.

Calling varplot(data) outputs:

1. Variable contribution/cos2 plots
2. Variable cos2 plots for PC1/PC2/PC3
3. Variable contributions for PC1/PC2/PC3

Let’s examine these plots to better understand variable relationships with the extracted principal components:

**Variable Contribution/Cos2 Plot**

* Positions variables based on *contributions* and *cos2* (quality of representation)
* Closer to edge = higher contribution, Higher cos2 = better represented

**Variable Cos2 Plots**

* Shows Cos2 values measuring representation on each PC
* High cos2 = variables well represented on that PC axis

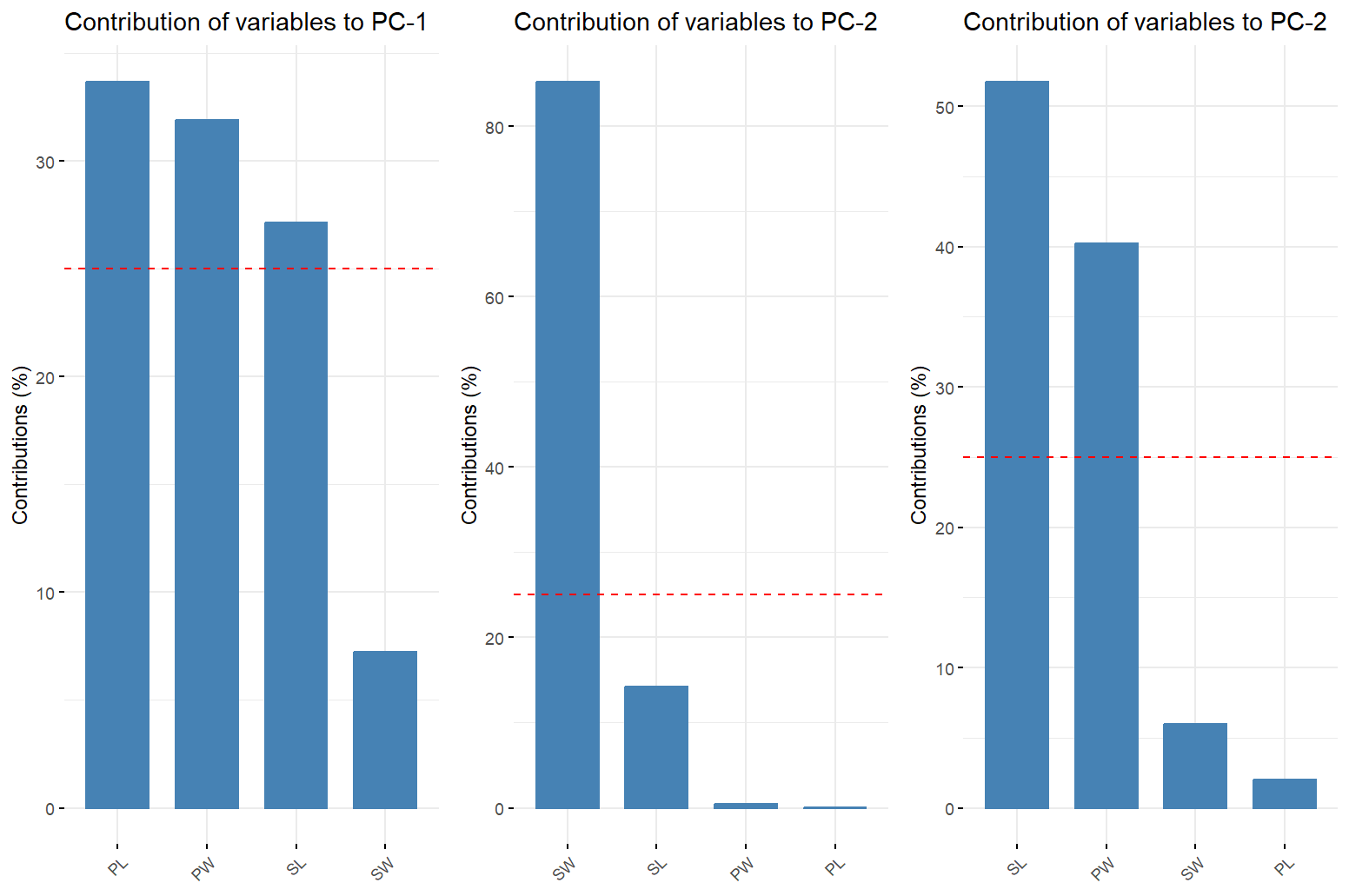
**Variable Contribution Plots**

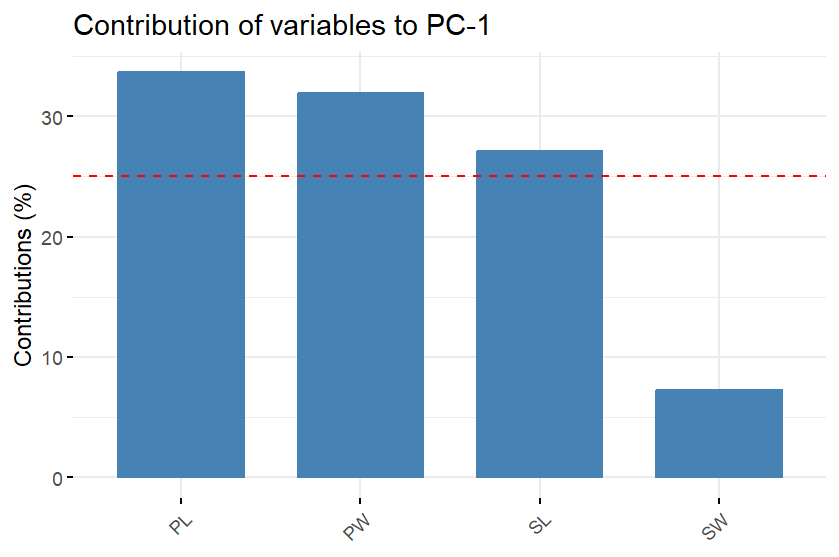
* Displays variable contributions to the PC variance
* Higher values have greater influence on that component

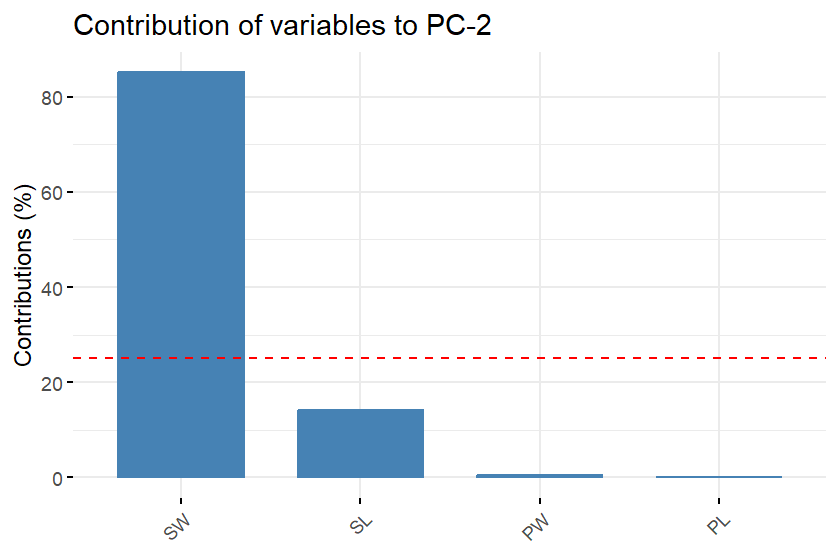
The varplot() combines these complementary plots, enabling deeper insight into how the original iris variables are expressed in the new PCA axes.

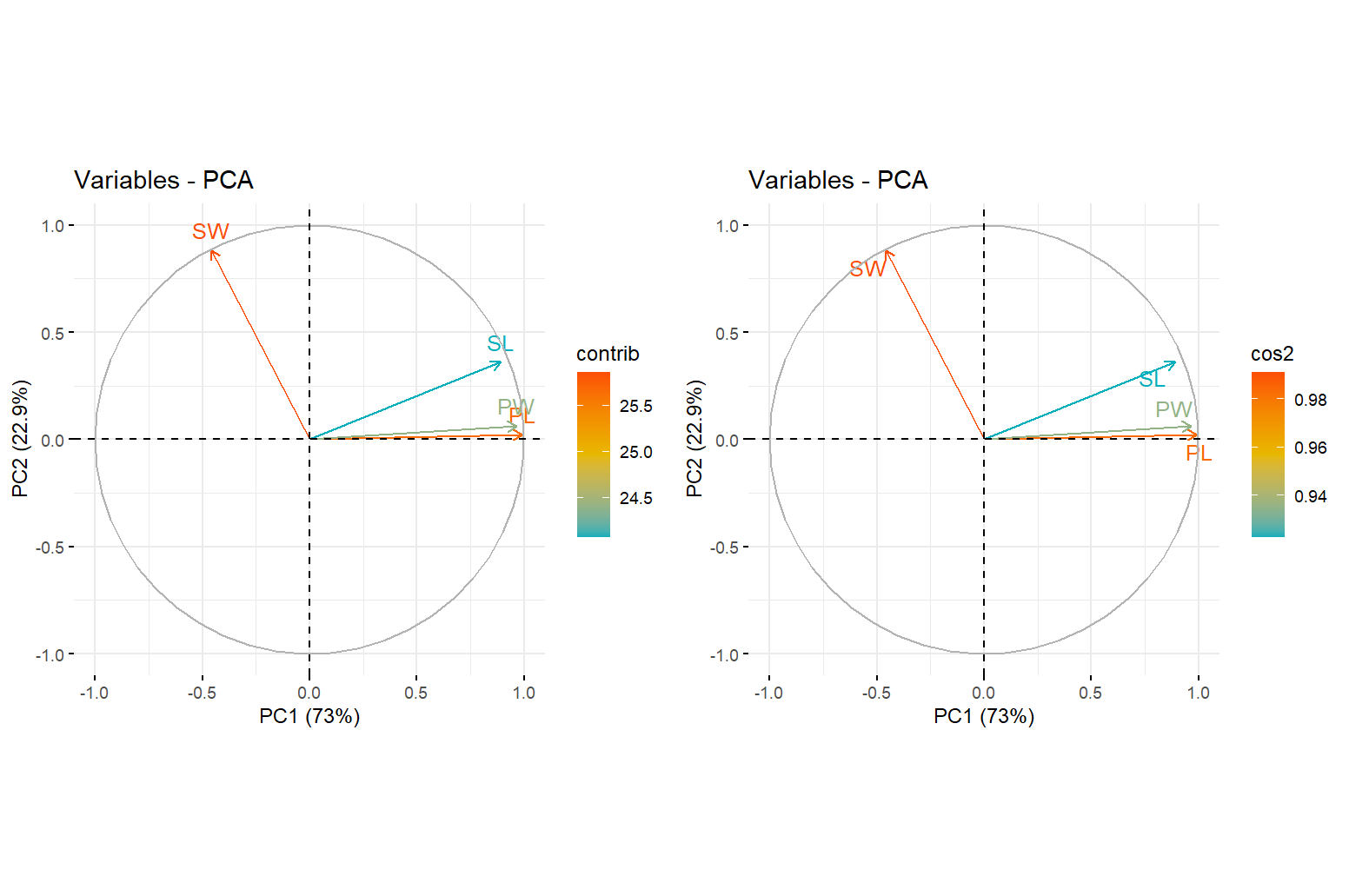
You can display or see the plots using the following code below,

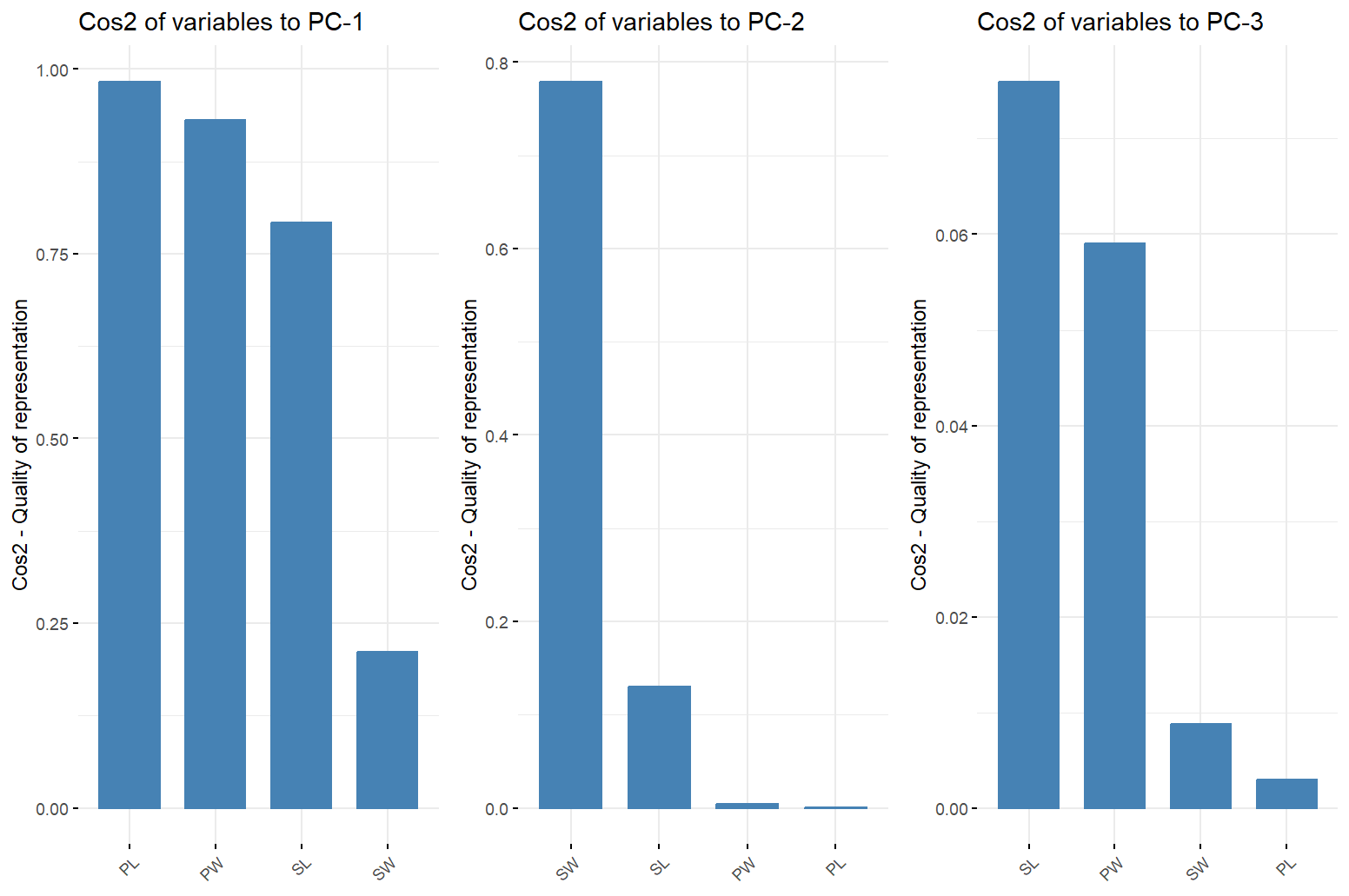
variable\_plot[[1]]  
variable\_plot[[2]]  
variable\_plot[[3]]  
variable\_plot[[4]]  
variable\_plot[[5]]











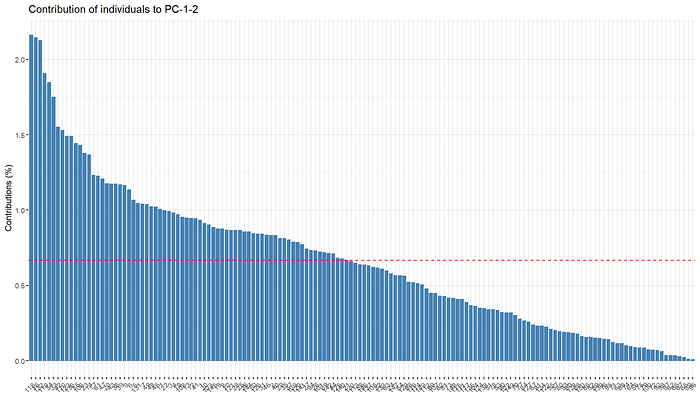
**6. Contribution of each individual:**

We can also visualize the contributions of individual genotypes - observations to the principal component variances using:

**Copy the code, paste in R and Run!!**

ind\_plot = fviz\_contrib(pca\_res, choice = "ind", axes = 1:2)  
ind\_plot$labels$title = "Contribution of Individuals to PC-1-2"  
ind\_plot

If everything goes well, we should see the plot as shown below,



**7. PCA biplot:**

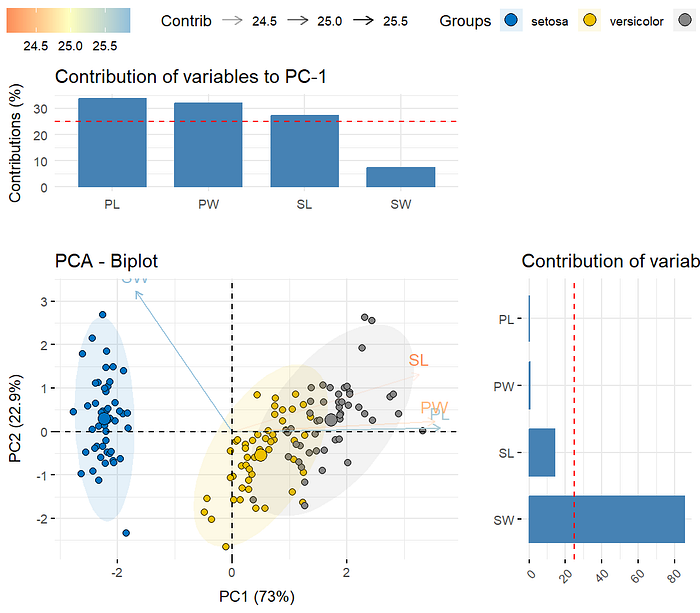
The PCA biplot overlays both samples and variables into a single plot for enhanced interpretation. We can generate a customized biplot using the biplot\_allan() function.

**Copy the code, paste in R and Run!!**

Also, a point to note is that you can increase the pt\_size to make the points in the biplot bigger, set contrib\_plot to TRUE to remove the contribution plots on X and Y axis. Set the ellipse to false to remove the ellipses in the biplot.

#Parameters you can adjust  
pt\_size=2  
contrib\_plot=T  
ellipse=T  
  
#To function for the biplot  
  
biplot\_allan=function(data,pca\_res,var\_plot,pt\_size,contrib\_plot=T,ellipse=T){  
 require(ggpubr)  
 grps=colnames(data)[2]  
 grp\_vec=as.factor(data[,2])  
 data=data[,-c(1,2)]  
 xplot <- var\_plot[[4]]  
 yplot <- var\_plot[[5]]  
 # Cleaning the plots  
   
 xplot=xplot+xlab("")+theme\_minimal()  
 yplot <- yplot + ylab("Contribution (%)")+ xlab("")+rotate()  
   
   
   
 y=fviz\_pca\_biplot(pca\_res,  
 # Individuals  
 geom.ind = "point",  
 fill.ind = grp\_vec, col.ind = "black",  
 pointshape = 21, pointsize = pt\_size,  
 palette = "jco",  
 addEllipses = ellipse,  
 # Variables  
 alpha.var ="contrib", col.var = "contrib",  
 gradient.cols = "RdYlBu",  
   
 legend.title = list(fill = grps, color = "Contrib",  
 alpha = "Contrib")  
 )  
   
   
   
 y$labels$x=gsub("Dim","PC",y$labels$x)  
 y$labels$y=gsub("Dim","PC",y$labels$y)  
   
 z=ggarrange(xplot, NULL, y, yplot,  
 ncol = 2, nrow = 2, align = "hv",  
 widths = c(2, 1), heights = c(1, 2),  
 common.legend = TRUE)  
 if(contrib\_plot){  
 return(z)  
 }else{  
 return(y)  
 }  
   
}  
  
  
  
biplot=biplot\_allan(data,pca\_res,  
 pt\_size=pt\_size,  
 contrib\_plot=contrib\_plot,  
 ellipse=ellipse,  
 var\_plot=variable\_plot)  
  
biplot

If the code runs successfully, you should see the biplot like below,



This biplot includes:

**Individuals**

* Plotted as points colored by iris species (The second column of the data is used as group here).
* Larger sizes indicate observations influencing PCs more.
* Ellipses show clustering pattern.

**Variables**

* Arrows indicate strength and direction.
* Color shading represents % contribution.

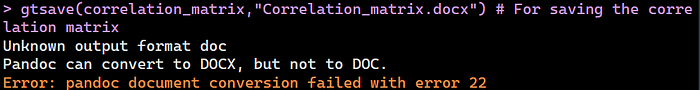
**Saving the results:**

**Saving individual elements**

The results of the above analysis include 4 tables and 4 plots. All the output tables are in “[*gt*](https://gt.rstudio.com/index.html)” format. So, the tables can be saved as Microsoft word, into the directory as,

gtsave(correlation\_matrix,"Correlation\_matrix.docx") # For saving the correlation matrix  
gtsave(covariance\_table,"covariance\_table.docx") # For saving the covariance matrix  
gtsave(eig\_table,"eigen\_table.docx") # For saving the eigen table  
gtsave(variable\_stats,"variable\_stats.docx") # For saving the variable statistics  
  
  
#You can also save the plot in different formats like  
# "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windows only)  
# by just changing the .pdf to a different extension  
  
  
# You can also play with the Width and Height of the plot to get the optimum results.  
  
ggsave(plot= screeplot,  
 filename="Scree\_plot.pdf",  
 width = 10,  
 height = 10) # For saving the Scree plot  
  
  
  
ggsave(plot= variable\_plot[[1]],  
 filename="varplot1.pdf",  
 width = 10,  
 height = 10) # For saving the variable plot  
ggsave(plot= variable\_plot[[2]],  
 filename="varplot2.pdf",  
 width = 10,  
 height = 10) # For saving the variable plot  
ggsave(plot= variable\_plot[[3]],  
 filename="varplot3.pdf",  
 width = 10,  
 height = 10) # For saving the variable plot  
ggsave(plot= variable\_plot[[4]],  
 filename="varplot4.pdf",  
 width = 10,  
 height = 10) # For saving the variable plot  
ggsave(plot= variable\_plot[[5]],  
 filename="varplot5.pdf",  
 width = 10,  
 height = 10) # For saving the variable plot  
  
  
  
ggsave(plot= ind\_plot,  
 filename="ind\_contrib.pdf",  
 width = 10,  
 height = 10) # For saving the individual contribution plot  
  
  
ggsave(plot= biplot,  
 filename="biplot.pdf",  
 width = 10,  
 height = 10) # For saving the variable plot

The below error may apper if there is a file with the same filename that we are saving like Correlation\_matrix.docx, covariance\_table.docx, eigen\_table.docx and variable\_stats.docx. If the error appears, delete the file from the folder and rerun the code.



**Saving as a combined file**

Saving the combined file is tedious and may require changing values in with the code, that may lead to error sometimes. So, I recommend using “[*PBperfect*](https://allanbiotools.shinyapps.io/pbperfect/)” to run your analysis and customize the plots with intuitive UI.