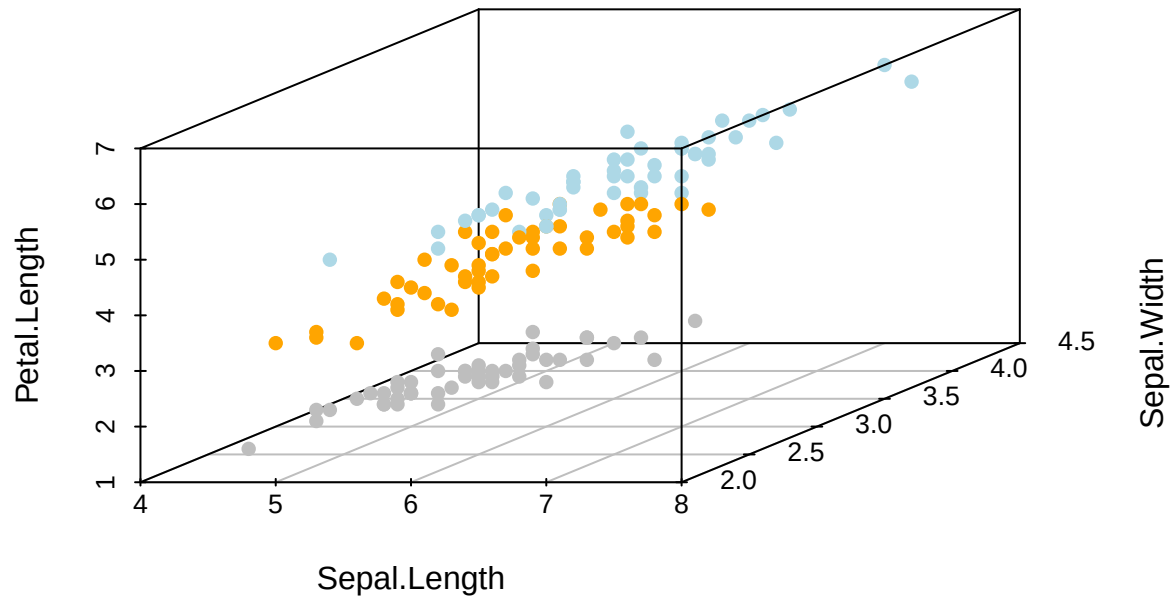
Dimensionality Reduction & Clustering Exercises - IRIS by Alexandre Huet

```
## Installing package into 'C:/Users/CYTech Student/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'webshot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\CYTech Student\AppData\Local\Temp\RtmpwjYx9r\downloaded_packages
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
library(scatterplot3d)
data(iris)

colors <- c("gray", "orange", "lightblue")[as.numeric(iris$Species)]

scatterplot3d(
  x = iris$Sepal.Length,
  y = iris$Sepal.Width,
  z = iris$Petal.Length,

  color = colors,

  pch = 16,
  xlab = "Sepal.Length",
  ylab = "Sepal.Width",
  zlab = "Petal.Length",
  angle = 45
)
```




```
#More interactive with plotly

# Load necessary libraries
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

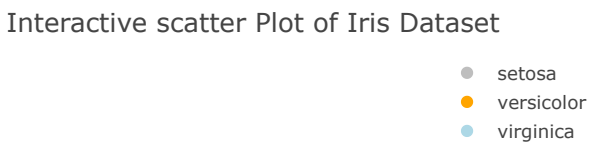
```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```r
data(iris)

#Interactive 3D plot
fig <- plot_ly(
  data = iris,
  x = ~Sepal.Length,
  y = ~Petal.Length,
  z = ~Sepal.Width,
  color = ~Species,  # Color points by species
  colors = c("gray", "orange", "lightblue")
) %>%
  add_markers() %>%
  layout(
    scene = list(
      xaxis = list(title = 'Sepal Length'),
      yaxis = list(title = 'Petal Length'),
      zaxis = list(title = 'Sepal Width')
    ),
    title = "Interactive scatter Plot of Iris Dataset"
  )

fig
```

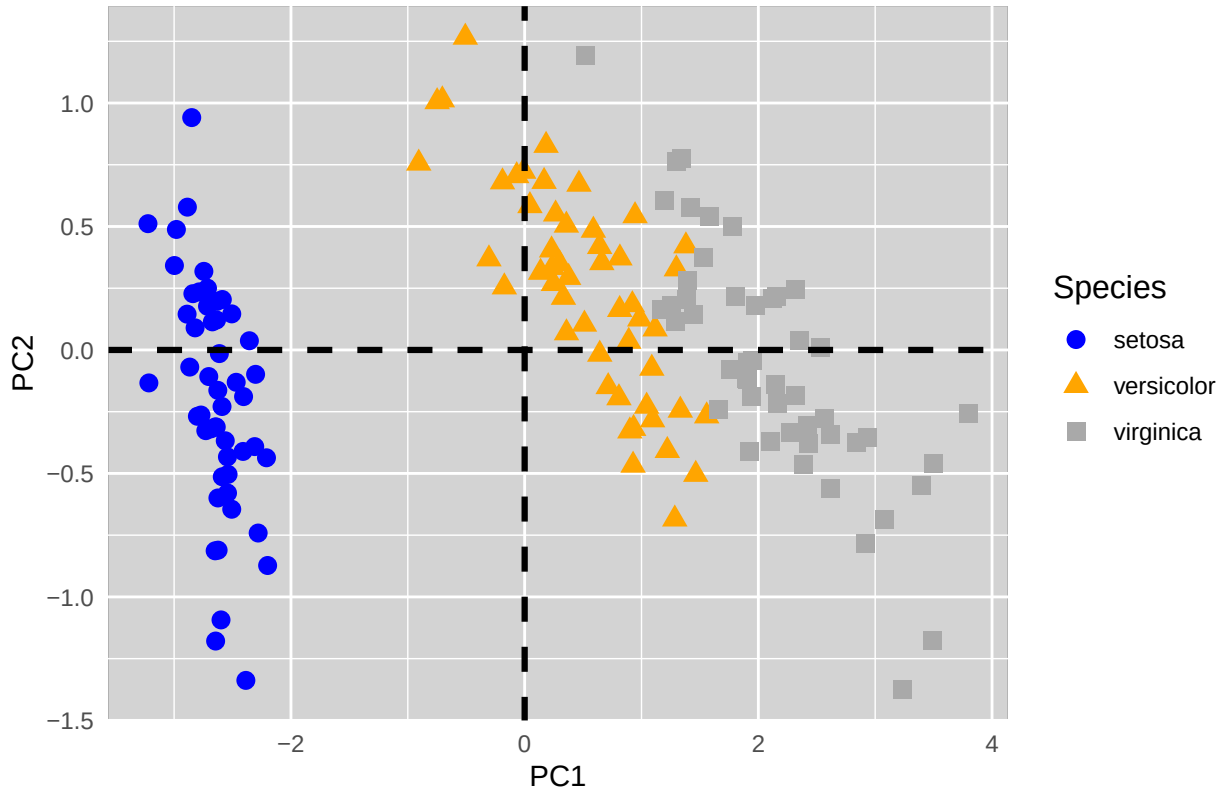



```
#None standardisation version
library(ggplot2)
```

```r
data(iris)

centered_data <- scale(iris[, 1:4], center = TRUE, scale = FALSE)
pca_res <- prcomp(centered_data)

df <- as.data.frame(pca_res$x)
df$Species <- iris$Species

# Plot PCA without standardization
p1 <- ggplot(df, aes(x = PC1, y = PC2, color = Species, shape = Species)) +
  geom_point(size = 3) +
  labs(title = "",
       x = "PC1",
       y = "PC2") +

  theme_minimal() +

  scale_color_manual(values = c("blue", "orange", "darkgray")) +

  scale_shape_manual(values = c(16, 17, 15)) +

  geom_hline(yintercept = 0, linetype = "dashed", size = 1.1) +

  geom_vline(xintercept = 0, linetype = "dashed", size = 1.1) +

  theme(legend.title = element_text(size = 12), legend.position = "right",
        panel.background = element_rect(fill = "lightgrey"),
        panel.grid.major = element_line(color = "white"),
        panel.grid.minor = element_line(color = "white"),
        panel.border = element_rect(color = "white", fill = NA))  # Set border to white
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

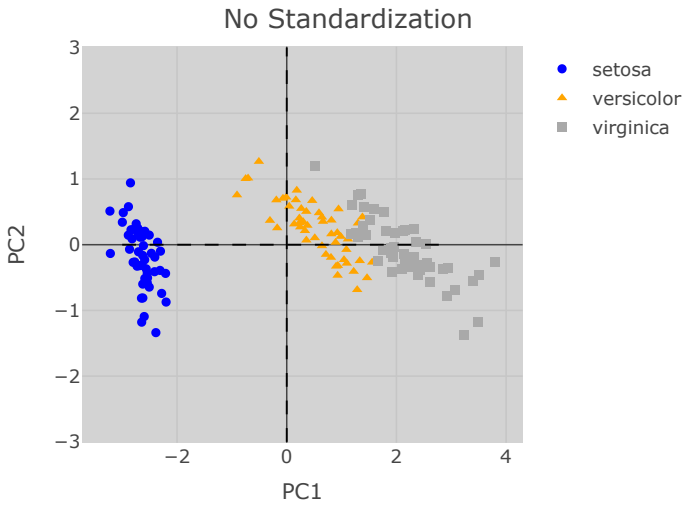
```r
print(p1)
```




```r
library(plotly)

data(iris)

centered_data <- scale(iris[, 1:4], center = TRUE, scale = FALSE)
pca_res <- prcomp(centered_data)

df <- as.data.frame(pca_res$x)
df$Species <- iris$Species

p1_plotly <- plot_ly(df,
                     x = ~PC1,
                     y = ~PC2,
                     color = ~Species,
                     colors = c("blue", "orange", "darkgray"),

                     symbol = ~Species,

                     symbols = c("circle", "triangle-up", "square"),

                     text = ~paste("Species:", Species, "PC1:", round(PC1, 2), "PC2:", round(PC2, 2)),

                     mode = "markers") %>%

  layout(title = "No Standardization",
```

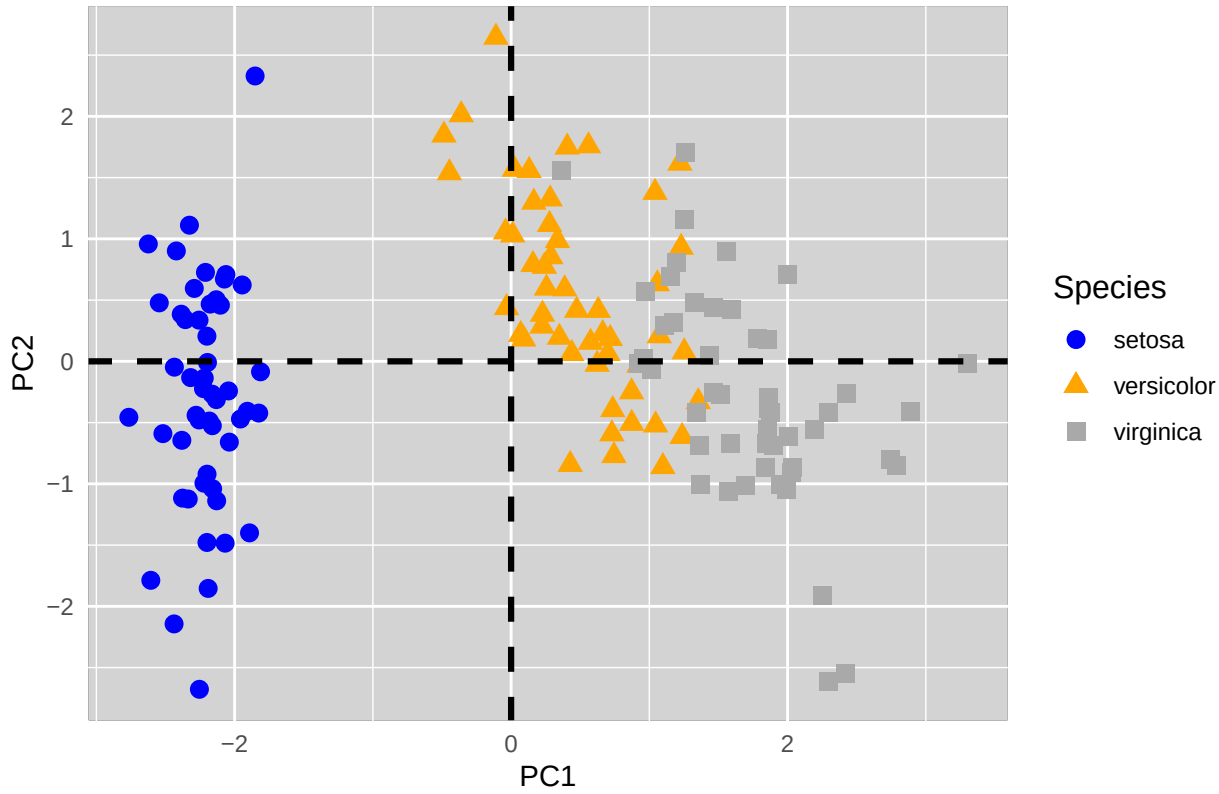
```
         xaxis = list(title = "PC1"),

         yaxis = list(title = "PC2"),

         plot_bgcolor = "lightgrey",

         paper_bgcolor = "white",

         shapes = list(
           list(type = "line", x0 = 0, x1 = 0, y0 = -3, y1 = 3, line = list(color = "black", dash = 'das

           list(type = "line", x0 = -3, x1 = 3, y0 = 0, y1 = 0, line = list(color = "black", dash = 'das
         ))

# Render the plot
p1_plotly
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plotly.com/r/reference/#scatter
```




```
#standardised version
standardized_data <- scale(iris[, 1:4])
```

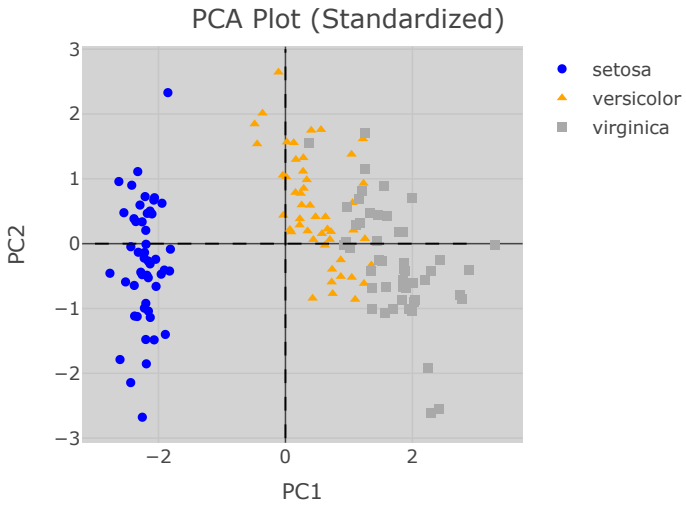
```r
pca_res_std <- prcomp(standardized_data)

#dataframe
df_std <- as.data.frame(pca_res_std$x)
df_std$Species <- iris$Species

p2 <- ggplot(df_std, aes(x = PC1, y = PC2, color = Species, shape = Species)) +
  geom_point(size = 3) +
  labs(title = "",
       x = "PC1",
       y = "PC2") +

  theme_minimal() +

  scale_color_manual(values = c("blue", "orange", "darkgray")) +

  scale_shape_manual(values = c(16, 17, 15)) +

  geom_hline(yintercept = 0, linetype = "dashed", size = 1.1) +

  geom_vline(xintercept = 0, linetype = "dashed", size = 1.1) +

  theme(legend.title = element_text(size = 12), legend.position = "right",
        panel.background = element_rect(fill = "lightgrey"),
        panel.grid.major = element_line(color = "white"),
        panel.grid.minor = element_line(color = "white"),
        panel.border = element_rect(color = "white", fill = NA))


print(p2)
```




```
library(plotly)

data(iris)

standardized_data <- scale(iris[, 1:4])
pca_res_standardized <- prcomp(standardized_data)

df_standardized <- as.data.frame(pca_res_standardized$x)
df_standardized$Species <- iris$Species

p2_plotly <- plot_ly(df_standardized,
                     x = ~PC1,
                     y = ~PC2,
                     color = ~Species,
                     colors = c("blue", "orange", "darkgray"),
                     symbol = ~Species,
                     symbols = c("circle", "triangle-up", "square"),
                     text = ~paste("Species:", Species, "<br>PC1:", round(PC1, 2), "<br>PC2:", round(PC
                     mode = "markers") %>%
  layout(title = "PCA Plot (Standardized)",
         xaxis = list(title = "PC1"),
         yaxis = list(title = "PC2"),
         plot_bgcolor = "lightgrey",
         paper_bgcolor = "white",
         shapes = list(
           list(type = "line", x0 = 0, x1 = 0, y0 = -3, y1 = 3, line = list(color = "black", dash = 'da
```

```
          list(type = "line", x0 = -3, x1 = 3, y0 = 0, y1 = 0, line = list(color = "black", dash = 'das
        ))

p2_plotly
```

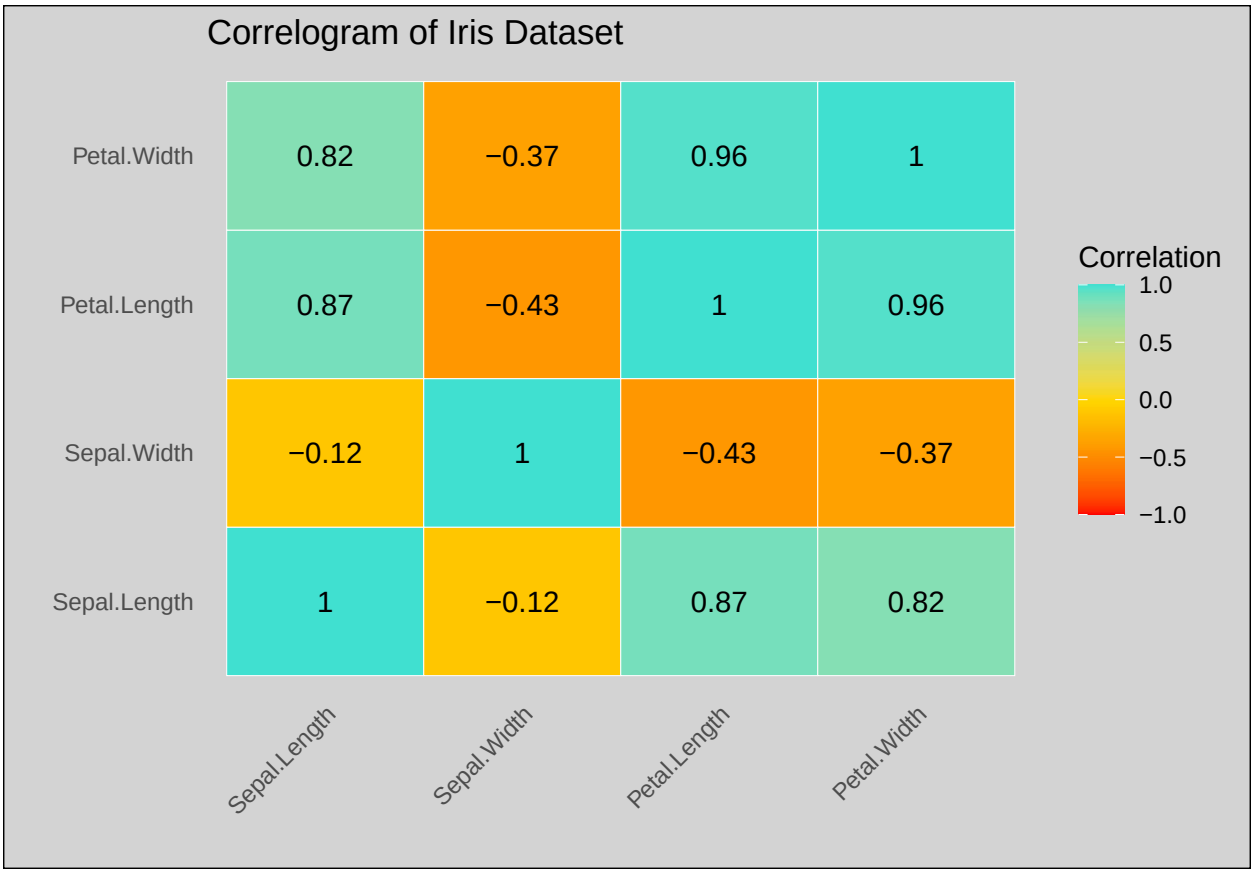
```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plotly.com/r/reference/#scatter
```


PCA Plot (Standardized)
PC2
PC1
3
2
1
0
−1
−2
−3
−2
0
2
setosa
versicolor
virginica

```r
install.packages("ggplot2")
```

```
## Warning: package 'ggplot2' is in use and will not be installed
```

```r
library(ggplot2)

data(iris)

cor_matrix <- cor(iris[, 1:4])

cor_df <- as.data.frame(as.table(cor_matrix))

colnames(cor_df) <- c("Var1", "Var2", "value")

ggplot(cor_df, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(value, 2)), size = 4, color = "black") +
  scale_fill_gradient2(low = "red", high = "turquoise", mid = "gold",
                       limit = c(-1, 1), name = "Correlation") +
  labs(title = "Correlogram of Iris Dataset", x = "", y = "") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        plot.background = element_rect(fill = "lightgrey"),
        axis.text.x = element_text(angle = 45, hjust = 1))
```



```r
# Load necessary libraries
library(FactoMineR)
```

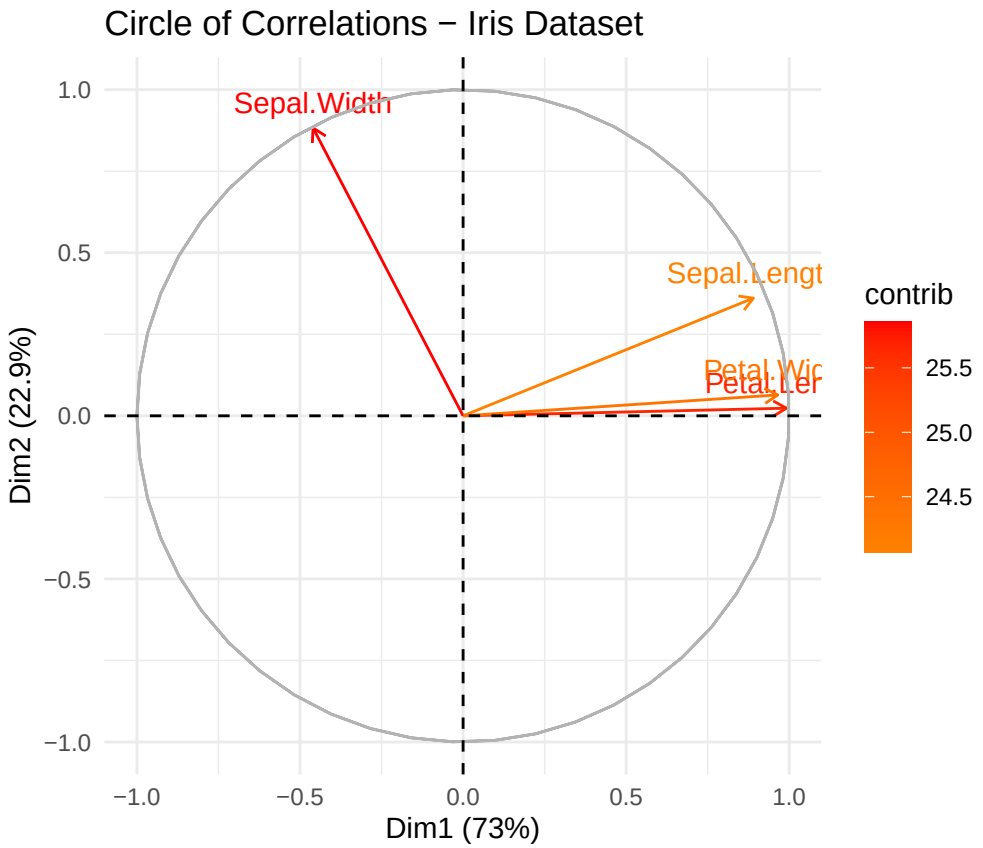
```
## Warning: package 'FactoMineR' was built under R version 4.3.3
```

```r
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.3.3
```

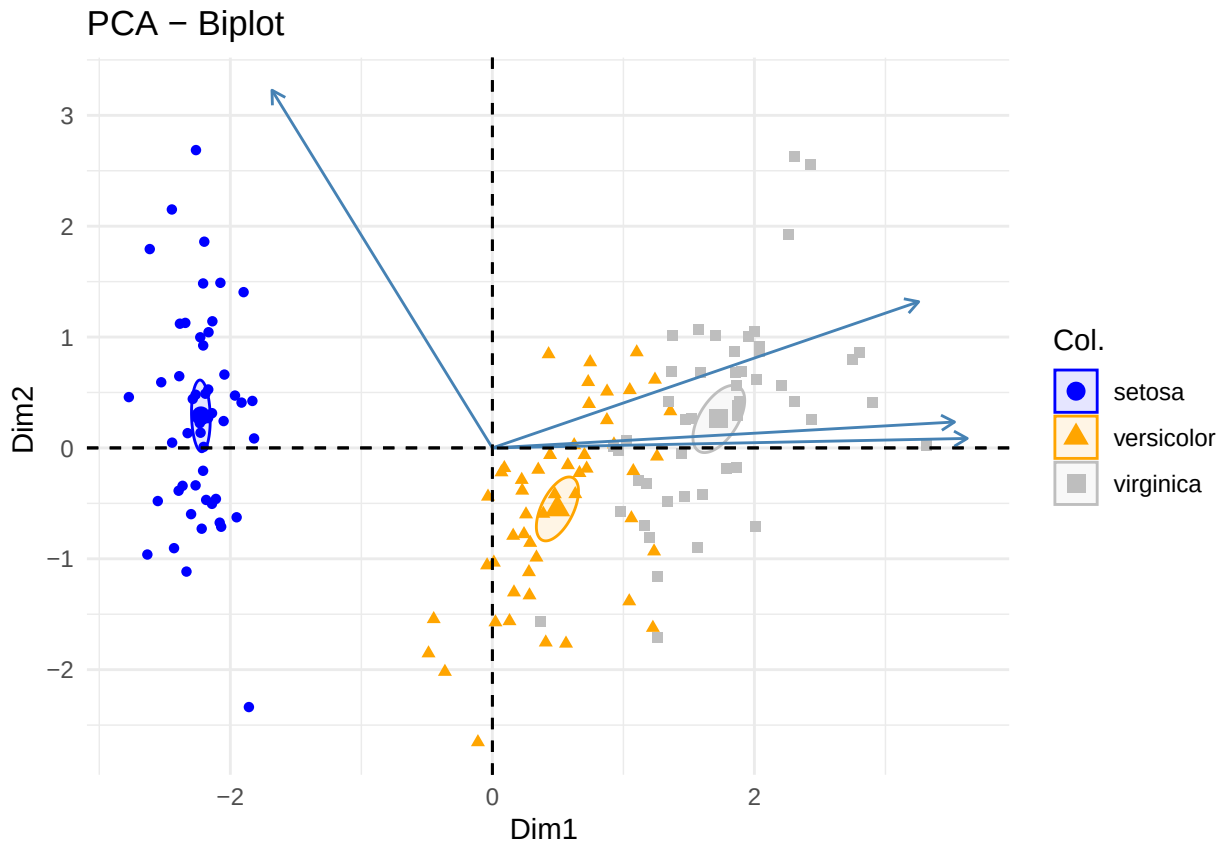
```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
iris_stand <- scale(iris[, -5])

#PCA on the standardized dataset
pca_res <- PCA(iris_stand, graph = FALSE)

fviz_pca_var(pca_res, col.var = "contrib",  # Color based on contribution
             title = "Circle of Correlations - Iris Dataset") +
  theme_minimal() +
  scale_color_gradient2(low = "blue", mid = "yellow", high = "red",
                        midpoint = median(pca_res$var$contrib))
```



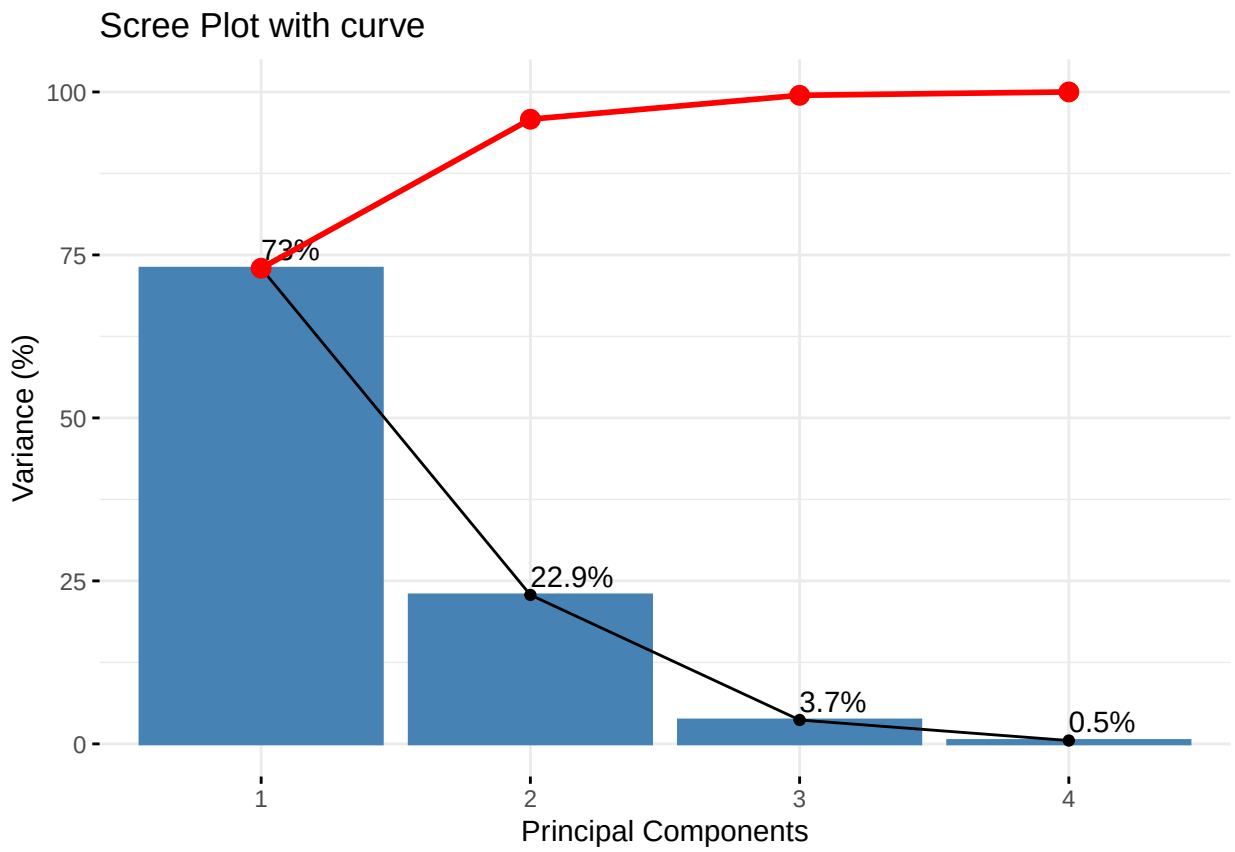
```r
#Note: made a gradient to see how much contribution each variable has
```

```r
# Load necessary libraries
library(FactoMineR)
library(factoextra)

# Perform PCA
pca_result <- PCA(iris[, -5], graph = FALSE)

fviz_pca_biplot(pca_result,
                geom.ind = "point",
                geom.var = "arrow",
                col.ind = iris$Species,
                addEllipses = TRUE,
                ellipse.type = "confidence",
                palette = c("blue", "orange", "gray"),
                title = "PCA - Biplot") +
  labs(x = "Dim1 ", y = "Dim2 ") +
  theme_minimal()
```




```r
pca_result <- PCA(iris[, -5], graph = FALSE)

fviz_eig(pca_result, addlabels = TRUE) +
  labs(title = "Scree Plot with curve",
       x = "Principal Components",
```

```
       y = "Variance (%)") +

  geom_line(aes(x = 1:length(pca_result$eig[,1]),
                y = cumsum(pca_result$eig[,2])),
            color = "red", size = 1) +

  geom_point(aes(x = 1:length(pca_result$eig[,1]),
                 y = cumsum(pca_result$eig[,2])),
             color = "red", size = 3)
```




```
library(FactoMineR)

pca_result <- PCA(iris[, -5], graph = FALSE)

eigenvalues <- pca_result$eig

eigen_table <- data.frame(
  Component = 1:nrow(eigenvalues),
  `Initial Eigenvalues` = round(eigenvalues[, 1], 3),
  `Variance` = round(eigenvalues[, 2], 3),
  `Cumulated Variance` = round(eigenvalues[, 3], 3)
)

print(eigen_table)
```

```
##        Component Initial.Eigenvalues Variance Cumulated.Variance
## comp 1         1               2.918   72.962             72.962
## comp 2         2               0.914   22.851             95.813
## comp 3         3               0.147    3.669             99.482
## comp 4         4               0.021    0.518            100.000
```

Ex9.)

```
data <- iris[, -5]

pca_result <- prcomp(data, scale = TRUE)

loadings <- pca_result$rotation %*% diag(pca_result$sdev)

print(loadings)
```

```
##                    [,1]        [,2]        [,3]        [,4]
## Sepal.Length  0.8901688 -0.36082989  0.27565767  0.03760602
## Sepal.Width  -0.4601427 -0.88271627 -0.09361987 -0.01777631
## Petal.Length  0.9915552 -0.02341519 -0.05444699 -0.11534978
## Petal.Width   0.9649790 -0.06399985 -0.24298265  0.07535950
```

```
strongest_cp1 <- loadings[which.max(abs(loadings[, 1])), ]
print(strongest_cp1)
```

```
## [1]  0.99155518 -0.02341519 -0.05444699 -0.11534978
```

Ex10.)

```
data <- iris[, -5]

pca_result <- prcomp(data, scale = TRUE)

loadings <- pca_result$rotation %*% diag(pca_result$sdev)

loadings_df <- as.data.frame(loadings)

#Identify the variable with the strongest saturation for each principal component
strongest_saturations <- apply(loadings_df, 2, function(x) rownames(loadings_df)[which.max(abs(x))])

results_table <- data.frame(
  Principal_Component = names(strongest_saturations),
  Strongest_Variable = strongest_saturations
)

print(results_table)
```

```
##    Principal_Component Strongest_Variable
## V1                  V1       Petal.Length
## V2                  V2        Sepal.Width
## V3                  V3       Sepal.Length
## V4                  V4       Petal.Length
```

Ex11.) Saturation and Arrow Length

Length of Arrows: Indicates strength of correlation. Long Arrows: Represent strong saturation with principal components. Short Arrows: Indicate weak correlation.

Orientation of Arrows

Direction: Shows relationship with principal components. Positive Contributions: Arrows pointing in the same direction as CP axis. Negative Contributions: Arrows pointing in the opposite direction.

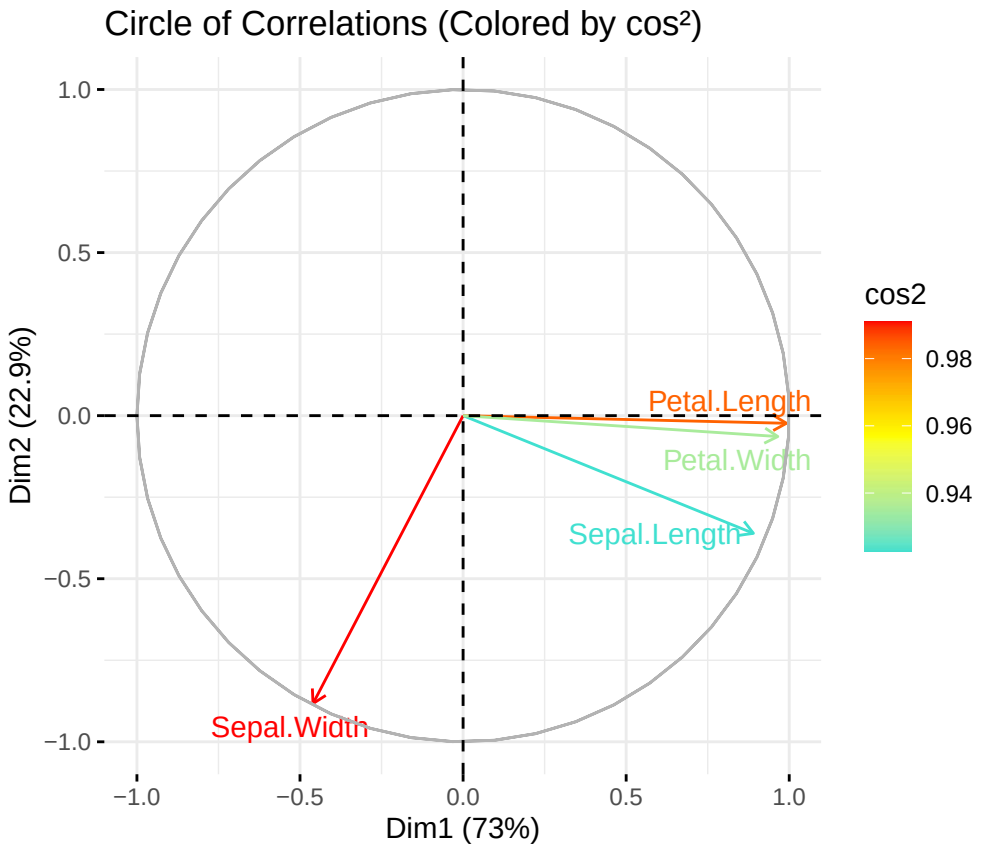
Interpretation of Relationships

Understanding Contributions: Strong saturation indicates significant role in variance explanation. Inter connectedness: Close arrows suggest positive correlation between variables.

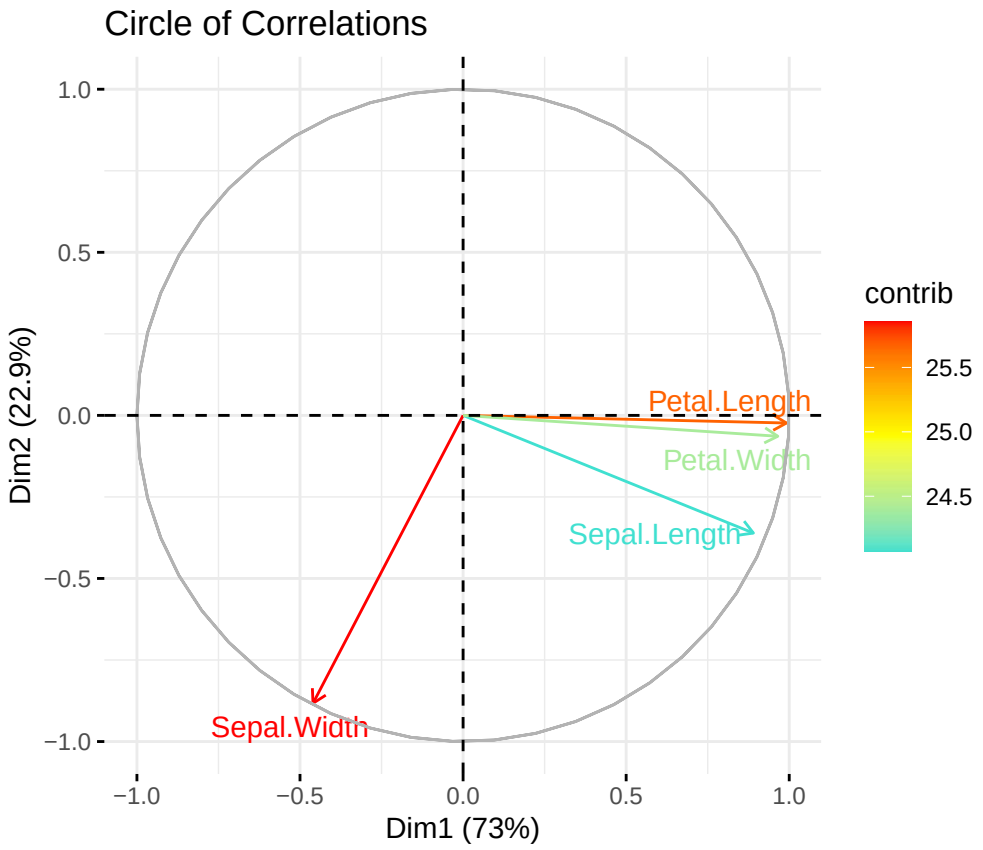
In other words... summary Arrow length = saturation strength. Arrow direction = positive/negative contribution to principal components.

Ex12.)

```
library(FactoMineR)
library(factoextra)

#based on cos^2
fviz_pca_var(pca_result, col.var = "cos2",
             gradient.cols = c("turquoise", "yellow", "red"),
             repel = TRUE) +
  labs(title = "Circle of Correlations (Colored by cos²)")
```



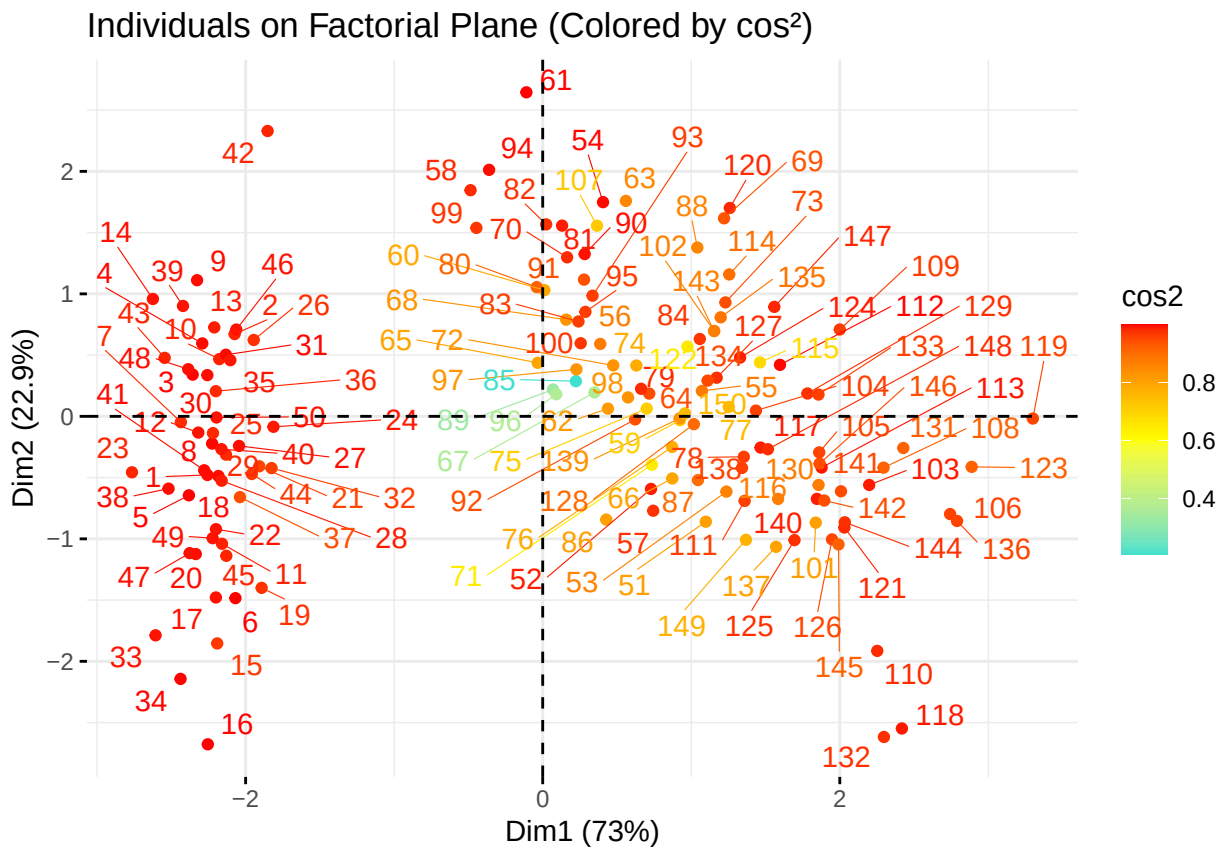
```r
#Circle of correlations with color based on contributions
fviz_pca_var(pca_result, col.var = "contrib",
             gradient.cols = c("turquoise", "yellow", "red"),
             repel = TRUE) +
  labs(title = "Circle of Correlations")
```



The $\cos^2$ metric indicates how well each variable is represented by the principal components, with higher values showing a closer alignment to the component axes, meaning the variable's variance is well captured. Contribution, on the other hand, measures how much each variable shapes a particular component, indicating its influence on the component's direction. While $\cos^2$ focuses on representation quality, contribution emphasizes the impact of each variable on defining the component.

Ex13

```r
library(FactoMineR)
library(factoextra)

fviz_pca_ind(pca_result, col.ind = "cos2",
             gradient.cols = c("turquoise", "yellow", "red"),
             repel = TRUE) +
  labs(title = "Individuals on Factorial Plane (Colored by cos²)")
```




```
#Extract contributions of individuals to first two axes
contrib_ind <- pca_res$ind$contrib[, 1:2]

contrib_table <- as.data.frame(contrib_ind)
colnames(contrib_table) <- c("Contribution to PC1", "Contribution to PC2")

print(contrib_table)
```

```
##     Contribution to PC1 Contribution to PC2
## 1          1.171580e+00        1.680655e-01
## 2          9.891845e-01        3.314667e-01
## 3          1.276816e+00        8.526419e-02
## 4          1.207737e+00        2.602978e-01
## 5          1.304631e+00        3.051656e-01
## 6          9.841236e-01        1.617488e+00
## 7          1.364464e+00        1.655648e-03
## 8          1.138852e+00        3.631904e-02
## 9          1.245057e+00        9.073044e-01
## 10         1.089896e+00        1.604423e-01
## 11         1.071990e+00        7.944959e-01
## 12         1.235998e+00        1.291703e-02
## 13         1.124214e+00        3.872730e-01
## 14         1.583742e+00        6.742993e-01
## 15         1.104326e+00        2.523484e+00
## 16         1.169007e+00        5.263227e+00
## 17         1.113231e+00        1.605415e+00
```

```
## 18         1.095913e+00        1.742924e-01
## 19         8.233861e-01        1.439834e+00
## 20         1.254385e+00        9.277913e-01
## 21         8.371047e-01        1.219237e-01
## 22         1.112651e+00        6.228825e-01
## 23         1.758208e+00        1.532253e-01
## 24         7.555391e-01        5.339181e-03
## 25         1.133062e+00        1.374045e-02
## 26         8.702431e-01        2.854745e-01
## 27         9.610474e-01        4.277260e-02
## 28         1.074235e+00        2.026822e-01
## 29         1.045682e+00        7.155516e-02
## 30         1.172158e+00        8.319405e-02
## 31         1.046228e+00        1.856695e-01
## 32         7.663165e-01        1.309347e-01
## 33         1.561980e+00        2.346322e+00
## 34         1.366864e+00        3.373797e+00
## 35         1.016960e+00        1.544702e-01
## 36         1.113454e+00        3.098384e-02
## 37         9.554283e-01        3.192156e-01
## 38         1.459063e+00        2.558709e-01
## 39         1.348437e+00        5.962905e-01
## 40         1.075358e+00        5.273048e-02
## 41         1.194214e+00        1.423092e-01
## 42         7.886749e-01        3.984922e+00
## 43         1.489595e+00        1.674178e-01
## 44         8.815162e-01        1.627170e-01
## 45         1.043236e+00        9.516004e-01
## 46         9.785494e-01        3.687667e-01
## 47         1.299059e+00        9.156243e-01
## 48         1.309586e+00        1.088123e-01
## 49         1.135386e+00        7.263971e-01
## 50         1.109448e+00        6.195362e-05
## 51         2.772937e-01        5.431777e-01
## 52         1.221757e-01        2.578810e-01
## 53         3.517859e-01        2.770315e-01
## 54         3.792875e-02        2.244953e+00
## 55         2.642103e-01        3.168336e-02
## 56         3.451041e-02        2.567277e-01
## 57         1.273045e-01        4.358417e-01
## 58         5.424787e-02        2.502829e+00
## 59         1.966769e-01        7.574657e-04
## 60         2.982306e-05        7.798382e-01
## 61         2.773852e-03        5.137759e+00
## 62         4.436317e-02        2.922062e-03
## 63         7.217543e-02        2.271443e+00
## 64         1.182730e-01        2.529427e-02
## 65         2.541344e-04        1.405670e-01
## 66         1.750530e-01        1.890135e-01
## 67         2.802269e-02        2.810869e-02
## 68         5.761099e-03        4.576183e-01
## 69         3.428372e-01        1.919466e+00
## 70         6.212765e-03        1.237589e+00
## 71         1.243050e-01        1.147073e-01
```

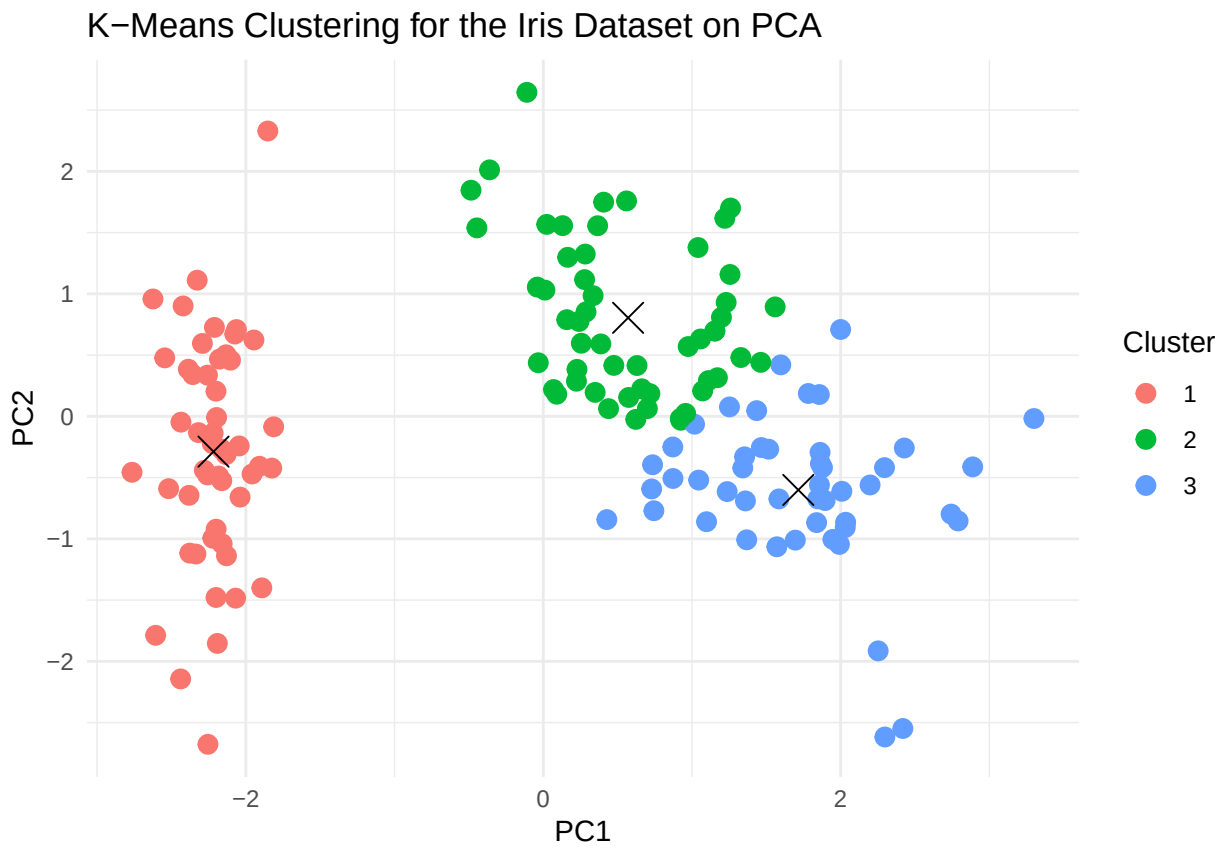
```
## 72         5.181878e-02       1.270244e-01
## 73         3.479405e-01       6.353522e-01
## 74         9.148759e-02       1.264573e-01
## 75         1.127824e-01       2.932841e-03
## 76         1.746000e-01       4.587544e-02
## 77         3.606456e-01       4.353241e-03
## 78         4.215101e-01       8.006110e-02
## 79         1.009559e-01       3.722954e-02
## 80         3.702260e-04       8.175402e-01
## 81         3.907804e-03       1.780169e+00
## 82         1.256419e-04       1.803499e+00
## 83         1.332666e-02       4.406326e-01
## 84         2.571921e-01       2.930298e-01
## 85         1.145943e-02       6.040177e-02
## 86         4.206739e-02       5.215066e-01
## 87         2.512321e-01       1.987812e-01
## 88         2.492254e-01       1.395036e+00
## 89         1.106171e-03       3.514231e-02
## 90         1.835633e-02       1.288873e+00
## 91         1.779098e-02       9.149687e-01
## 92         8.910690e-02       4.530537e-04
## 93         2.587009e-02       7.125528e-01
## 94         2.996446e-02       2.973877e+00
## 95         1.902395e-02       5.340992e-01
## 96         1.906636e-03       2.394565e-02
## 97         1.184513e-02       1.080660e-01
## 98         7.588915e-02       1.749455e-02
## 99         4.577829e-02       1.738304e+00
## 100        1.505583e-02       2.615693e-01
## 101        7.772113e-01       5.525952e-01
## 102        3.062511e-01       3.562384e-01
## 103        1.110891e+00       2.303758e-01
## 104        4.737675e-01       1.610328e-03
## 105        7.969220e-01       6.349274e-02
## 106        1.729841e+00       4.672746e-01
## 107        3.076971e-02       1.778417e+00
## 108        1.210949e+00       1.287011e-01
## 109        9.198318e-01       3.691671e-01
## 110        1.166489e+00       2.691581e+00
## 111        4.250988e-01       3.500332e-01
## 112        5.867354e-01       1.297048e-01
## 113        8.107097e-01       1.282016e-01
## 114        3.627186e-01       9.852693e-01
## 115        4.920299e-01       1.426679e-01
## 116        5.775450e-01       3.335462e-01
## 117        4.945719e-01       4.765889e-02
## 118        1.344772e+00       4.767541e+00
## 119        2.503732e+00       2.305993e-04
## 120        3.648238e-01       2.124641e+00
## 121        9.484988e-01       6.046122e-01
## 122        2.184791e-01       2.384417e-01
## 123        1.917969e+00       1.247945e-01
## 124        4.060326e-01       1.693175e-01
## 125        6.607271e-01       7.498200e-01
```

```
## 126        8.724563e-01        7.407599e-01
## 127        3.154291e-01        7.301395e-02
## 128        2.380997e-01        3.019893e-03
## 129        7.305574e-01        2.560398e-02
## 130        7.933721e-01        2.306056e-01
## 131        1.355462e+00        4.903441e-02
## 132        1.213568e+00        5.030886e+00
## 133        7.925683e-01        2.325227e-02
## 134        2.835535e-01        6.258264e-02
## 135        3.302937e-01        4.800952e-01
## 136        1.789303e+00        5.354394e-01
## 137        5.675483e-01        8.328428e-01
## 138        4.140263e-01        1.301544e-01
## 139        1.953747e-01        2.163569e-04
## 140        7.835243e-01        3.334311e-01
## 141        9.272946e-01        2.748673e-01
## 142        8.261745e-01        3.468260e-01
## 143        3.062511e-01        3.562384e-01
## 144        9.511464e-01        5.489182e-01
## 145        9.120198e-01        8.028580e-01
## 146        7.992200e-01        1.092179e-01
## 147        5.591717e-01        5.864482e-01
## 148        5.285732e-01        5.280510e-02
## 149        4.304832e-01        7.458799e-01
## 150        2.108071e-01        4.318091e-04
```

The contribution of an individual to a principal component (for instance PC1 or PC2) quantifies how much that individual influences the component's direction. Higher contributions indicate that the individual has a stronger effect on shaping the component's axis. This is useful in finding which individuals are most significant in defining the structure of the data along the principal components.

Ex 16

```r
pca_scores <- pca_result$x[, 1:2]
colnames(pca_scores) <- c("PC1", "PC2")

#K-Means Clustering
k <- 3
kmeans_result <- kmeans(pca_scores, centers = k)

pca_df <- data.frame(pca_scores, Cluster = as.factor(kmeans_result$cluster), Species = iris$Species)

centers <- as.data.frame(kmeans_result$centers)
ggplot(pca_df, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point(size = 3) +
  geom_point(data = centers, aes(x = PC1, y = PC2), color = "black", size = 5, shape = 4) +  # Add cent
  labs(title = "K-Means Clustering for the Iris Dataset on PCA",
       x = "PC1",
       y = "PC2") +
  theme_minimal()
```




```
#Calculate means of quantitative variables for each cluster
means_clusters <- aggregate(iris[, -5], by = list(Cluster = kmeans_result$cluster), FUN = mean)

#Calculate proportions of species in each cluster
species_distribution <- as.data.frame(table(kmeans_result$cluster, iris$Species))
proportions <- species_distribution %>%
  group_by(Var1) %>%
  mutate(Percentage = Freq / sum(Freq) * 100)

print("Means of Quantitative Variables for Each Cluster:")
```

```
## [1] "Means of Quantitative Variables for Each Cluster:"
```

```
print(means_clusters)
```

```
##   Cluster Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1       1     5.006000    3.428000     1.462000    0.246000
## 2       2     5.801887    2.673585     4.369811    1.413208
## 3       3     6.780851    3.095745     5.510638    1.972340
```

```
print("Proportions of Species in Each Cluster:")
```

```
## [1] "Proportions of Species in Each Cluster:"
```

```r
print(proportions)
```

```
## # A tibble: 9 x 4
## # Groups:   Var1 [3]
##   Var1  Var2        Freq Percentage
##   <fct> <fct>      <int>      <dbl>
## 1 1     setosa        50      100
## 2 2     setosa         0        0
## 3 3     setosa         0        0
## 4 1     versicolor     0        0
## 5 2     versicolor    39       73.6
## 6 3     versicolor    11       23.4
## 7 1     virginica      0        0
## 8 2     virginica     14       26.4
## 9 3     virginica     36       76.6
```

Ex 17

```r
library(dplyr)
```

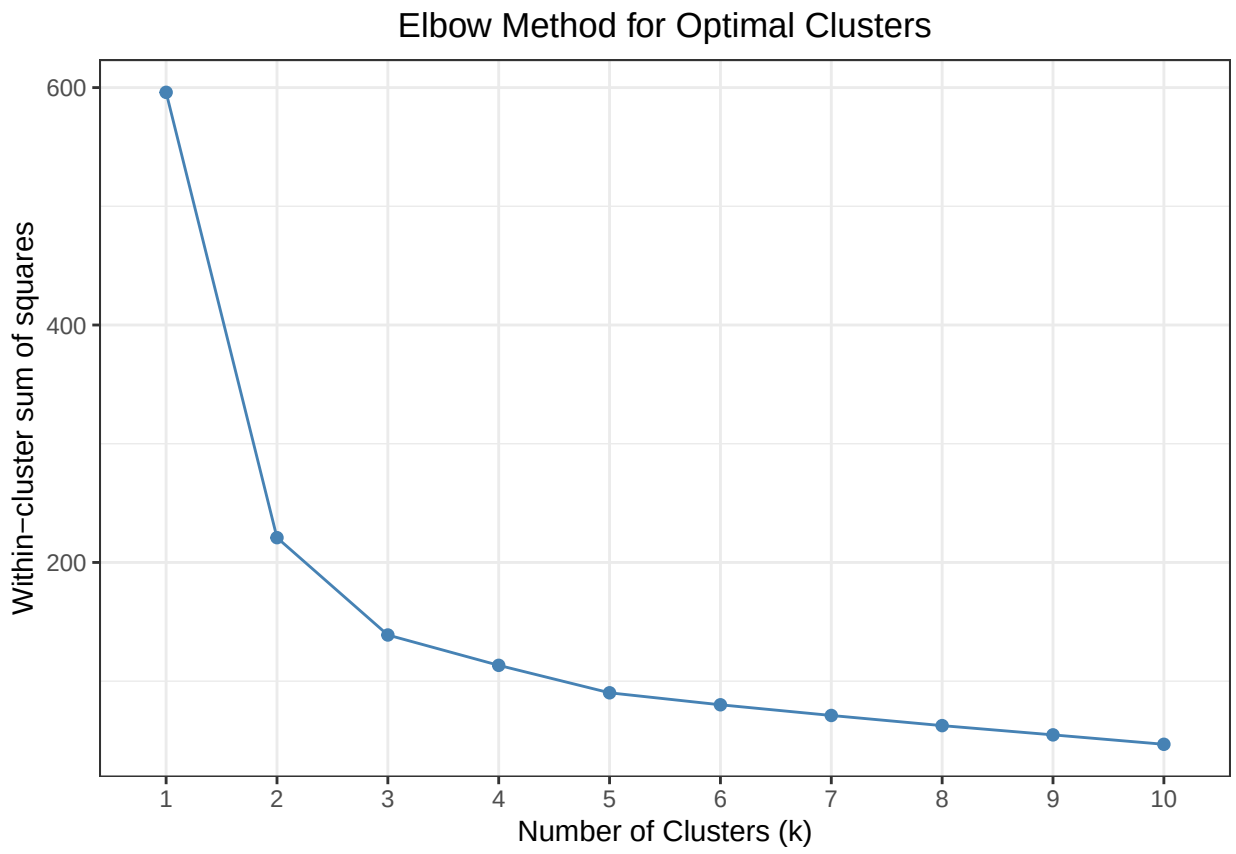
```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
iris_data <- iris %>% select(-Species)

# Rescale the data (standardizing)
iris_rescaled <- scale(iris_data)

# Visualize the Elbow method using WSS
fviz_nbclust(
  x = iris_rescaled,
  FUNcluster = kmeans,
  method = "wss",
  k.max = 10,
  nstart = 10
) +
  labs(title = "Elbow Method for Optimal Clusters") +
  xlab("Number of Clusters (k)") +
  ylab("Within-cluster sum of squares") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```




```
install.packages("vegan")
```

```
## Installing package into 'C:/Users/CYTech Student/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'vegan' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'vegan'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\CYTech
## Student\AppData\Local\R\win-library\4.3\00LOCK\vegan\libs\x64\vegan.dll to
## C:\Users\CYTech
## Student\AppData\Local\R\win-library\4.3\vegan\libs\x64\vegan.dll: Permission
## denied

## Warning: restored 'vegan'

##
## The downloaded binary packages are in
##  C:\Users\CYTech Student\AppData\Local\Temp\RtmpwjYx9r\downloaded_packages
```

```r
library(vegan)
```

```
## Warning: package 'vegan' was built under R version 4.3.3
```

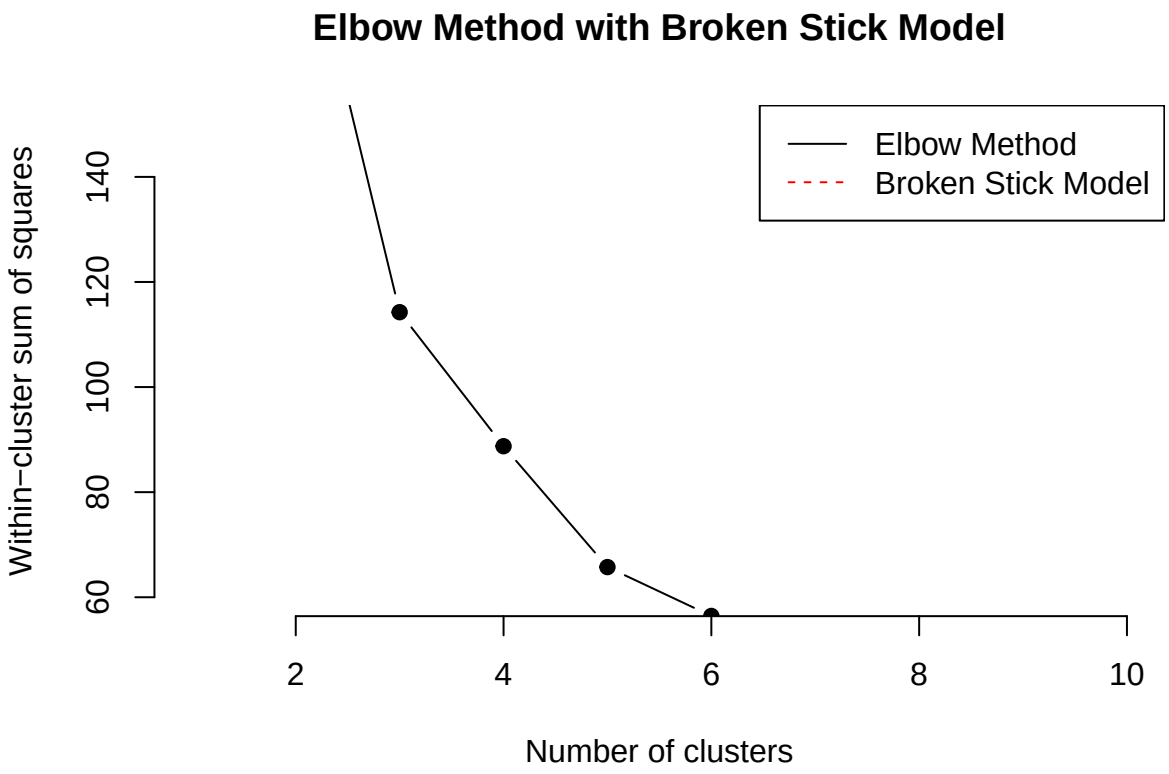
```
## Loading required package: permute
```

```
## Warning: package 'permute' was built under R version 4.3.3
```

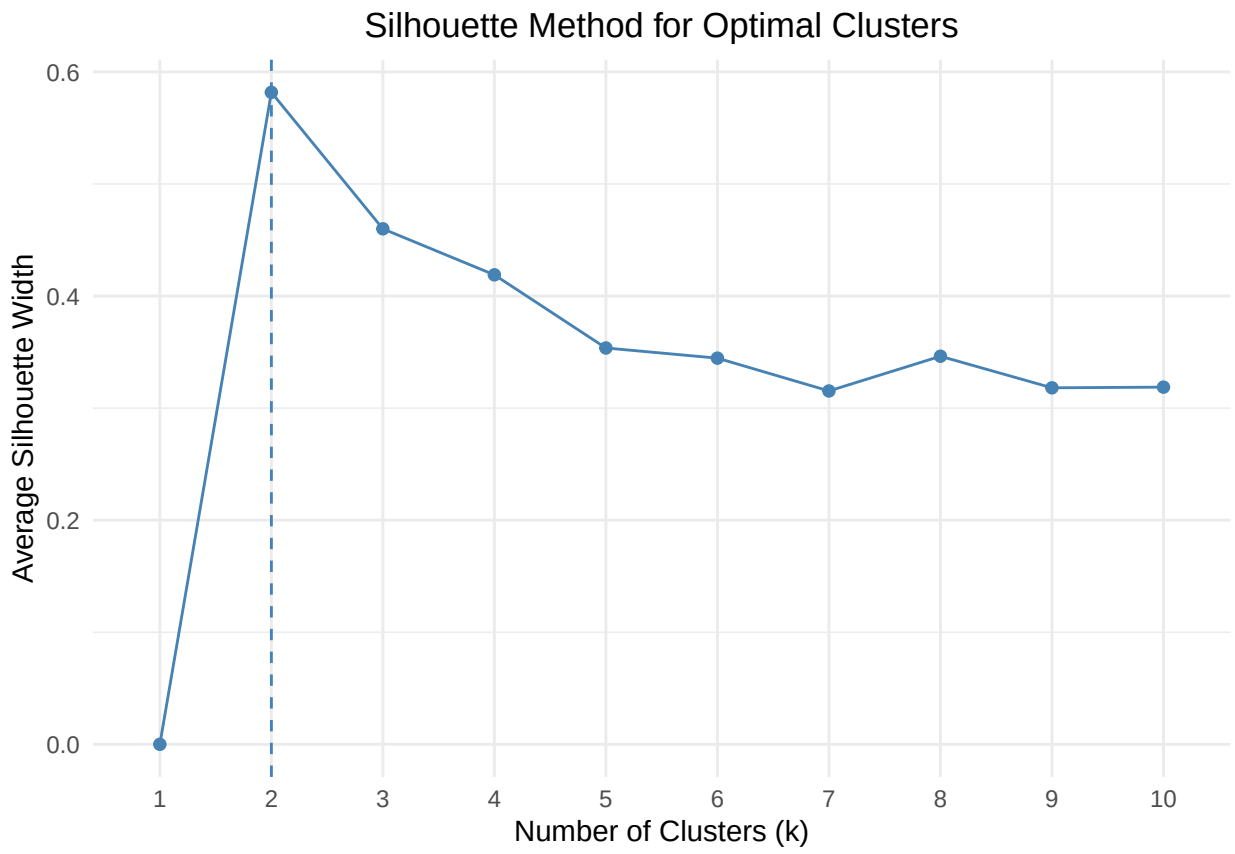
```
## Loading required package: lattice
```

```
## This is vegan 2.6-8
```

```r
wcss <- vector()

for (i in 1:10) {
  kmeans_result <- kmeans(pca_scores, centers = i, nstart = 25)
  wcss[i] <- kmeans_result$tot.withinss
}

stick <- bstick(n = 10)

plot(1:10, wcss, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters",
     ylab = "Within-cluster sum of squares",
     main = "Elbow Method with Broken Stick Model",
     ylim = c(60, 150))
lines(1:10, stick, type = "b", col = "red", lty = 2)
legend("topright", legend = c("Elbow Method", "Broken Stick Model"),
       col = c("black", "red"), lty = c(1, 2))
```




```r
library(factoextra)
library(ggplot2)

fviz_nbclust(
  x = iris_rescaled,
  FUNcluster = kmeans,
  method = "silhouette",
  k.max = 10
) +
  labs(title = "Silhouette Method for Optimal Clusters",
       x = "Number of Clusters (k)",
       y = "Average Silhouette Width") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

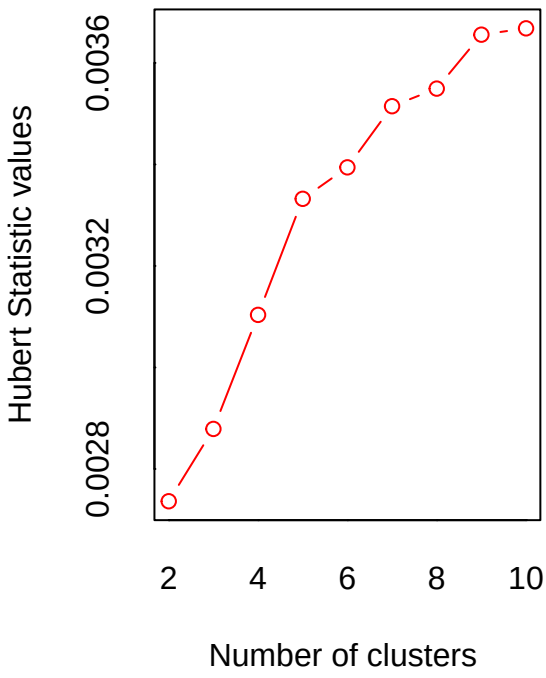


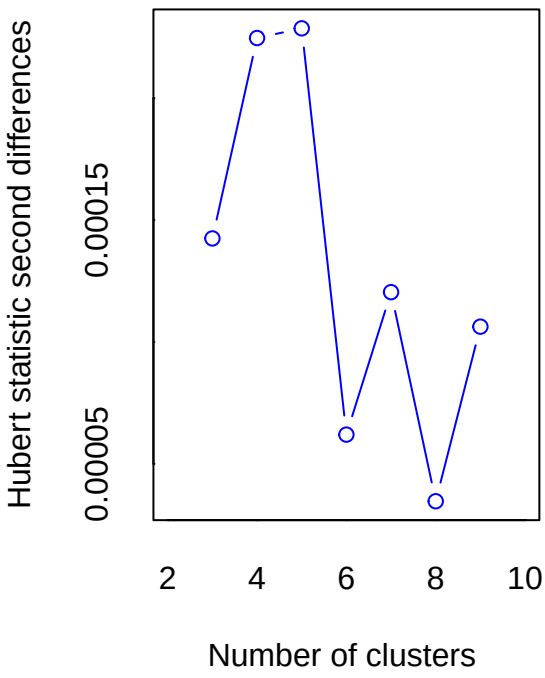

```
install.packages("NbClust")
```

```
## Installing package into 'C:/Users/CYTech Student/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

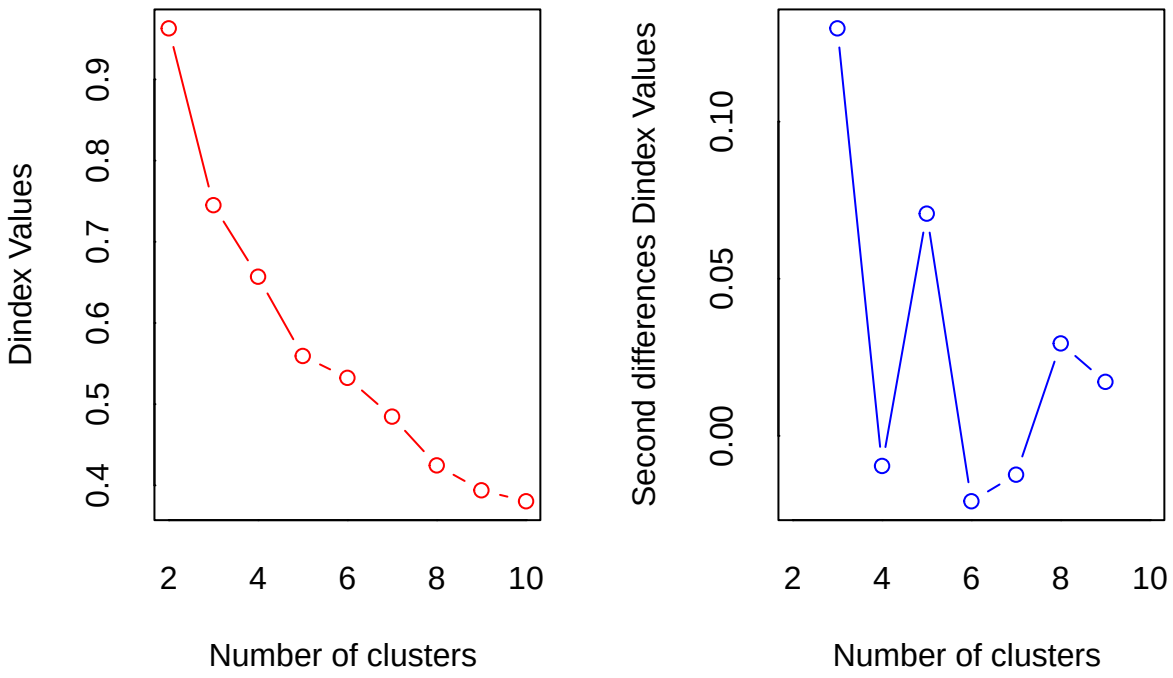
```
## package 'NbClust' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\CYTech Student\AppData\Local\Temp\RtmpwjYx9r\downloaded_packages
```

```
library(NbClust)

#finding optimal number of clusters using NbClust
nb_result <- NbClust(pca_scores, min.nc = 2, max.nc = 10, method = "kmeans")
```






```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```



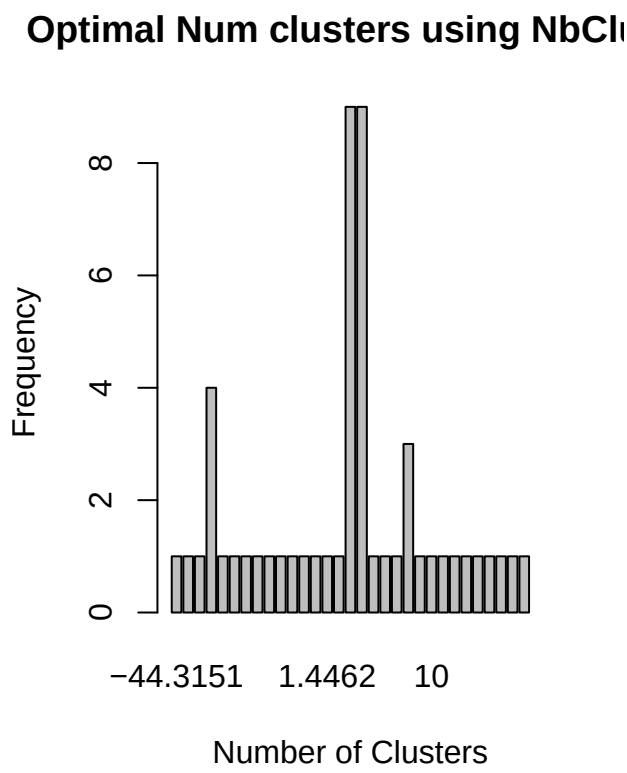

```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 9 proposed 3 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 3 proposed 9 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
print(nb_result$Best.nc)
```

```
##                     KL       CH Hartigan    CCC    Scott  Marriot   TrCovW
## Number_clusters  9.0000   9.0000   3.0000 3.0000   3.0000    5.000     3.00
```

```
## Value_Index     14.0393 295.9081  63.7653 5.8556 130.2769 8642.763 17538.55
##                   TraceW Friedman   Rubin Cindex     DB Silhouette   Duda
## Number_clusters  3.0000   8.0000  9.0000 3.0000 2.0000     2.0000 2.0000
## Value_Index     56.3319   9.5436 -1.6317 0.2113 0.6485     0.6145 1.7818
##                 PseudoT2   Beale Ratkowsky    Ball PtBiserial   Frey McClain
## Number_clusters   2.0000  2.0000    3.0000  3.0000     2.0000 3.0000  2.0000
## Value_Index     -44.3151 -0.4298    0.4615 59.9592     0.7896 1.4462  0.3202
##                   Dunn Hubert SDindex Dindex    SDbw
## Number_clusters 2.0000      0  2.0000      0 10.0000
## Value_Index     0.2504      0  1.3813      0  0.1838
```

```r
#Barplot to show the frequency of optimal clusters suggested by different indices
barplot(table(nb_result$Best.nc),
        xlab = "Number of Clusters",
        ylab = "Frequency",
        main = "Optimal Num clusters using NbClust")
```




```r
library(FactoMineR)
library(ggplot2)
```

Part 4 Ex1.)

```r
data("decathlon")

str(decathlon)
```

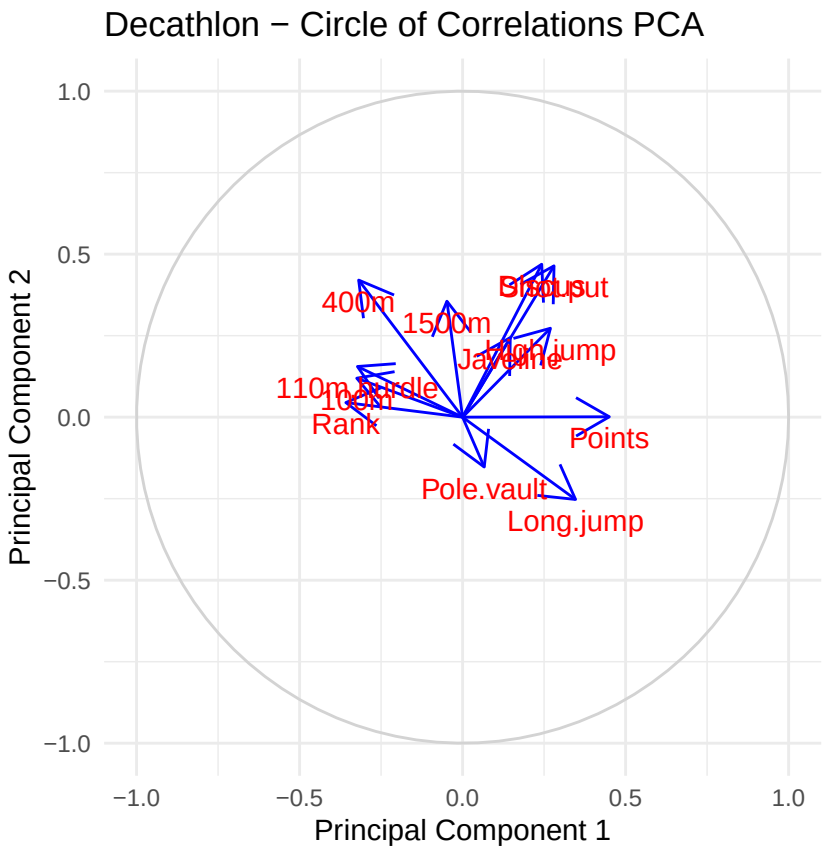
```
## 'data.frame':    41 obs. of  13 variables:
##  $ 100m       : num  11 10.8 11 11 11.3 ...
##  $ Long.jump  : num  7.58 7.4 7.3 7.23 7.09 7.6 7.3 7.31 6.81 7.56 ...
##  $ Shot.put   : num  14.8 14.3 14.8 14.2 15.2 ...
##  $ High.jump  : num  2.07 1.86 2.04 1.92 2.1 1.98 2.01 2.13 1.95 1.86 ...
##  $ 400m       : num  49.8 49.4 48.4 48.9 50.4 ...
##  $ 110m.hurdle: num  14.7 14.1 14.1 15 15.3 ...
##  $ Discus     : num  43.8 50.7 49 40.9 46.3 ...
##  $ Pole.vault : num  5.02 4.92 4.92 5.32 4.72 4.92 4.42 4.42 4.92 4.82 ...
##  $ Javeline   : num  63.2 60.1 50.3 62.8 63.4 ...
##  $ 1500m      : num  292 302 300 280 276 ...
##  $ Rank       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Points     : int  8217 8122 8099 8067 8036 8030 8004 7995 7802 7733 ...
##  $ Competition: Factor w/ 2 levels "Decastar","OlympicG": 1 1 1 1 1 1 1 1 1 1 ...
```

```
decathlon_numeric <- decathlon[, sapply(decathlon, is.numeric)]

#Standardize
decathlon_scaled <- scale(decathlon_numeric)

pca_result <- prcomp(decathlon_scaled, center = TRUE, scale. = TRUE)

summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.1815 1.3191 1.1895 1.06385 0.92841 0.77931 0.71446
## Proportion of Variance 0.3966 0.1450 0.1179 0.09431 0.07183 0.05061 0.04254
## Cumulative Proportion  0.3966 0.5416 0.6595 0.75380 0.82563 0.87624 0.91878
##                           PC8    PC9    PC10    PC11    PC12
## Standard deviation     0.64116 0.4850 0.43286 0.37545 0.00735
## Proportion of Variance 0.03426 0.0196 0.01561 0.01175 0.00000
## Cumulative Proportion  0.95303 0.9726 0.98825 1.00000 1.00000
```

```
pca_loadings <- pca_result$rotation

correlation_circle <- as.data.frame(pca_loadings)

radius <- 1

theta <- seq(0, 2 * pi, length.out = 100)
circle <- data.frame(x = radius * cos(theta), y = radius * sin(theta))

ggplot(circle, aes(x = x, y = y)) +
  geom_path(color = "lightgrey") +
  geom_segment(data = correlation_circle, aes(x = 0, y = 0,
                                              xend = PC1, yend = PC2),
               arrow = arrow(length = unit(0.2, "inches")),
               color = "blue") +
  geom_text(data = correlation_circle, aes(x = PC1, y = PC2, label = rownames(correlation_circle)),
            vjust = 1.5, color = "red") +
  labs(title = "Decathlon - Circle of Correlations PCA",
       x = "Principal Component 1",
```

```
       y = "Principal Component 2") +
  coord_fixed() +
  theme_minimal()
```




```
library(dplyr)
```

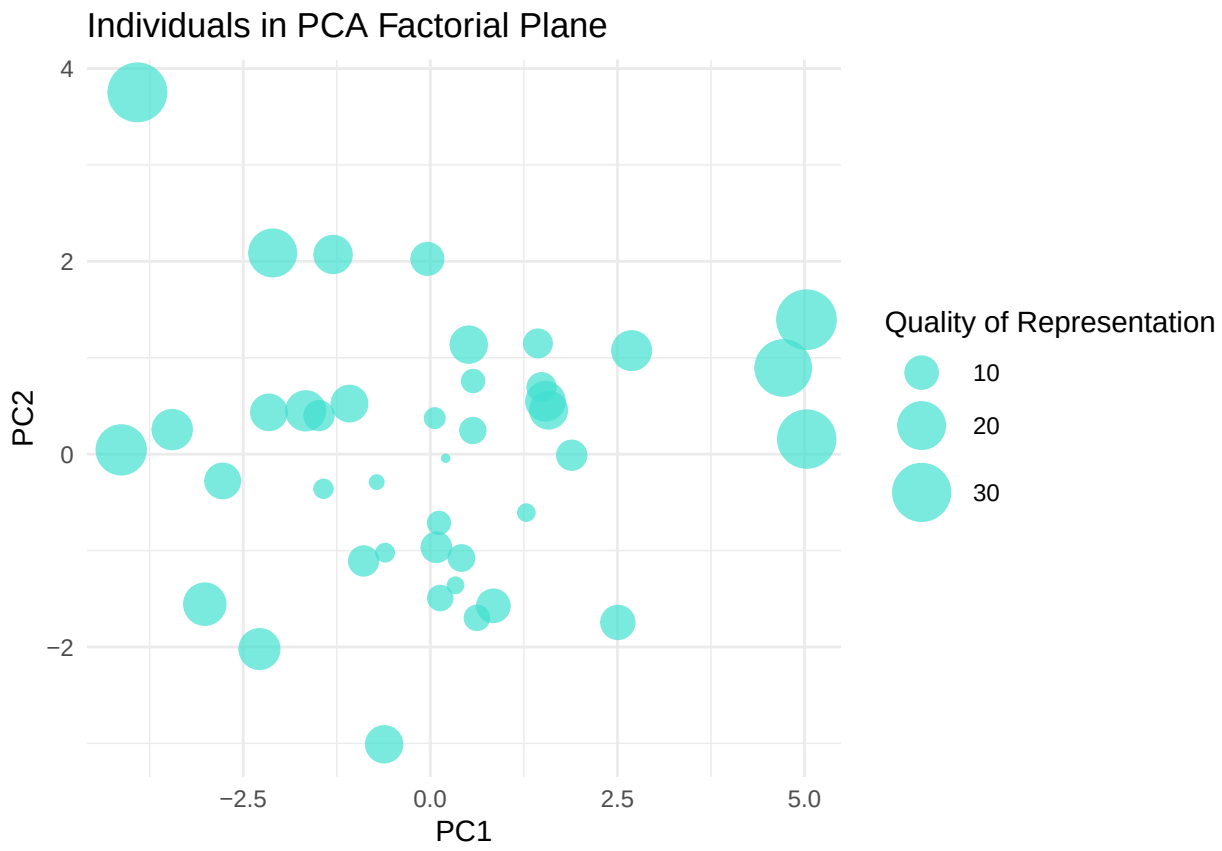
Ex2.)

```
data("decathlon")

str(decathlon)
```

```
## 'data.frame':    41 obs. of  13 variables:
##  $ 100m       : num  11 10.8 11 11 11.3 ...
##  $ Long.jump  : num  7.58 7.4 7.3 7.23 7.09 7.6 7.3 7.31 6.81 7.56 ...
##  $ Shot.put   : num  14.8 14.3 14.8 14.2 15.2 ...
##  $ High.jump  : num  2.07 1.86 2.04 1.92 2.1 1.98 2.01 2.13 1.95 1.86 ...
##  $ 400m       : num  49.8 49.4 48.4 48.9 50.4 ...
##  $ 110m.hurdle: num  14.7 14.1 14.1 15 15.3 ...
##  $ Discus     : num  43.8 50.7 49 40.9 46.3 ...
##  $ Pole.vault : num  5.02 4.92 4.92 5.32 4.72 4.92 4.42 4.42 4.92 4.82 ...
##  $ Javeline   : num  63.2 60.1 50.3 62.8 63.4 ...
##  $ 1500m      : num  292 302 300 280 276 ...
##  $ Rank       : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
##  $ Points     : int  8217 8122 8099 8067 8036 8030 8004 7995 7802 7733 ...
##  $ Competition: Factor w/ 2 levels "Decastar","OlympicG": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
#Remove any non numeric columns if present
decathlon_numeric <- decathlon[, sapply(decathlon, is.numeric)]

decathlon_scaled <- scale(decathlon_numeric)

pca_result <- prcomp(decathlon_scaled, center = TRUE, scale. = TRUE)

pca_scores <- as.data.frame(pca_result$x)

#Calculate quality of representation (squared cosines)
quality_representation <- rowSums(pca_scores^2)

plot_data <- pca_scores %>%
  mutate(quality = quality_representation)

#Bubble plot
ggplot(plot_data, aes(x = PC1, y = PC2)) +
  geom_point(aes(size = quality), alpha = 0.7, color = "turquoise") +
  scale_size(range = c(1, 10), name = "Quality of Representation") +  # Adjust size range as needed
  labs(title = "Individuals in PCA Factorial Plane",
       x = "PC1",
       y = "PC2") +
  theme_minimal()
```



```r
decathlon_numeric <- decathlon[, sapply(decathlon, is.numeric)]
decathlon_scaled <- scale(decathlon_numeric)

pca_result <- prcomp(decathlon_scaled, center = TRUE, scale. = TRUE)

pca_summary <- summary(pca_result)

#Extract variance explained for the first five components
variance_explained <- round(pca_summary$importance[2, 1:5] * 100, 3)  #getting percentage so times 100

#Extract loadings for the first five components and round them
loadings_table <- round(as.data.frame(pca_result$rotation[, 1:5]), 3)

variance_table <- data.frame(
  Component = paste0("PC", 1:5),
  Variance_Explained = variance_explained
)

print("Variance Explained by each PC:")
```

```
## [1] "Variance Explained by each PC:"
```

```r
print(variance_table)
```

```
##     Component Variance_Explained
## PC1       PC1              39.657
## PC2       PC2              14.501
## PC3       PC3              11.791
## PC4       PC4               9.431
## PC5       PC5               7.183
```

```r
print("Loadings of variables on the first five PCs:")
```

```
## [1] "Loadings of variables on the first five PCs:"
```

```r
print(loadings_table)
```

```
##                 PC1    PC2    PC3    PC4    PC5
## 100m         -0.325  0.119 -0.130  0.194 -0.554
## Long.jump     0.347 -0.252  0.153  0.007  0.051
## Shot.put      0.280  0.464 -0.017  0.103  0.079
## High.jump     0.269  0.273 -0.201 -0.081 -0.434
## 400m         -0.319  0.420  0.111  0.083  0.080
## 110m.hurdle -0.322  0.155 -0.068  0.384 -0.120
## Discus        0.243  0.469  0.046 -0.257 -0.105
## Pole.vault    0.066 -0.153  0.592  0.554 -0.110
## Javeline      0.146  0.244 -0.328  0.563  0.465
## 1500m        -0.048  0.356  0.657 -0.179  0.124
## Rank         -0.358  0.045 -0.083 -0.233  0.468
## Points        0.450  0.001 -0.053  0.123  0.050
```

```r
loadings <- as.data.frame(round(pca_result$rotation[, 1:5], 3))

loadings$Variable <- rownames(loadings)
loadings <- loadings[, c("Variable", "PC1", "PC2", "PC3", "PC4", "PC5")]

print("Loadings for the first five components:")
```

```
## [1] "Loadings for the first five components:"
```

```r
print(loadings)
```

```
##                Variable    PC1    PC2    PC3    PC4    PC5
## 100m               100m -0.325  0.119 -0.130  0.194 -0.554
## Long.jump     Long.jump  0.347 -0.252  0.153  0.007  0.051
## Shot.put       Shot.put  0.280  0.464 -0.017  0.103  0.079
## High.jump     High.jump  0.269  0.273 -0.201 -0.081 -0.434
## 400m               400m -0.319  0.420  0.111  0.083  0.080
## 110m.hurdle 110m.hurdle -0.322  0.155 -0.068  0.384 -0.120
## Discus           Discus  0.243  0.469  0.046 -0.257 -0.105
## Pole.vault   Pole.vault  0.066 -0.153  0.592  0.554 -0.110
## Javeline       Javeline  0.146  0.244 -0.328  0.563  0.465
## 1500m             1500m -0.048  0.356  0.657 -0.179  0.124
## Rank               Rank -0.358  0.045 -0.083 -0.233  0.468
## Points           Points  0.450  0.001 -0.053  0.123  0.050
```

```r
library(ggplot2)

data("decathlon")

str(decathlon)
```

```
## 'data.frame':    41 obs. of  13 variables:
##  $ 100m       : num  11 10.8 11 11 11.3 ...
##  $ Long.jump  : num  7.58 7.4 7.3 7.23 7.09 7.6 7.3 7.31 6.81 7.56 ...
##  $ Shot.put   : num  14.8 14.3 14.8 14.2 15.2 ...
##  $ High.jump  : num  2.07 1.86 2.04 1.92 2.1 1.98 2.01 2.13 1.95 1.86 ...
##  $ 400m       : num  49.8 49.4 48.4 48.9 50.4 ...
##  $ 110m.hurdle: num  14.7 14.1 14.1 15 15.3 ...
##  $ Discus     : num  43.8 50.7 49 40.9 46.3 ...
##  $ Pole.vault : num  5.02 4.92 4.92 5.32 4.72 4.92 4.42 4.42 4.92 4.82 ...
##  $ Javeline   : num  63.2 60.1 50.3 62.8 63.4 ...
##  $ 1500m      : num  292 302 300 280 276 ...
##  $ Rank       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Points     : int  8217 8122 8099 8067 8036 8030 8004 7995 7802 7733 ...
##  $ Competition: Factor w/ 2 levels "Decastar","OlympicG": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
decathlon_numeric <- decathlon[, sapply(decathlon, is.numeric)]

decathlon_scaled <- scale(decathlon_numeric)

pca_result <- prcomp(decathlon_scaled, center = TRUE, scale. = TRUE)
```

```r
pca_summary <- summary(pca_result)

pca_loadings <- as.data.frame(pca_result$rotation)

variance_explained <- pca_summary$importance[2, 1:5] * 100

component_names <- c("Athleticism", "Upper Strength", "Jumping", "Endurance", "Technique")
justifications <- c(
  "There are High loadings across the running events for instance in 100m, 400m or 1500m, meaning that

  "There are also high loadings on throwing events like Shot Put and Javelin, which show strength.",

  "Next, there is high loadings on Long Jump and High Jump which are jumping events.",

  "There is high loading specifically on 1500m which demonstrate a athelete stamina.",
  "Lastly, there are loadings on very mechanical and technique based events like Pole Vault, where the a
)

result_table <- data.frame(
  Component = paste0("PC", 1:5),
  Name = component_names,
  Variance_Explained = round(variance_explained, 2),
  Justification = justifications
)

print(result_table)
```

```
##     Component           Name Variance_Explained
## PC1       PC1    Athleticism              39.66
## PC2       PC2 Upper Strength              14.50
## PC3       PC3        Jumping              11.79
## PC4       PC4      Endurance               9.43
## PC5       PC5      Technique               7.18
##
## PC1  There are High loadings across the running events for instance in 100m, 400m or 1500m, meaning
## PC2                                                  There are also high loadings on throwing events li
## PC3                                                                Next, there is high loadings on Long
## PC4                                                                There is high loading specifically
## PC5 Lastly, there are loadings on very mechanical and technique based events like Pole Vault, where
```