DS 116 -Data Visualization

PCA and Clustering

Habet Madoyan

American University of Armenia

Section 1

Dimensionality Reduction / Principal Component Analysis

Principal Component Analysis

Why to use dimension reduction?

Find latent hidden variables that are not/cannot be directly measured - Reveal the hidden structure of the data - Transform the feature space into variables that are not correlated, thus new variables can be used in different machine learning techniques

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(p)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(p)} \\ \cdots & \cdots & \cdots & \cdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(p)} \end{pmatrix} \rightarrow \mathbf{Z}= \begin{pmatrix} z_1^{(1)} & z_1^{(2)} & \cdots & z_1^{(p')} \\ z_2^{(1)} & z_2^{(2)} & \cdots & z_2^{(p')} \\ \cdots & \cdots & \cdots & \cdots \\ z_n^{(1)} & z_n^{(2)} & \cdots & z_n^{(p')} \end{pmatrix}$$

$$p' \ll p$$

Principal Components Analysis (PCA)

- PCA is a technique that can be used to simplify a dataset
- It is a linear transformation that chooses a new coordinate system for the data set such that greatest variance by any projection of the data set comes to lie on the first axis (called the first principal component), the second greatest variance on the second axis, and so on.
- PCA can be used for reducing dimensionality by eliminating the later principal components.

Principal Components Analysis (PCA)

Each new Component/factor is a linear combination of all variables.

Imagine we have p variables $(X_j, j = 1, 2, ..., p)$

$$Z_i = \sum_{i=1}^{p} w_{ij} X_j$$

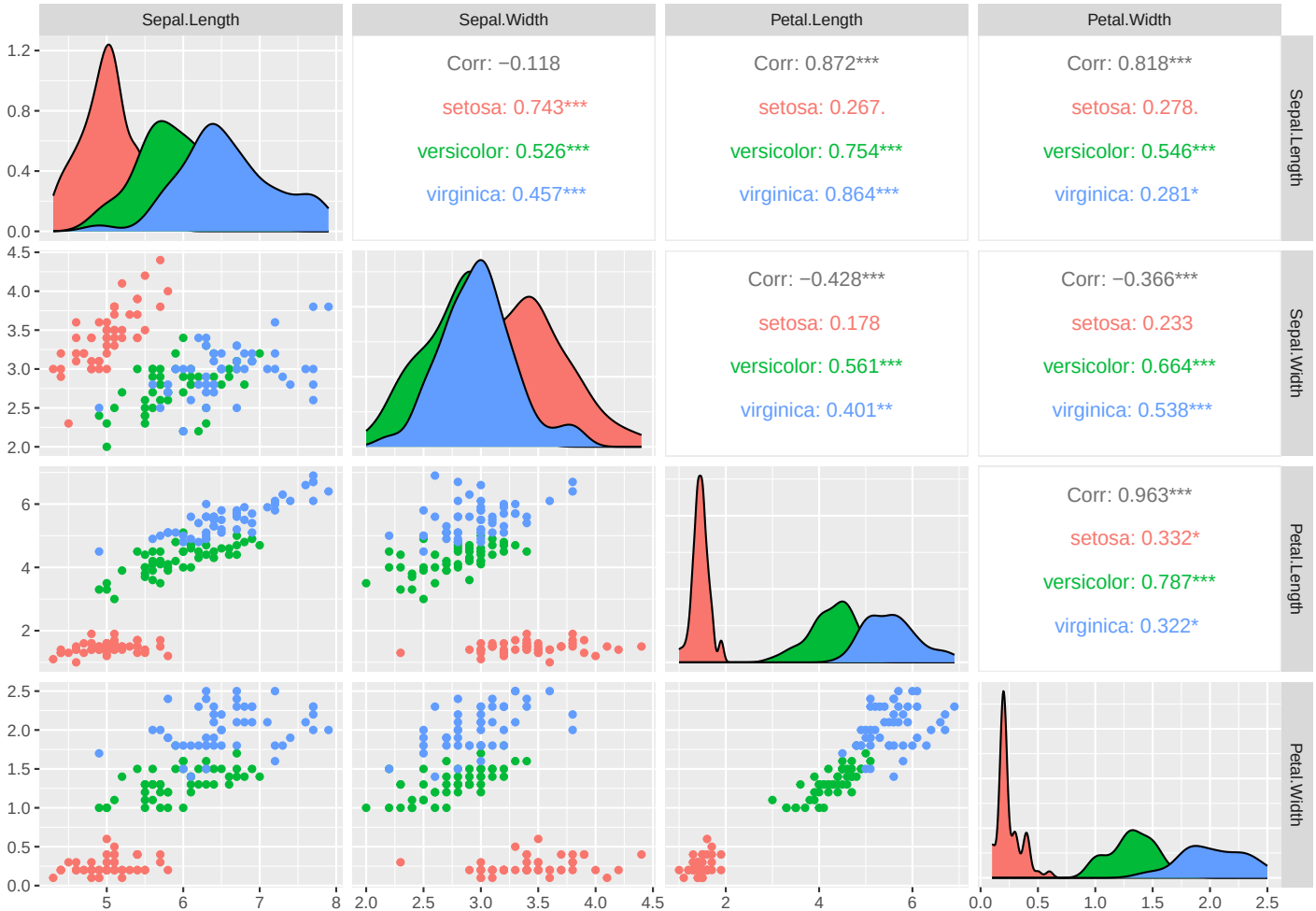
$$Z_i = w_{i1} X_1 + w_{i2} X_2 + ... + w_{ip} X_p$$

where each pair of Z's are orthogonal (correlation $= 0$)!

After estimating weights w, Z's are ordered by their variance, with Z_1 having the largest variance and Z_p having the smallest variance.

Principal Component Analysis

```
ggpairs(iris,columns = 1:4, mapping = aes(color = Species))
```



Principal Component Analysis

```
iris_pca = prcomp(x = iris[,1:4], center = T, scale. = T)
iris_pca
```

```
## Standard deviations (1, .., p=4):
## [1] 1.7083611 0.9560494 0.3830886 0.1439265
##
## Rotation (n x k) = (4 x 4):
##                     PC1         PC2         PC3         PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

Principal Component Analysis

Types of visualizations

- **Scree Plot**
  - Shows the variance explained by each principal component.
  - Helps determine the number of components to retain by visualizing where the "elbow" occurs.
- **Score Plot**
  - Plots the data in the new component space, often the first two principal components.
  - Reveals clusters or patterns in data points, such as grouping countries based on similarities.

Principal Component Analysis

Types of visualizations

- **Loading Plot**
  - Visualizes how strongly each original variable contributes to each principal component.
  - Helps interpret the meaning of each component based on variable correlations.
- **Biplot**
  - Combines the PCA score plot with loadings to show both data points (e.g., countries) and feature correlations.
  - Useful for interpreting how original variables contribute to principal components.

Principal Component Analysis - Scree plot

Variance explained

```
iris_pca$sdev^2
```

```
## [1] 2.91849782 0.91403047 0.14675688 0.02071484
```

Proportion of Variance explained

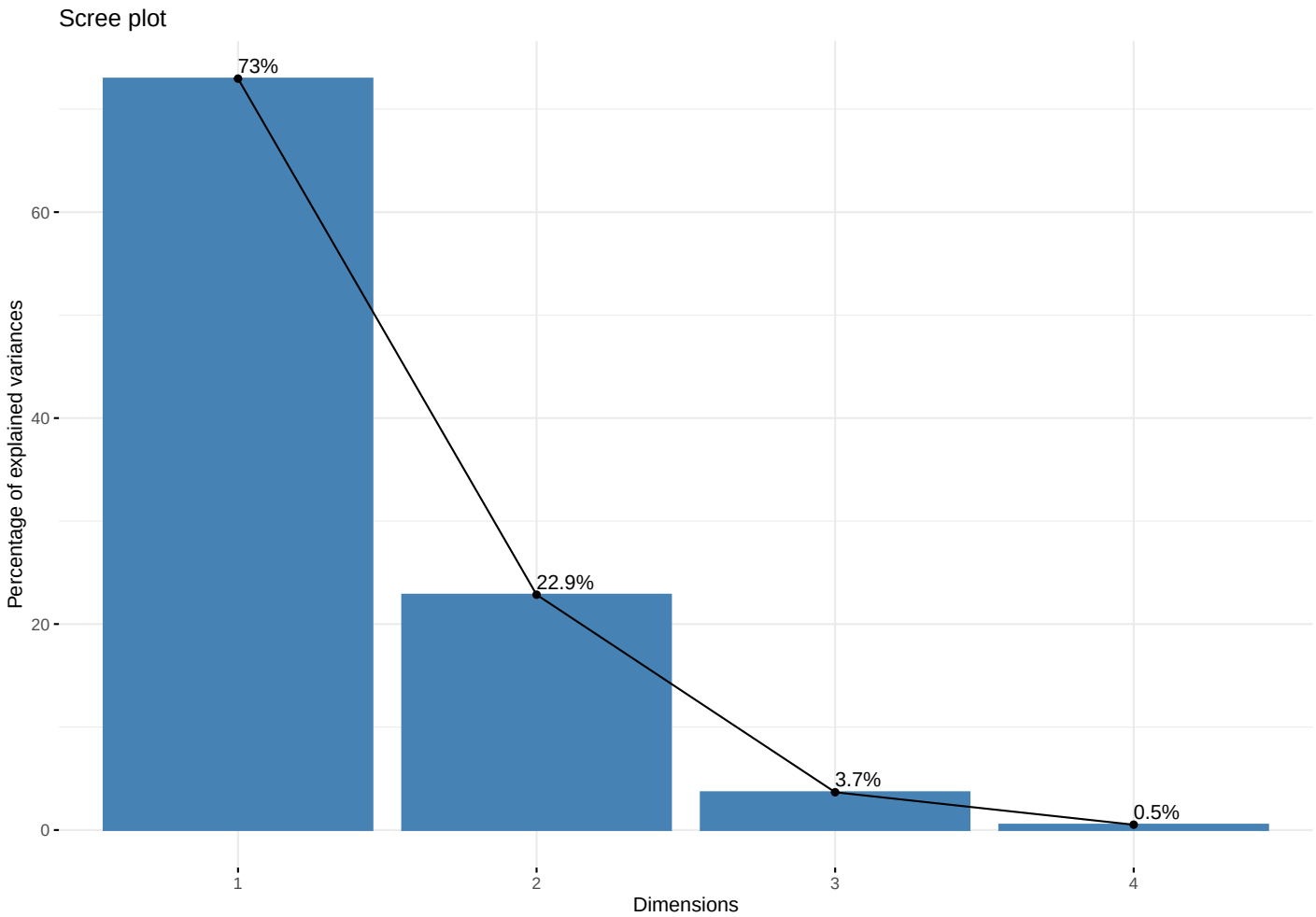
```
var_explained = iris_pca$sdev^2/sum(iris_pca$sdev^2)
var_explained
```

```
## [1] 0.729624454 0.228507618 0.036689219 0.005178709
```

Principal Component Analysis - Scree plot

Scree plot

```
fviz_screeplot(iris_pca, addlabels = T)
```



Principal Component Analysis - Score plot

```
head(iris_pca$x)
```

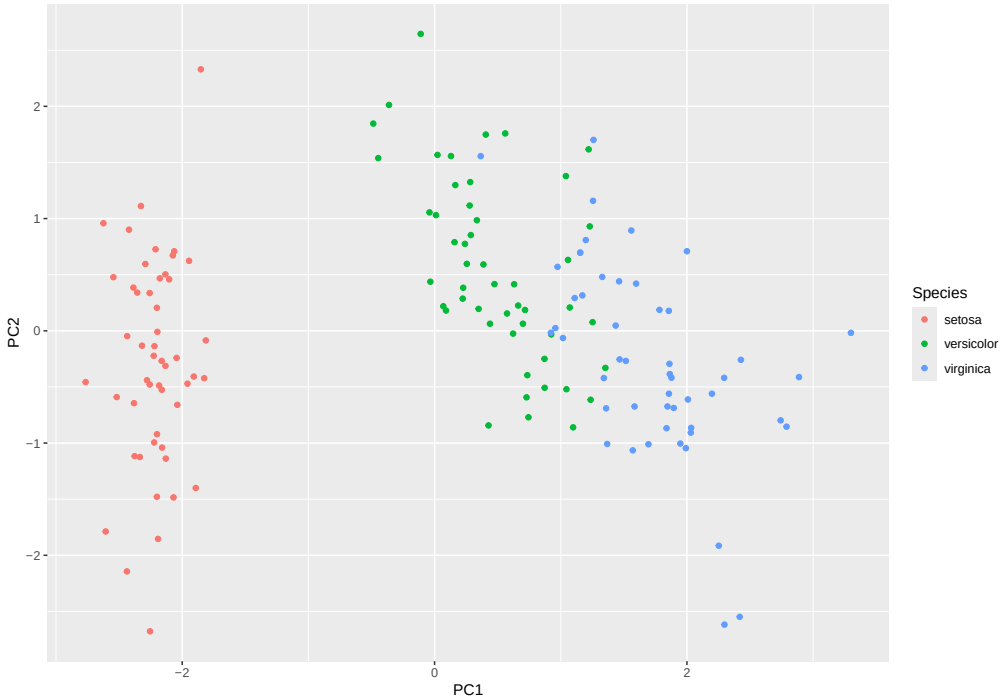
```
##              PC1         PC2          PC3          PC4
## [1,] -2.257141 -0.4784238  0.12727962  0.024087508
## [2,] -2.074013  0.6718827  0.23382552  0.102662845
## [3,] -2.356335  0.3407664 -0.04405390  0.028282305
## [4,] -2.291707  0.5953999 -0.09098530 -0.065735340
## [5,] -2.381863 -0.6446757 -0.01568565 -0.035802870
## [6,] -2.068701 -1.4842053 -0.02687825  0.006586116
```

Principal Component Analysis - Score plot

Combine together into one dataframe and visualize

```
pca_data_frame = data.frame(iris, iris_pca$x[,1:2])
```

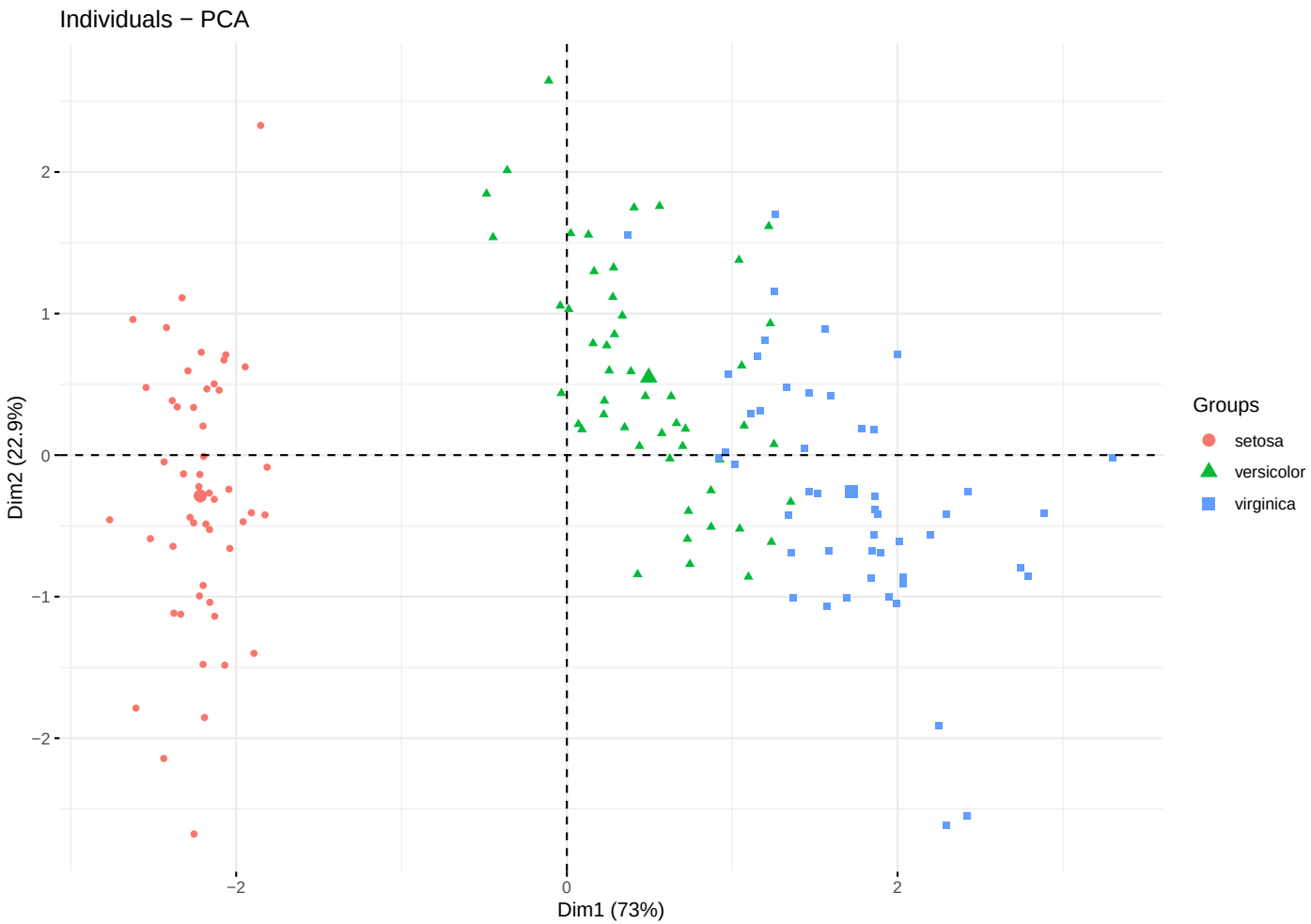
```
ggplot(data = pca_data_frame, aes(x = PC1, y = PC2,
    color = Species)) + geom_point()
```



Principal Component Analysis - Score plot

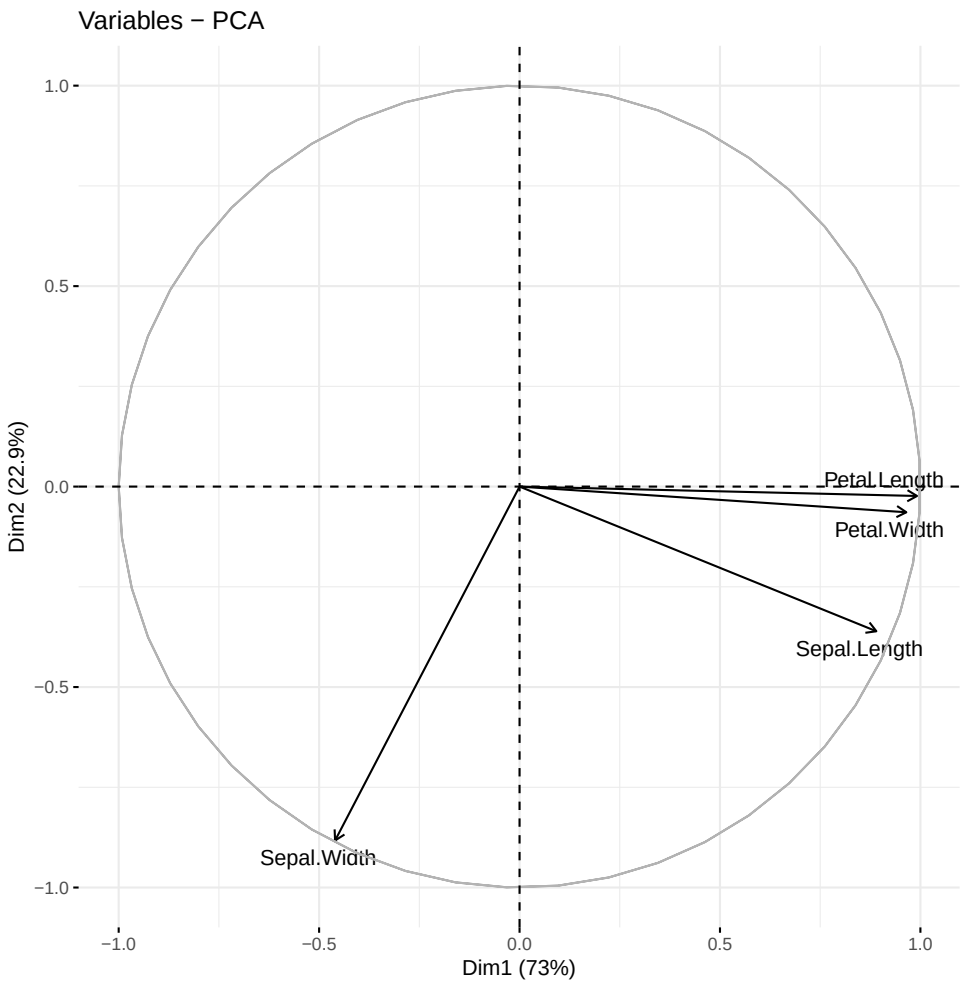
Using library factoextra

```
fviz_pca_ind(iris_pca, geom = "point", habillage = iris$Species)
```



Principal Component Analysis - Loading plot (Variables plot)

```
fviz_pca_var(iris_pca, repel = T)
```



Principal Component Analysis - Loading plot (Variables plot)

Loadings also help to interpret the graph.

PC1

- **Petal.Length** has the highest positive loading on PC1, meaning it contributes the most to this component.
- **Petal.Width** also has a positive loading, though smaller than Petal.Length, so it also influences PC1 but to a lesser extent.

```
iris_pca$rotation
```

```
##                    PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

Principal Component Analysis - Loading plot (Variables plot)

- **Sepal.Width** has a high negative loading, indicating it is strongly associated with PC2 in the negative direction.
- **Sepal.Length** also has a strong negative loading.

```
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

Principal Component Analysis - Loading plot (Variables plot)

Lets look at the correlations

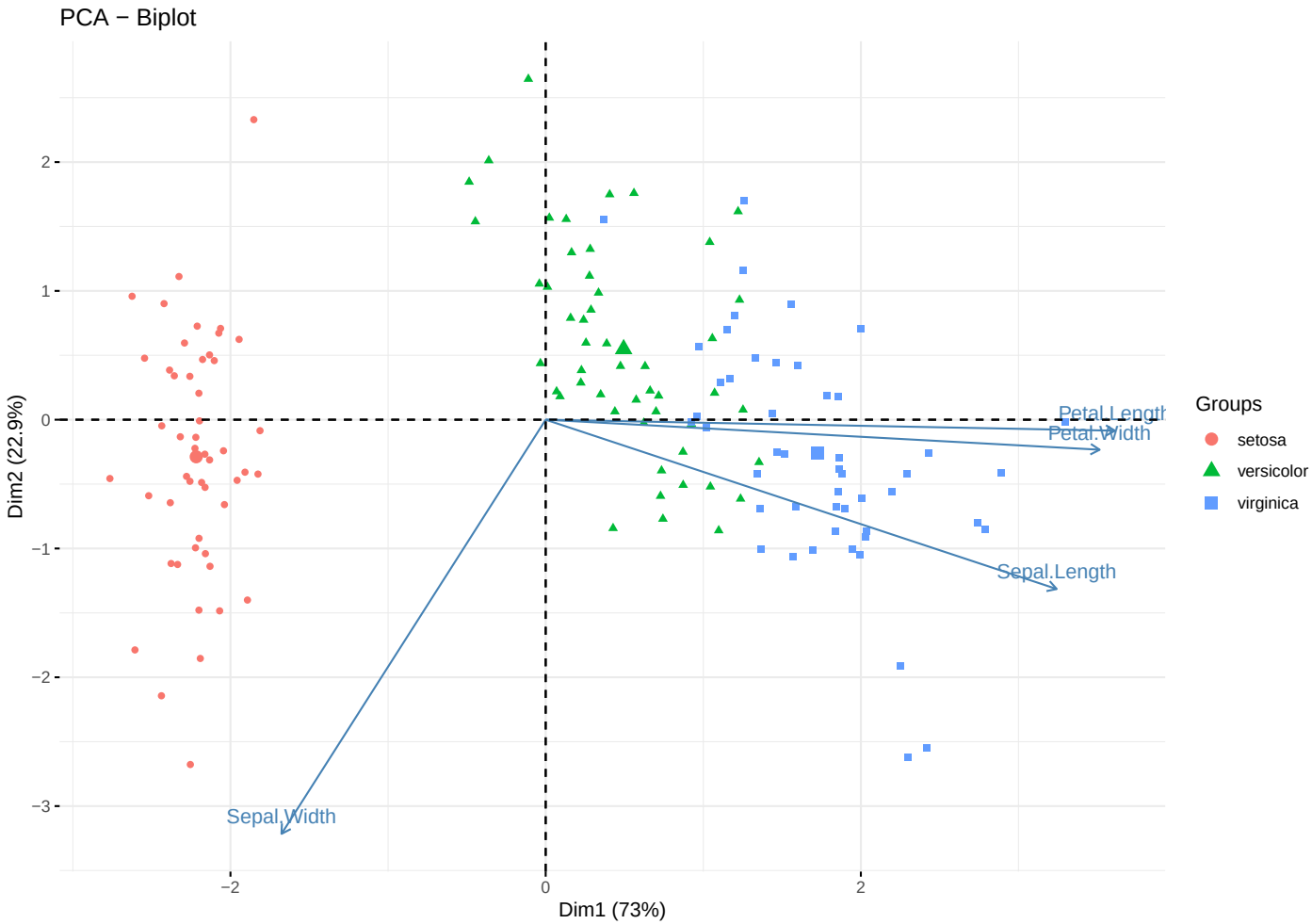
```
cor(pca_data_frame[,-5]) %>% round(4)
```

```
##                Sepal.Length Sepal.Width Petal.Length Petal.Width     PC1     PC2
## Sepal.Length         1.0000     -0.1176       0.8718      0.8179  0.8902 -0.3608
## Sepal.Width         -0.1176      1.0000      -0.4284     -0.3661 -0.4601 -0.8827
## Petal.Length         0.8718     -0.4284       1.0000      0.9629  0.9916 -0.0234
## Petal.Width          0.8179     -0.3661       0.9629      1.0000  0.9650 -0.0640
## PC1                  0.8902     -0.4601       0.9916      0.9650  1.0000  0.0000
## PC2                 -0.3608     -0.8827      -0.0234     -0.0640  0.0000  1.0000
```

Principal Component Analysis - Biplot

Biplot - variables and cases together

```
fviz_pca_biplot(iris_pca, label = "var", habillage = iris$Species)
```



Principal Component Analysis

Decathlon data

```
data("decathlon")
decathlon = decathlon %>% filter(Competition == "OlympicG")
head(decathlon, 3)
```

```
##          100m Long.jump Shot.put High.jump  400m 110m.hurdle Discus Pole.vault
## Sebrle 10.85      7.84    16.36      2.12 48.36       14.05  48.72        5.0
## Clay   10.44      7.96    15.23      2.06 49.19       14.13  50.11        4.9
## Karpov 10.50      7.81    15.93      2.09 46.81       13.97  51.65        4.6
##        Javeline  1500m Rank Points Competition
## Sebrle    70.52 280.01    1   8893    OlympicG
## Clay      69.71 282.00    2   8820    OlympicG
## Karpov    55.54 278.11    3   8725    OlympicG
```

Principal Component Analysis

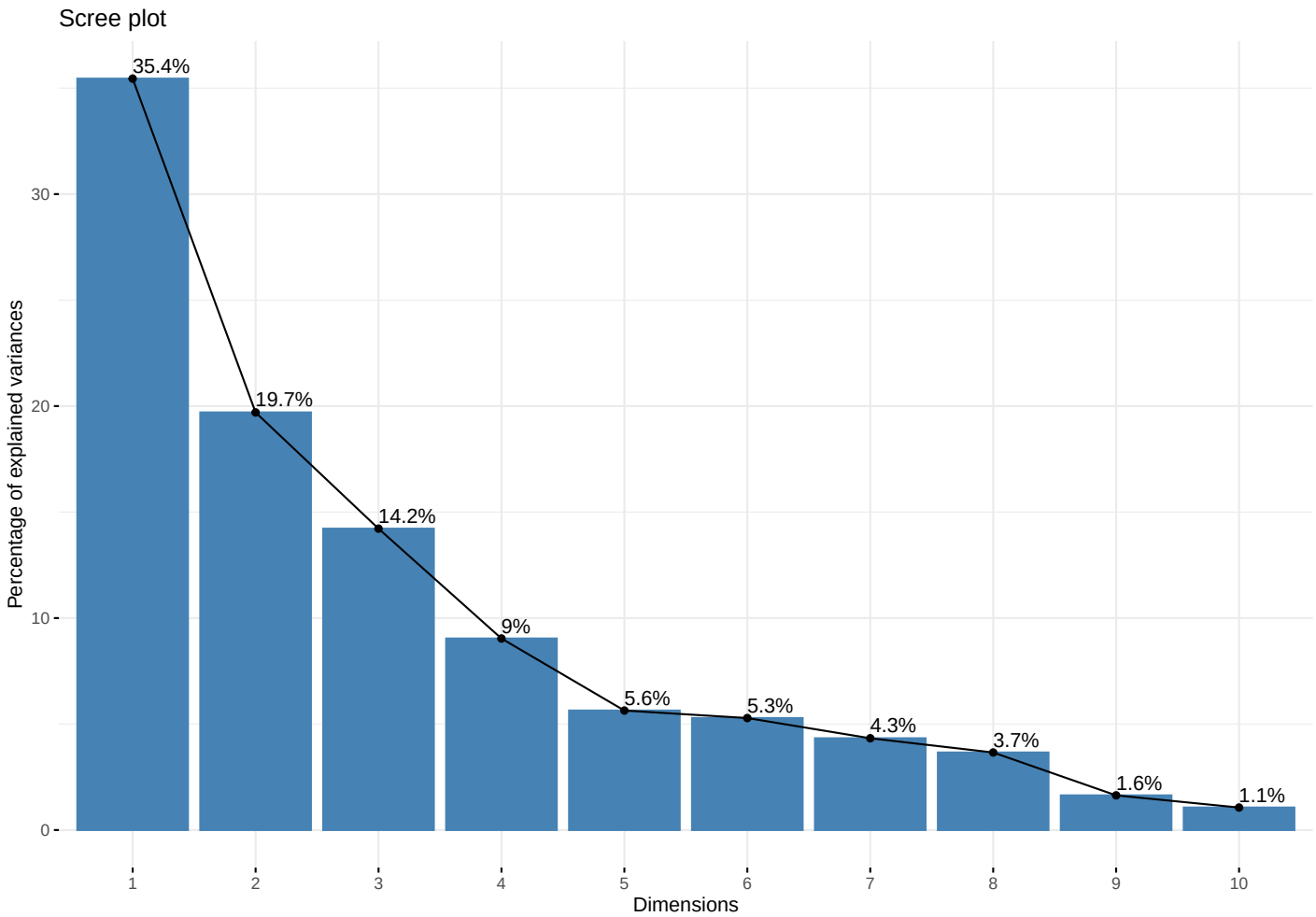
```
decathlon_pca = prcomp(decathlon[,1:10], center = T, scale. = T)
summary(decathlon_pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.8827 1.4036 1.1924 0.95052 0.75075 0.72679 0.65792
## Proportion of Variance 0.3545 0.1970 0.1422 0.09035 0.05636 0.05282 0.04329
## Cumulative Proportion  0.3545 0.5515 0.6936 0.78398 0.84035 0.89317 0.93645
##                           PC8     PC9    PC10
## Standard deviation     0.60482 0.40435 0.32580
## Proportion of Variance 0.03658 0.01635 0.01061
## Cumulative Proportion  0.97304 0.98939 1.00000
```

Principal Component Analysis

Scree plot

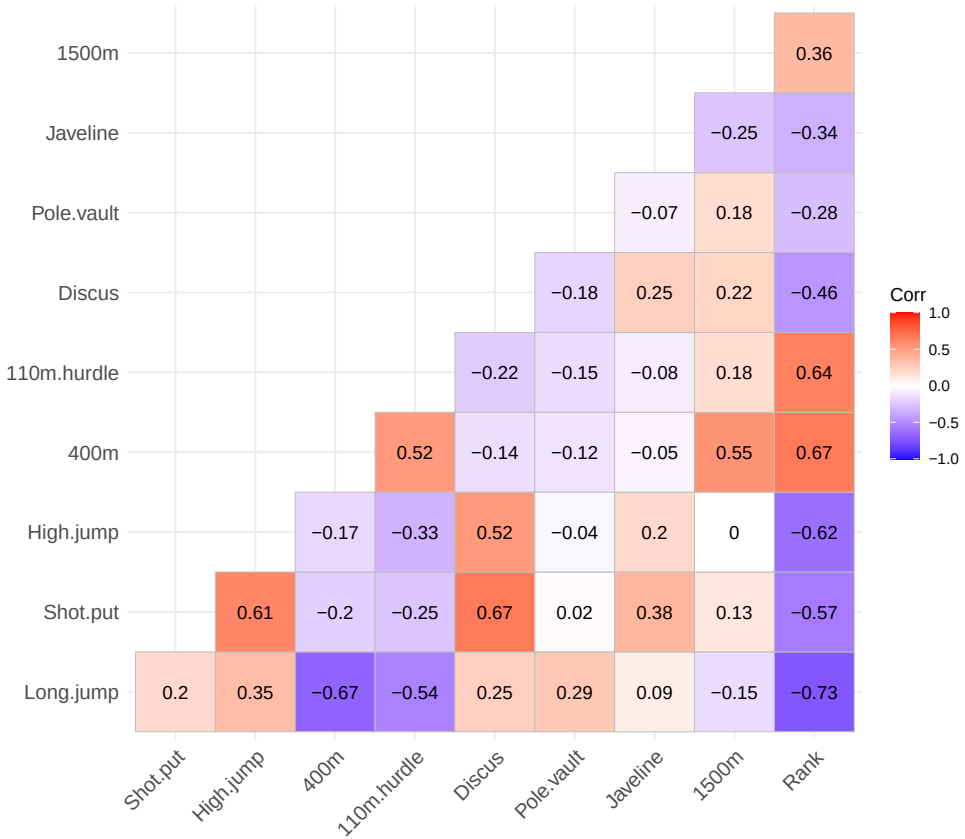
```
fviz_screeplot(decathlon_pca, addlabels = TRUE)
```



Principal Component Analysis

First lets look at the correlation heatmap

```
decathlon_cor_mat = cor(decathlon[,2:11])
ggcorrplot(decathlon_cor_mat , type = "lower", lab = T)
```



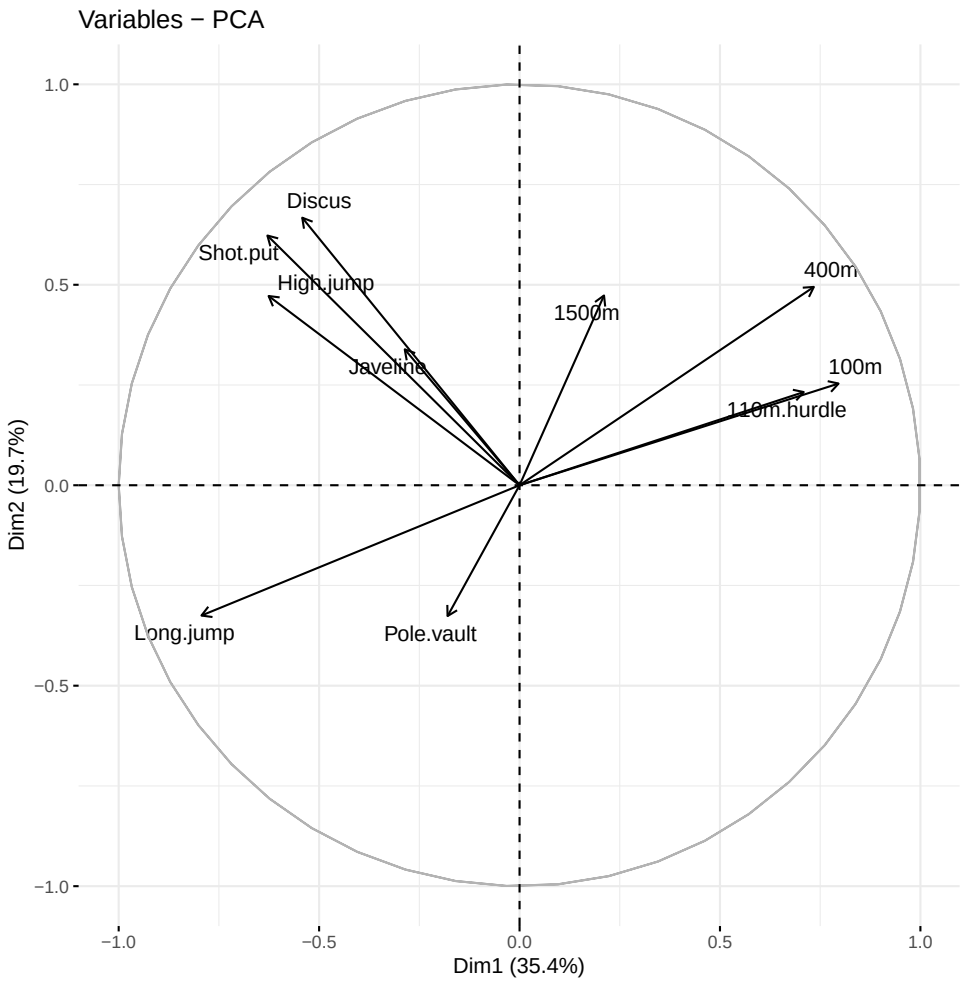
Principal Component Analysis

Groups of variables:

- **Sprints** 400m, 100m, 110m hurdle
- **Throws** Discus, shot put, javeline
- **Jumps** long jump, pole vault

Principal Component Analysis (Var Plot)

```
fviz_pca_var(decathlon_pca, repel = T)
```



Principal Component Analysis

400m, 100m, 110m hurdle

400m

100m

110m hurdle

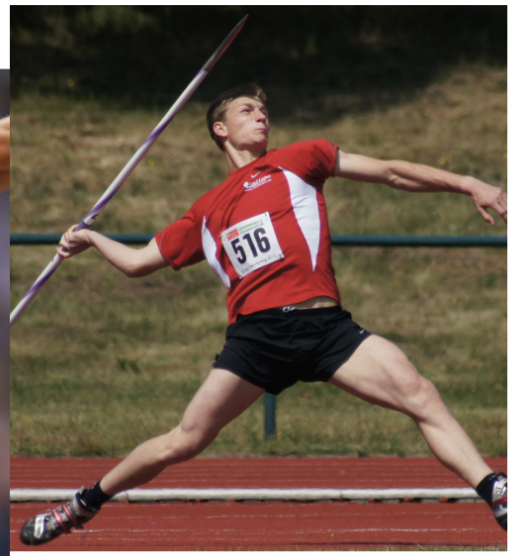
Principal Component Analysis

Discus, shot put, javeline

Discus

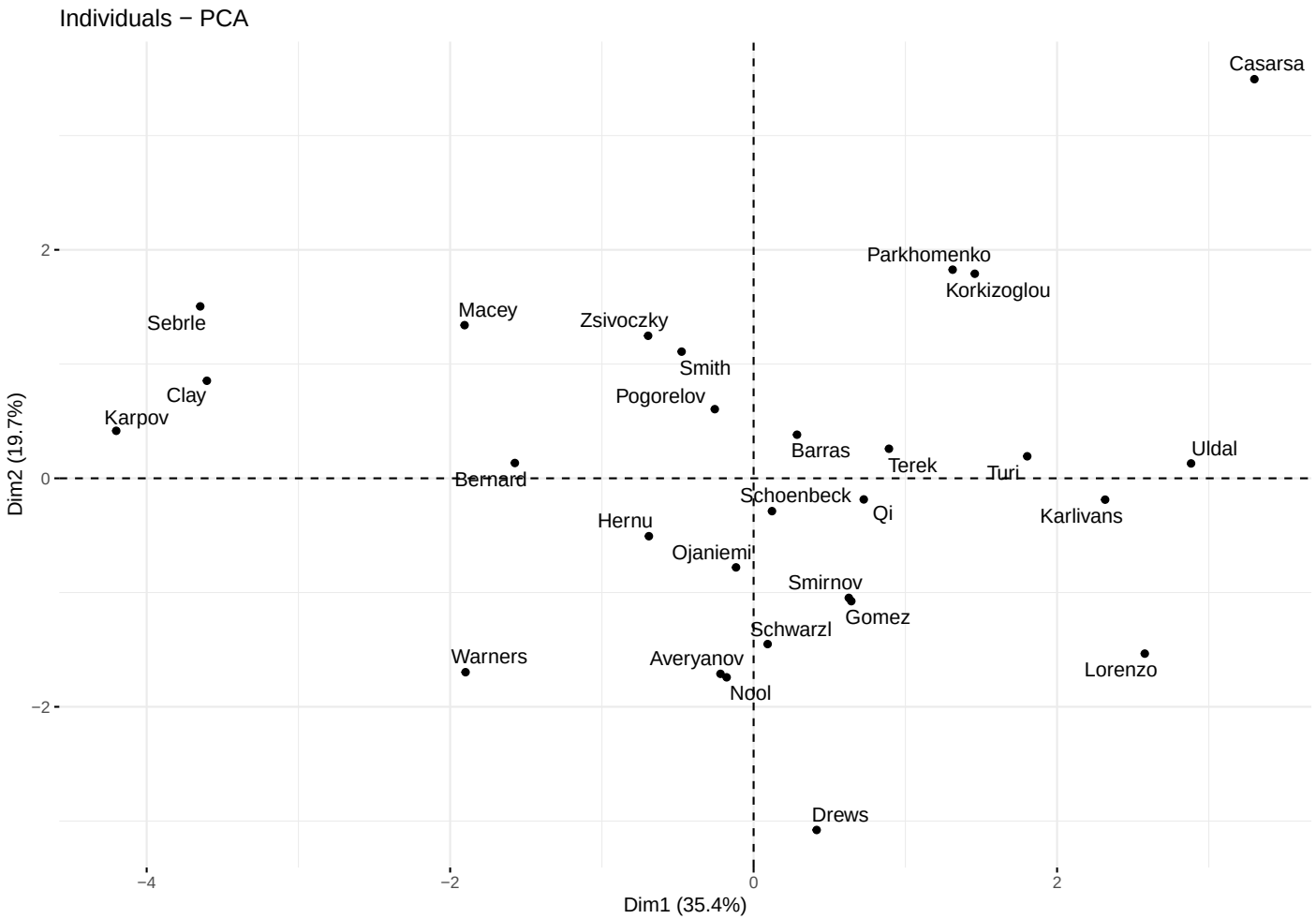
Shot put

Javeline



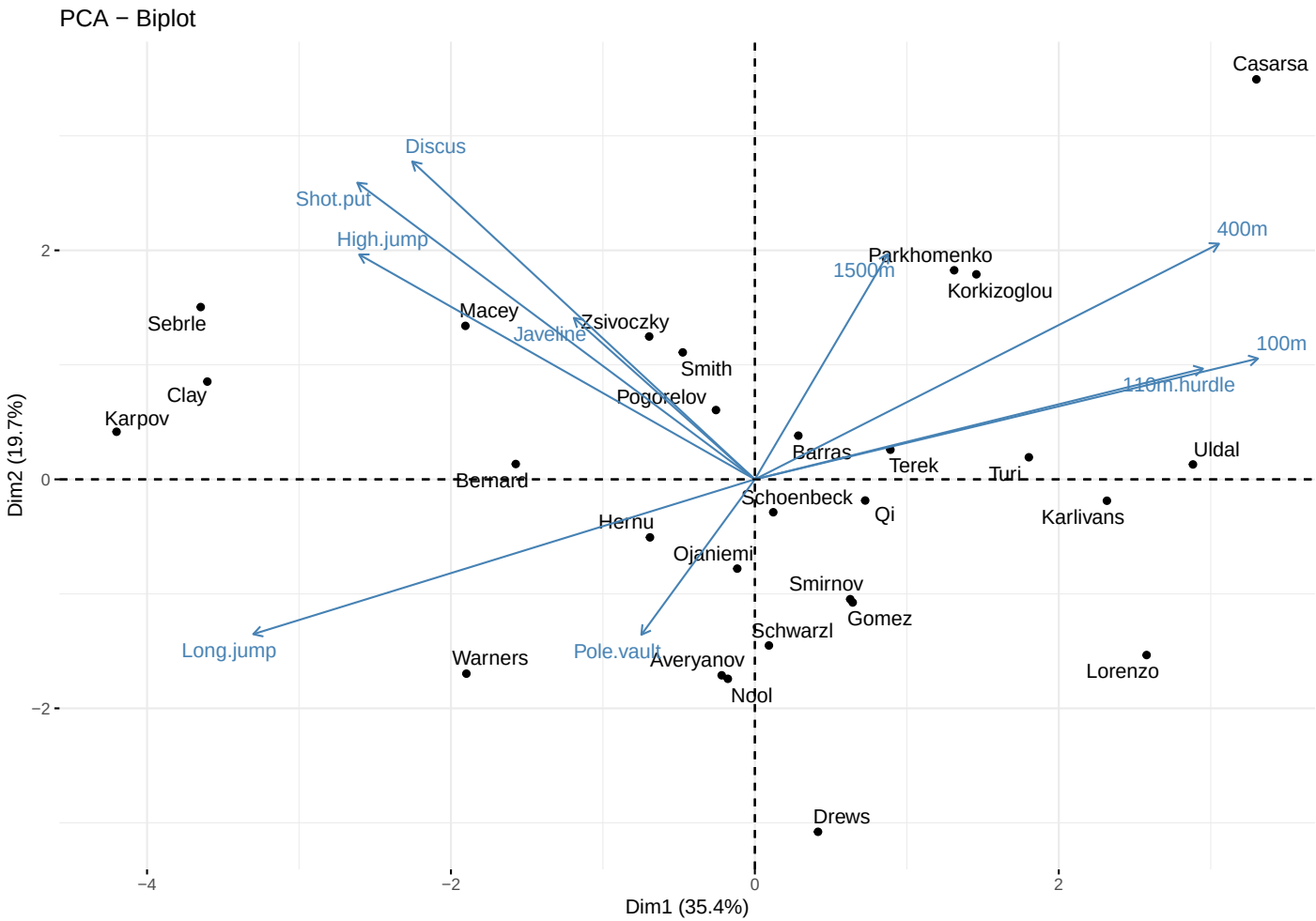
Principal Component Analysis

```
fviz_pca_ind(decathlon_pca, repel = T)
```



Principal Component Analysis

```
fviz_pca_biplot(decathlon_pca, repel = T)
```

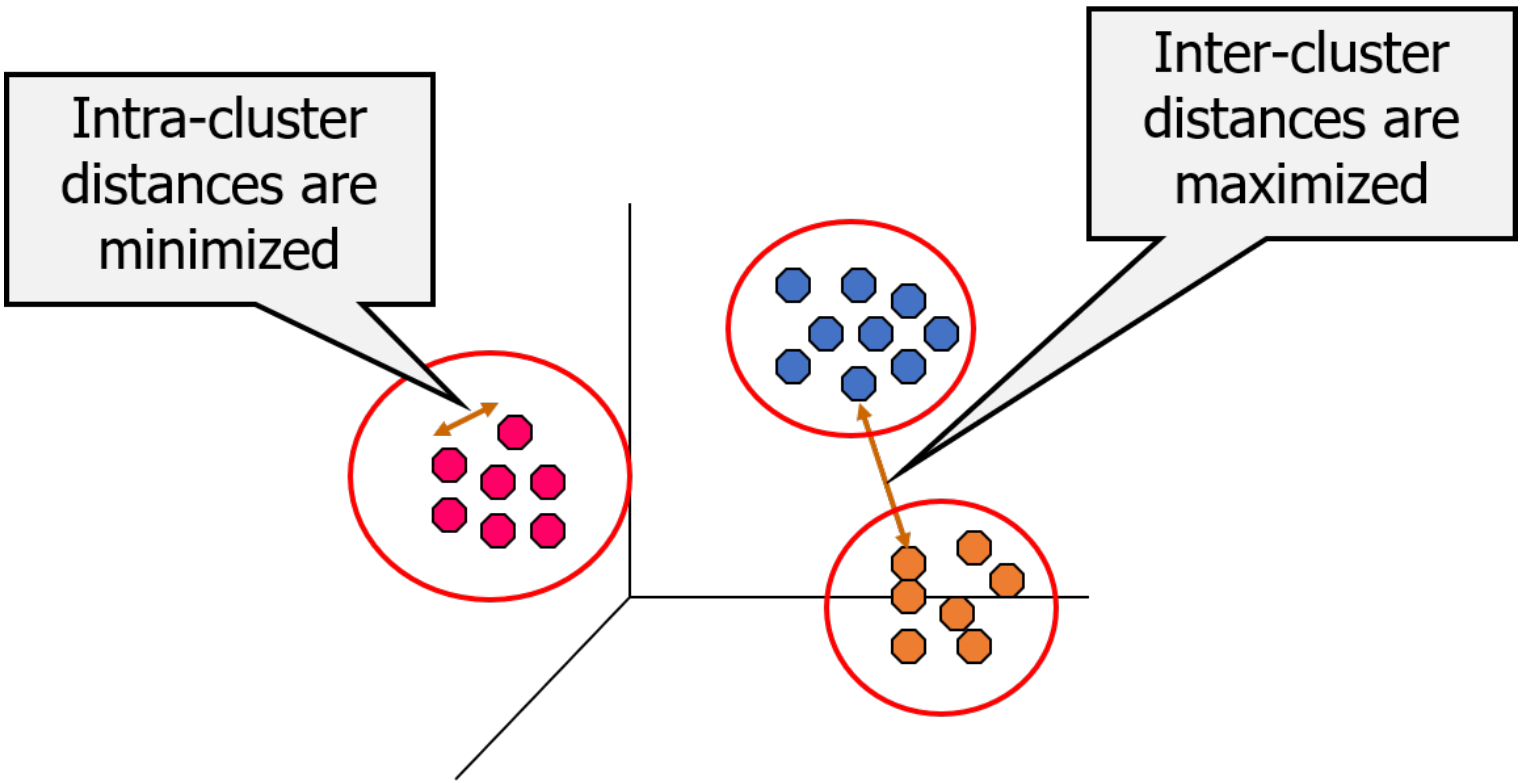


Section 2

Cluster Analysis

Cluster Analysis

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Cluster Analysis

- Cluster is a group of similar objects (cases, points, observations, examples, members, customers, patients, locations, etc)
- Finding the groups of cases/observations/objects in the population such that the objects are
  - Homogeneous within the group (high intra-class similarity)
  - Heterogeneous between the groups (low inter-class similarity)

Cluster Analysis

Clustering methods

- Hierarchical clustering
- Partitioning based clustering

Cluster Analysis

Hierarchical clustersing

Agglomerative Methods

- Begin with n-clusters (each record its own cluster)
- Keep joining records into clusters until one cluster is left (the entire data set)
- Most popular

Divisive Methods

- Start with one all-inclusive cluster
- Repeatedly divide into smaller clusters

Cluster Analysis

Distance measures

- To combine similar records together we need to measure the distance, similarity/dissimilarity between records and clusters
- Two groups of distance measures:
  1. Distance between records
  2. Distance between clusters

Doing in R: Hierarchical Clustering

- Read the data
- Set country names as row names for the DataFrame
- Select two columns for clustering
- Scale to z-scores
- Remove countries with NA values

```
countries = read.csv("data/countries.csv")

rownames(countries)=countries$ISO_code

countries_subset = countries[,c("Unemployment", "GDP.per.Capita.PPP")]

countries_subset = as.data.frame(scale(countries_subset,
                                       center=TRUE, scale=TRUE))

countries_subset = na.omit(countries_subset)
```

Doing in R: Hierarchical Clustering

- Calculate the distance matrix
- The hclust function takes distance matrix as a first argument

```
countries_distance = dist(countries_subset, method="euclidian")
```

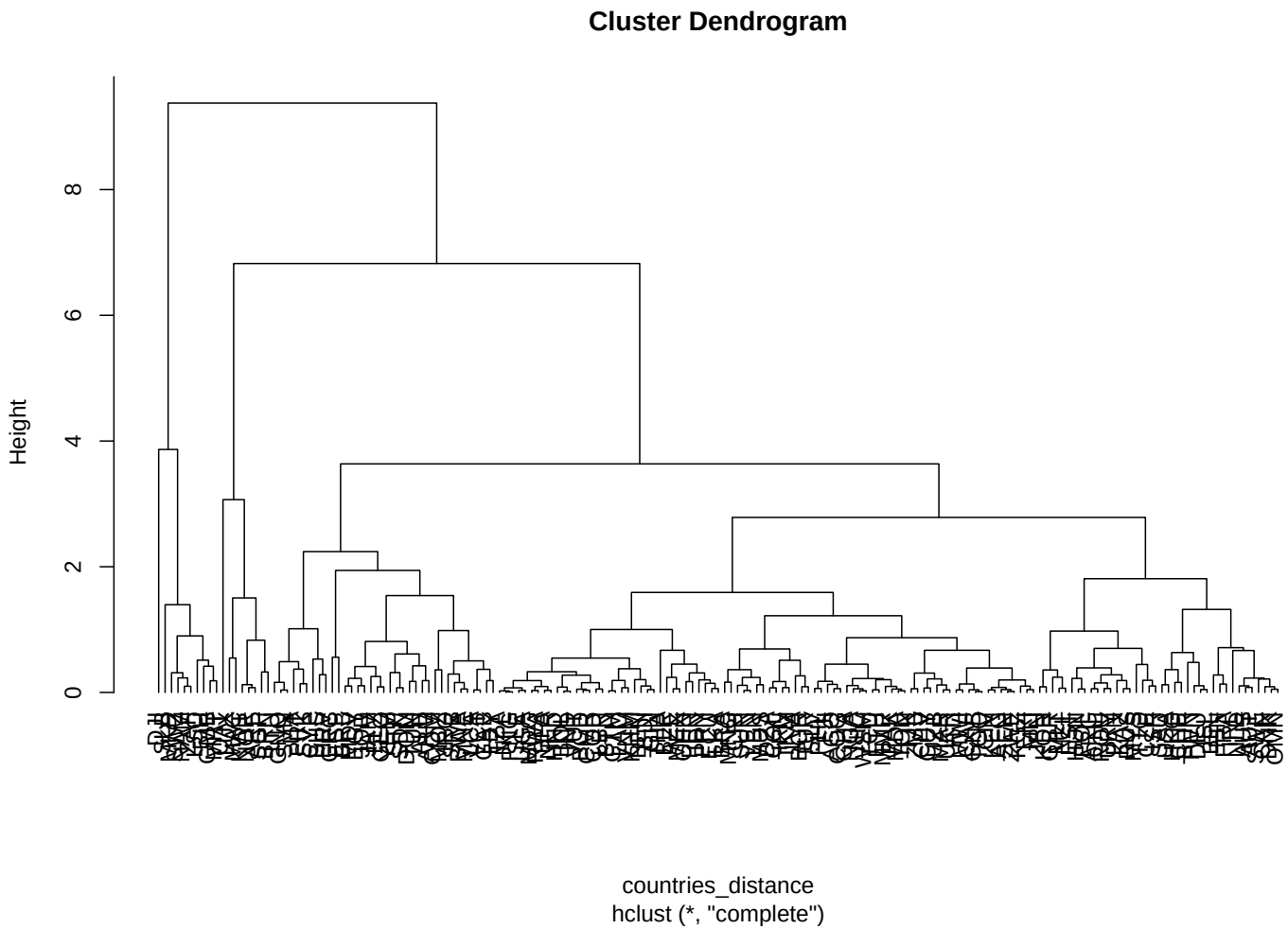
```
cl  =  hclust(countries_distance, method="complete")
cl
```

```
##
## Call:
## hclust(d = countries_distance, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 175
```

Cluster Analysis (Dendrogram)

Dendrogram represents how the clusters are created.

```
plot(cl, hang=-1)
```



Cluster Analysis (Dendrogram)

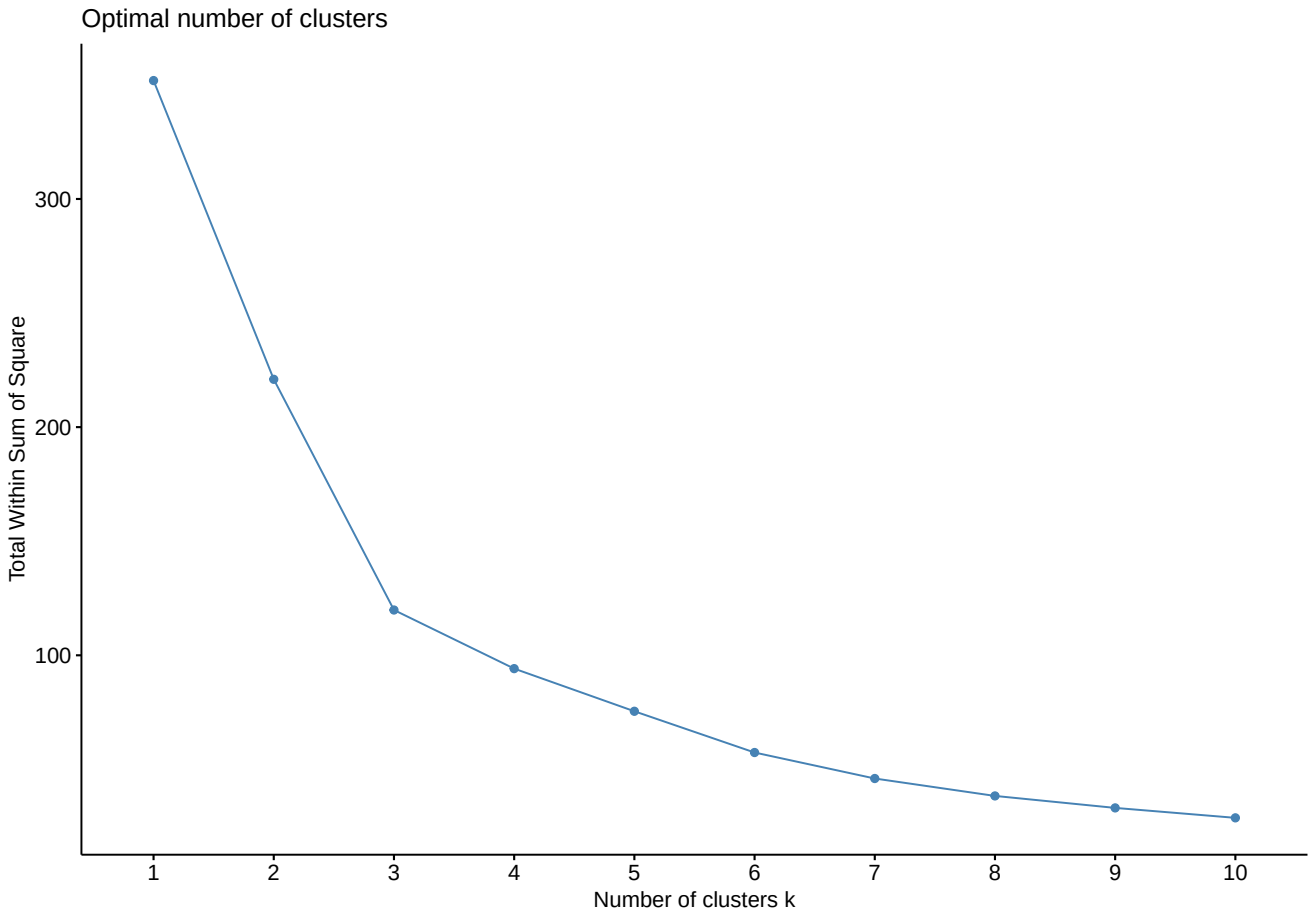
Understanding a Dendrogram

- **Hierarchical Structure:** A dendrogram visualizes hierarchical clustering, showing the arrangement of observations into nested clusters.
- **Height of Branches:** The height (or length) of the branches represents the distance or dissimilarity between clusters. Higher branches mean greater dissimilarity.
- **Merging Points:** Where branches join indicates clusters that have been combined. Lower merging points imply more similar clusters.

Cluster Analysis (Dendrogram)

How many clusters to choose? Look for an elbow.

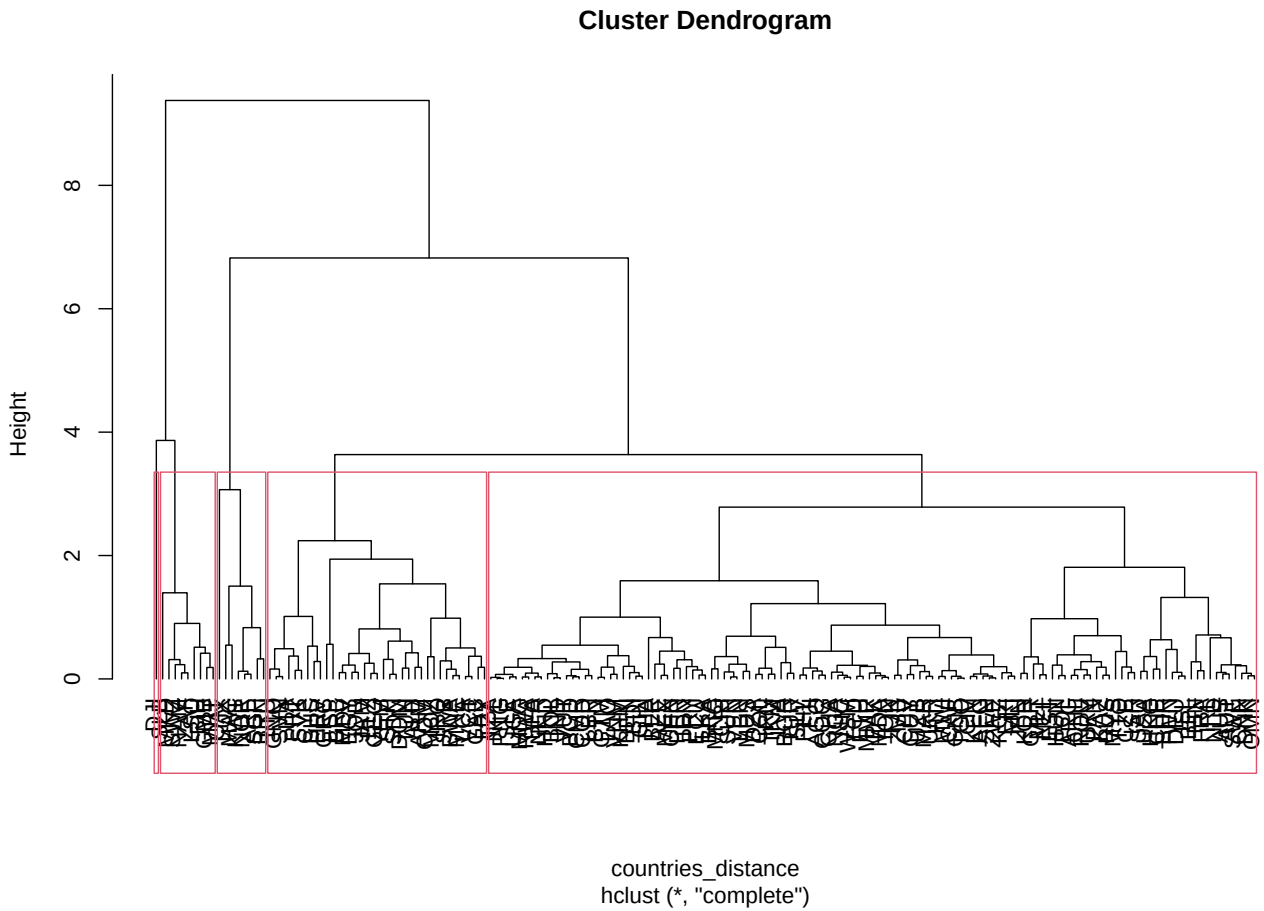
```
fviz_nbclust(countries_subset, FUNcluster = factoextra::hcut,
             method = "wss")
```



Cluster Analysis

If we choose **5** clusters

```
plot(cl, hang=-1)
rect.hclust(cl, 5)
```



Cluster Analysis

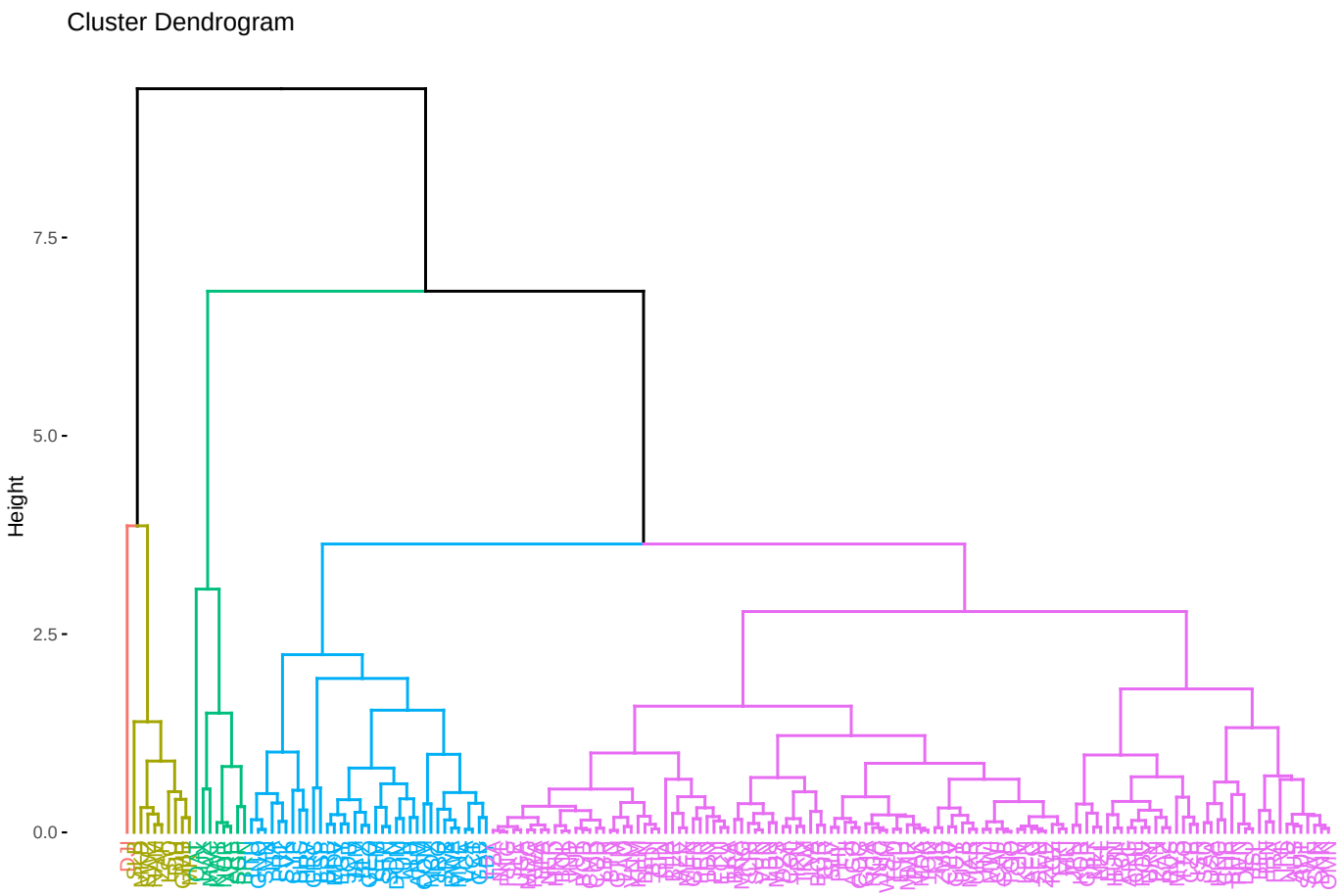
Interpreting a Dendrogram

- **Cutting the Dendrogram:** By "cutting" the dendrogram at a specific height, you can choose the number of clusters. Lower cuts result in more clusters, while higher cuts yield fewer.
- **Cluster Similarity:** Observations or groups of observations that merge at a lower height are more similar to each other compared to those merging at a higher height.
- **Cluster Size and Shape:** Dense clusters merge early, while more dispersed clusters join at higher levels, often indicating more variance within those groups.

Cluster analysis

Using factoviz library

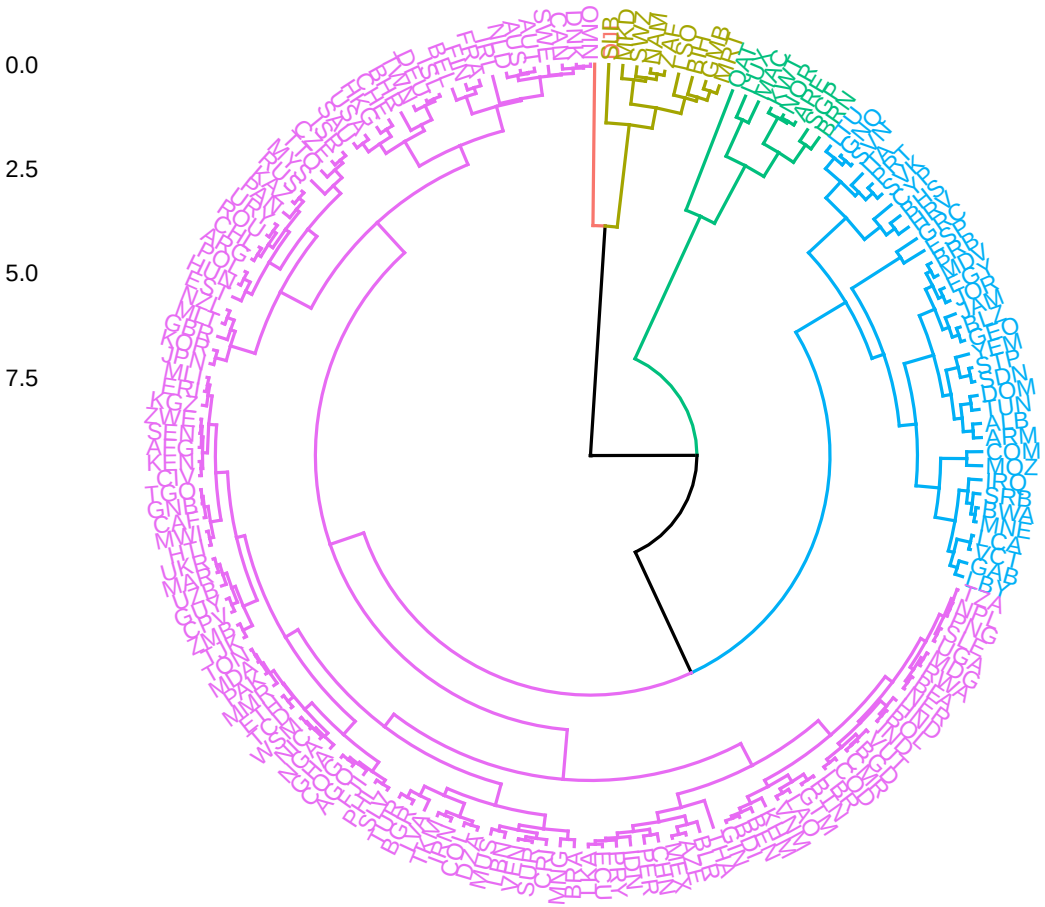
```
fviz_dend(cl, k = 5)
```



Cluster analysis

Circular dendrogram

```
fviz_dend(cl, k = 5, type = "circular")
```



Cluster analysis

We can create a new variable with cluster membership.

Cluster membership

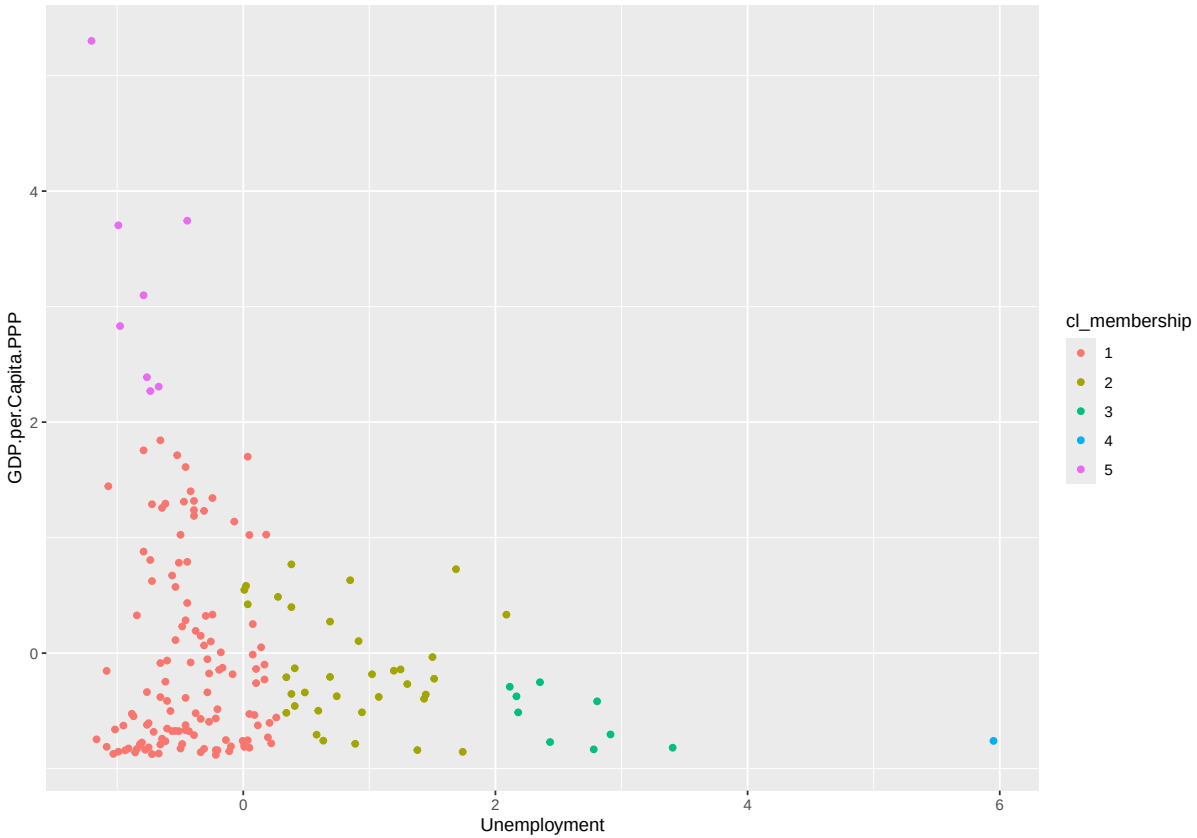
```
countries_subset$cl_membership = as.factor(cutree(cl, k=5))
table(countries_subset$cl_membership)
```

```
##
##   1   2   3   4   5
## 122  35   9   1   8
```

Cluster analysis

Visualizing clusters

```
ggplot(data = countries_subset, aes(x = Unemployment, y = GDP.per.Capita.PPP,
       color = cl_membership)) + geom_point()
```



Section 3

K-Means Clustering

Cluster Analysis: K-means

- Is used when the number of the clusters we are looking for is known beforehand
- We assume that the data is divided equally among the clusters

Cluster Analysis: K-means

The number of clusters is defined by the user

Given k, the *k-means* algorithm consists of four steps:

- Select initial centroids at random.
- Assign each object to the cluster with the nearest centroid.
- Compute each centroid as the mean of the objects assigned to it.
- Repeat previous 2 steps until no change.

Cluster Analysis: K-means

With this steps K-means algorithm **maximizes** Between group Sum of Squares and **minimizes** within group Sum of Squares.

Total Sum of Squares

$$SST = \sum_{j=1}^{m} \sum_{i=1}^{n} (y_{ij} - \bar{y}_m)^2$$

Between Groups Sum of Squares

$$SSB = \sum_{l=1}^{k} \sum_{j=1}^{m} (\bar{y}_{km} - \bar{y}_m)^2$$

Cluster Analysis: K-means

Within group sum of squares

$$SSW = \sum_{l=1}^{k} \sum_{j=1}^{m} \sum_{i=1}^{n} (y_{ijk} - \bar{y}_{km})^2$$

Where i is the number of the row/case, j number of the variables, k is the number of the groups/clusters

$$SST = SSB + SSW$$

Cluster Analysis: K-means

Doing in R

```
countries_kmeans = kmeans(countries_subset[,1:2], centers = 3)
```

What is inside ?

```
names(countries_kmeans)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     '
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Cluster Analysis: K-means

Shows the means/centers of each variable for each cluster

```
countries_kmeans$centers
```

```
##   Unemployment GDP.per.Capita.PPP
## 1    1.9010286          -0.3355714
## 2   -0.5249025           1.6604855
## 3   -0.2916477          -0.4107595
```

Cluster Analysis: K-means

How good is the clustering?

```
countries_kmeans$betweenss/countries_kmeans$totss
```

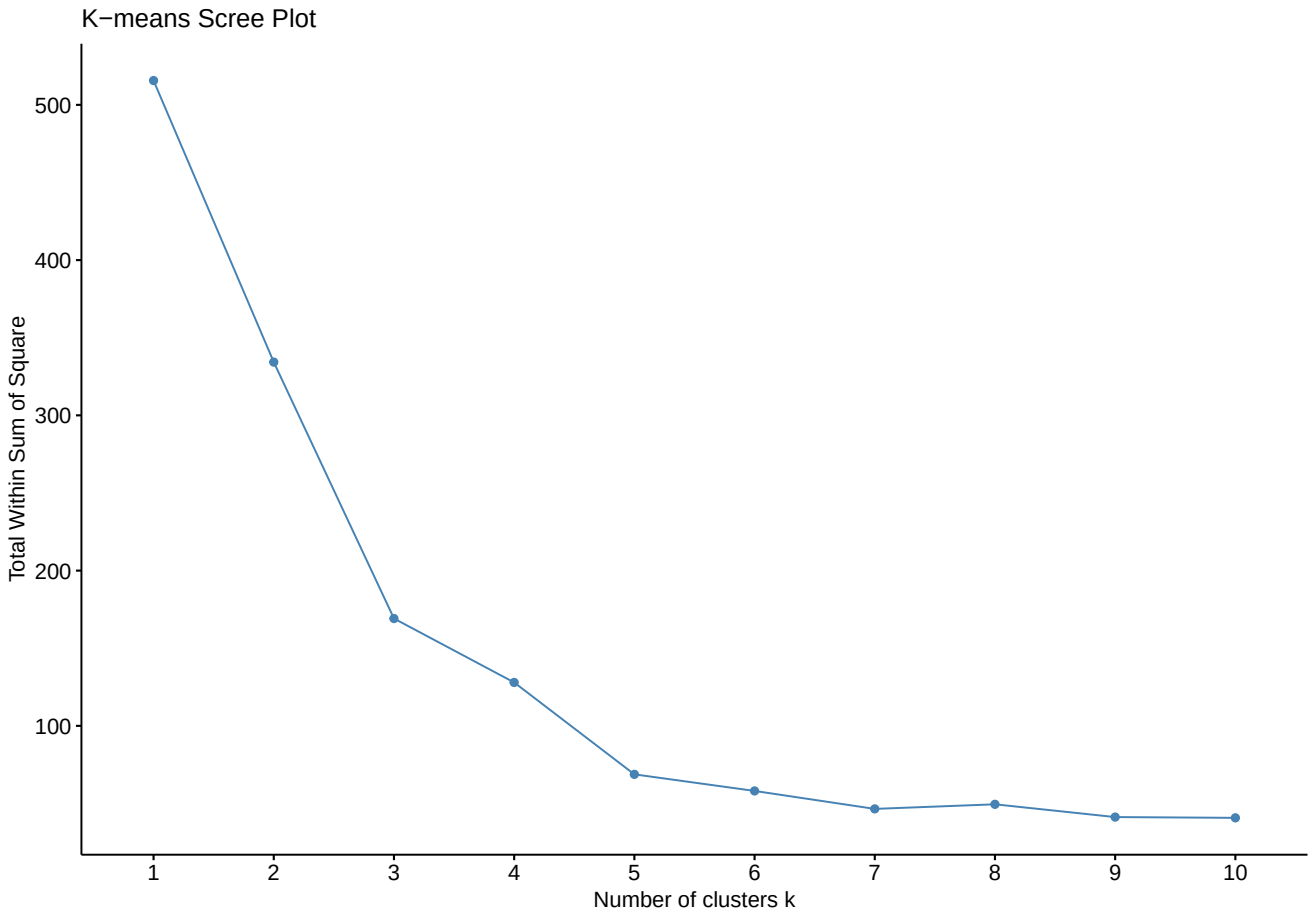
```
## [1] 0.6689472
```

66.7% of the total variance in the data can be explained by the clusters

Cluster Analysis: K-means

How many clusters to chose?

```
fviz_nbclust(countries_subset, kmeans, method = "wss") +
  labs(title = "K-means Scree Plot")
```



Cluster Analysis: K-means

Look at the cluster membership

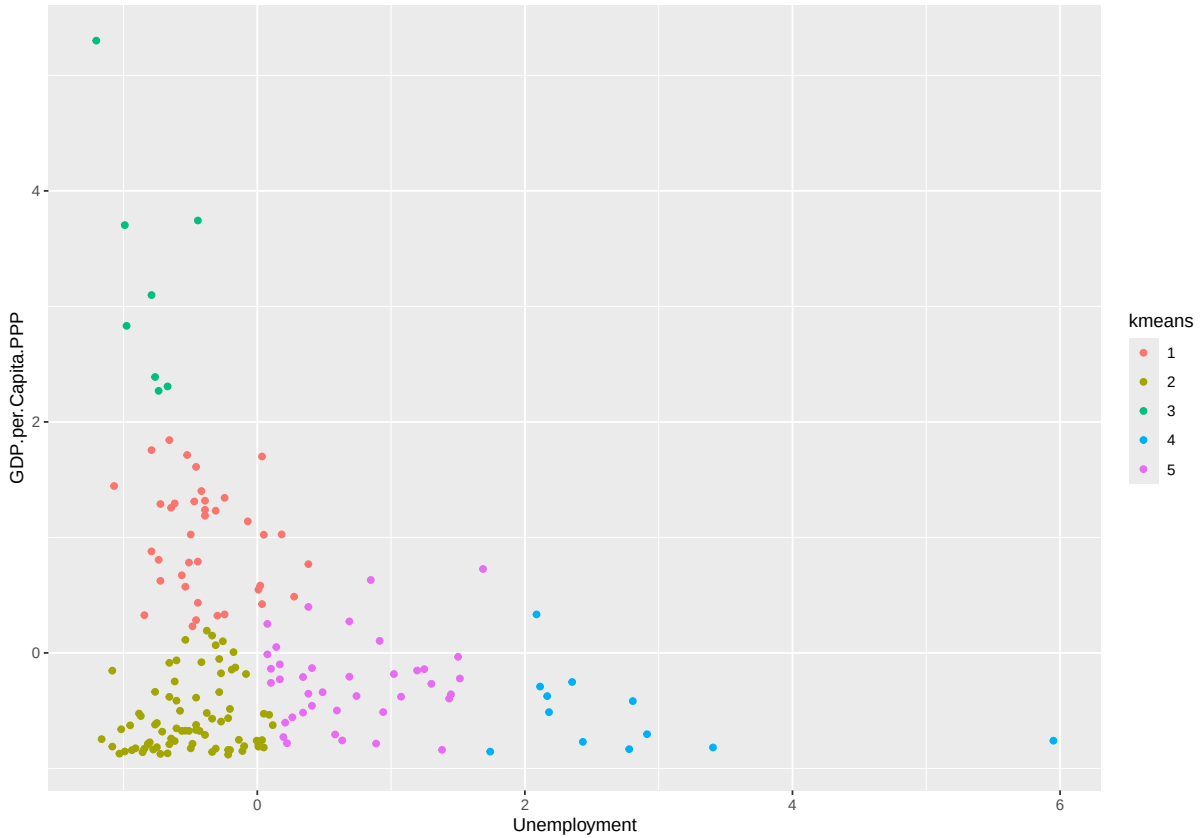
```
countries_kmeans = kmeans(countries_subset[,1:2], centers = 5)
countries_subset$kmeans = as.factor(countries_kmeans$cluster)
table(countries_subset$kmeans)
```

```
##
##  1  2  3  4  5
## 38 78  8 12 39
```

Cluster Analysis: K-means

Visualizing the clusters

```
ggplot(data = countries_subset, aes(x = Unemployment,
      y = GDP.per.Capita.PPP, color = kmeans)) + geom_point()
```



Section 4

Dendrogram Heatmaps

Dendrogram Heatmaps

A **dendrogram heatmap** is a combined visualization that merges a heatmap with dendrograms to represent hierarchical clustering.

- **Heatmap Component:** A matrix-like visual where colors represent the intensity of values in the data, allowing quick comparison across rows and columns.
- **Dendrogram Component:** Tree-like structures on the sides of the heatmap (typically on rows and columns) that illustrate hierarchical clustering, grouping similar data points together.
  - **Clustering by Rows and Columns:** Both rows (data points) and columns (features) can be clustered, allowing for simultaneous grouping based on both samples and variables.

Dendrogram Heatmaps

- **Hierarchical Clustering Integration:** The dendrograms reorder the rows and columns of the heatmap based on similarity, making patterns and clusters more evident.
- **Purpose:** Provides a dual insight by showing both the actual data values (via color) and the relationships between data points (via clusters).
- **Usefulness:** Makes it easier to identify groups, trends, and outliers in complex datasets by visually clustering similar items.

Dendrogram Heatmaps

What a Dendrogram Heatmap Helps Us Achieve

- **Identify Patterns and Clusters:** Combines heatmap intensity with hierarchical clustering, making it easy to spot groups or clusters of similar data points.
- **Reveal Data Structure:** Shows how data points or features relate to each other in a hierarchical way, highlighting nested groupings.
- **Simplify Complex Data:** Reduces complexity by visually grouping related rows and columns, making large datasets easier to interpret.

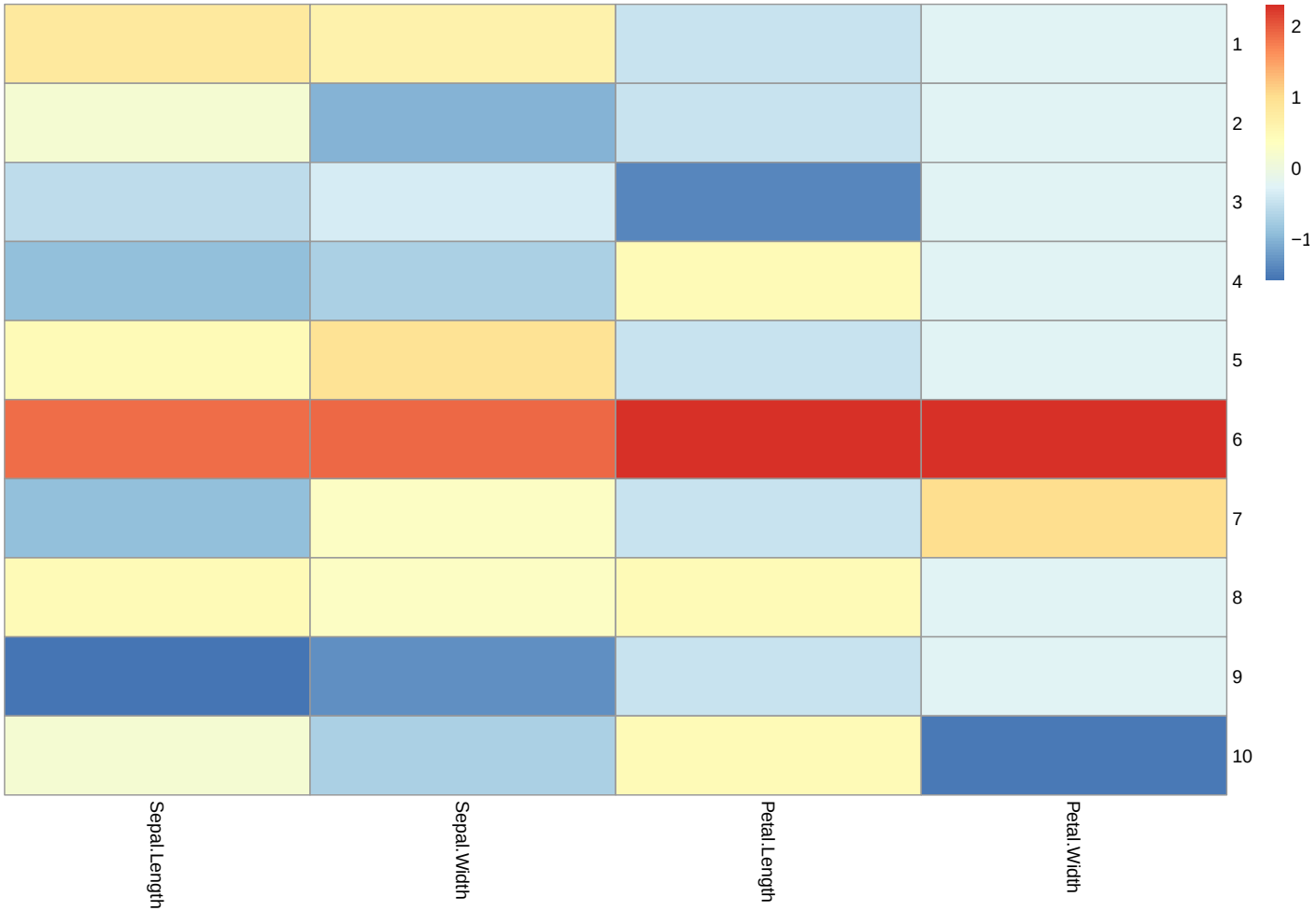
Dendrogram Heatmaps

Get subset of iris data and scale it

```
iris_scaled <- iris[1:10, -5]
iris_scaled <- scale(iris_scaled)
```

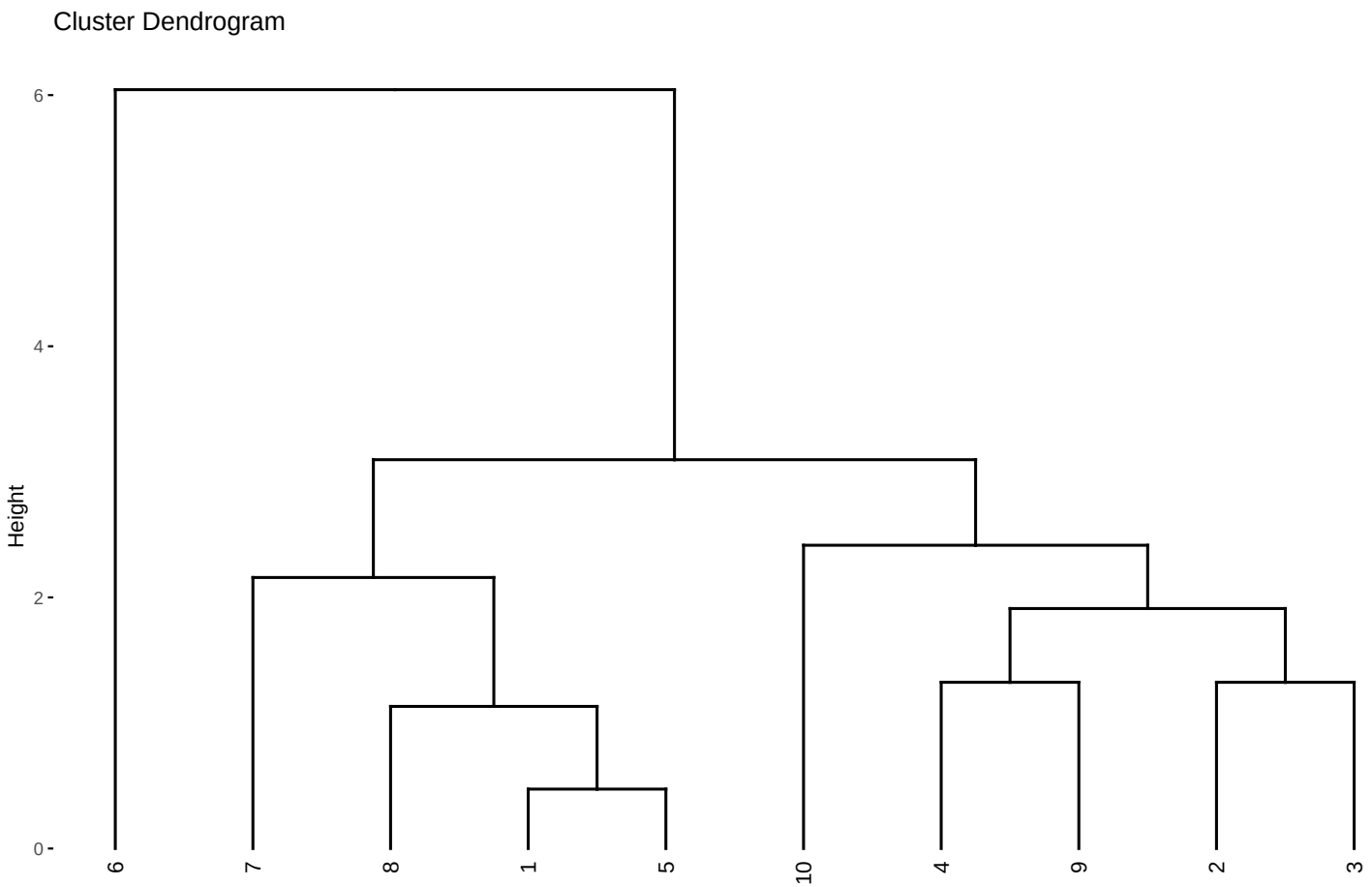
Dendrogram Heatmaps

```
pheatmap(iris_scaled, cluster_rows = FALSE, cluster_cols = FALSE)
```



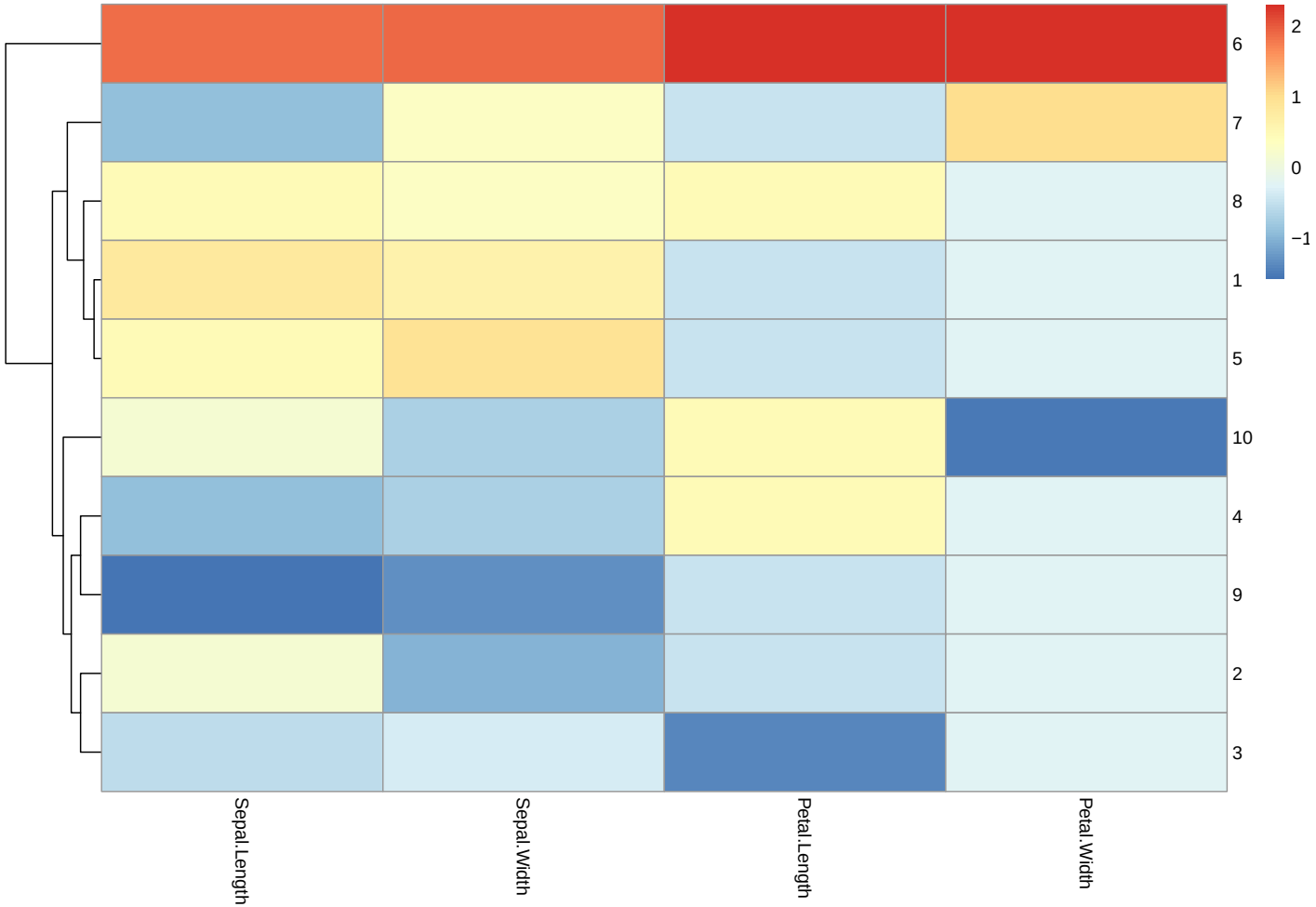
Dendrogram Heatmaps

```
iris_clust = hclust(d = dist(iris_scaled), method = "complete")
fviz_dend(iris_clust)
```



Dendrogram Heatmaps

```
pheatmap(iris_scaled, cluster_rows = T, cluster_cols = F)
```



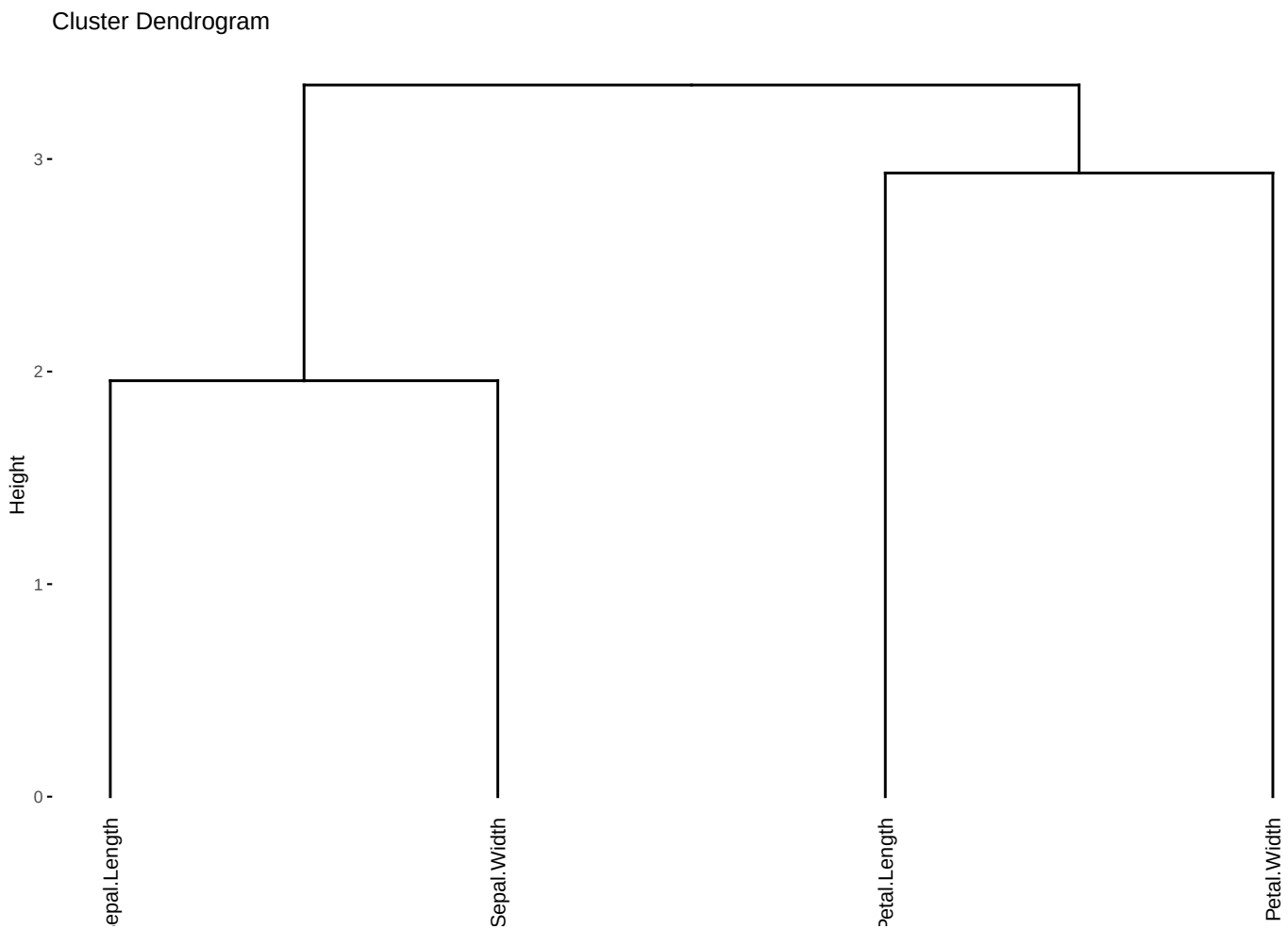
Dendrogram Heatmaps

- transpose iris_scaled
- calculate the distance matrix
- cluster

```
iris_scaled_t = t(iris_scaled)
iris_clust_t = hclust(d = dist(iris_scaled_t))
```

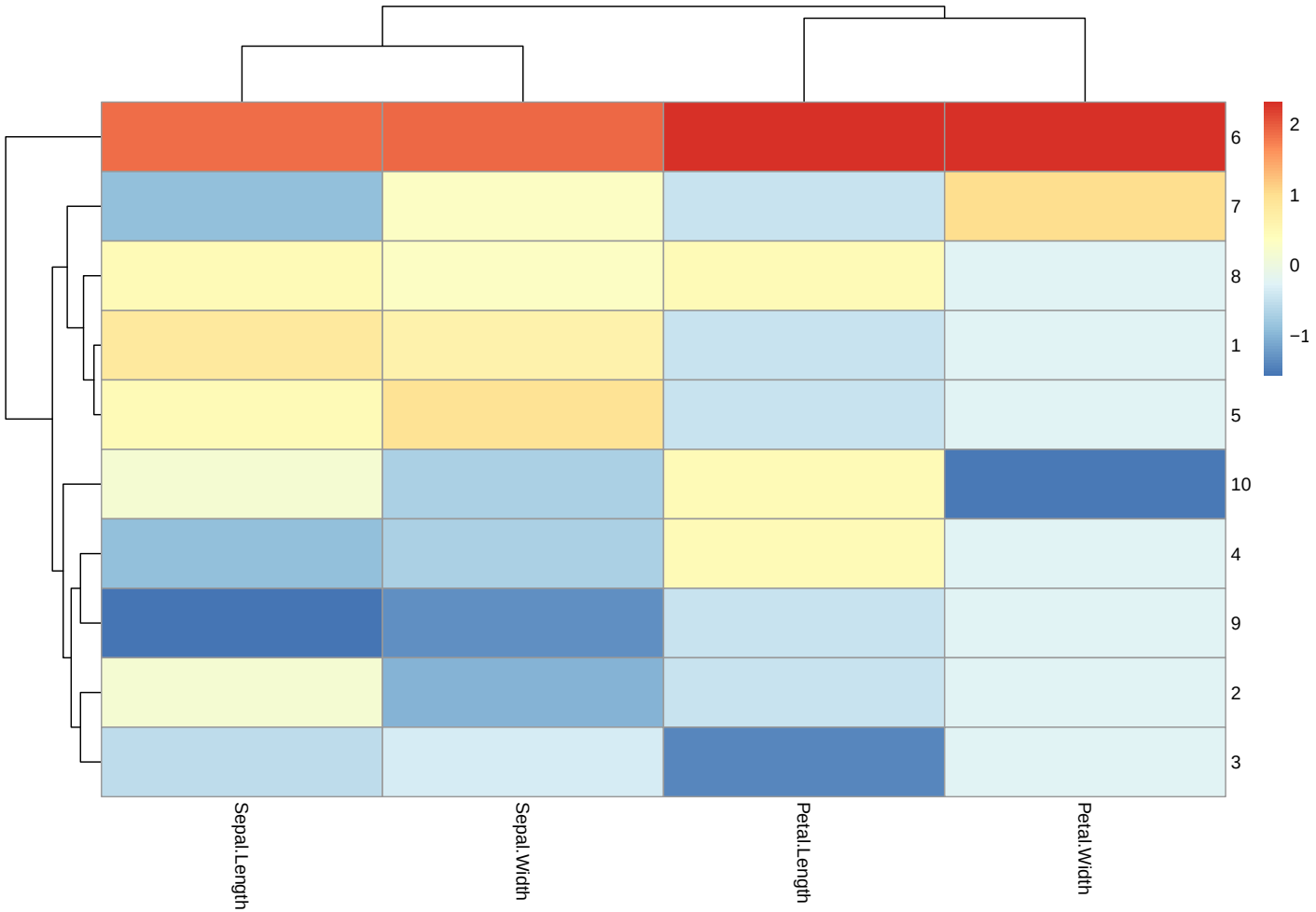
Dendrogram Heatmaps

```
fviz_dend(iris_clust_t, repel = T)
```



Dendrogram Heatmaps

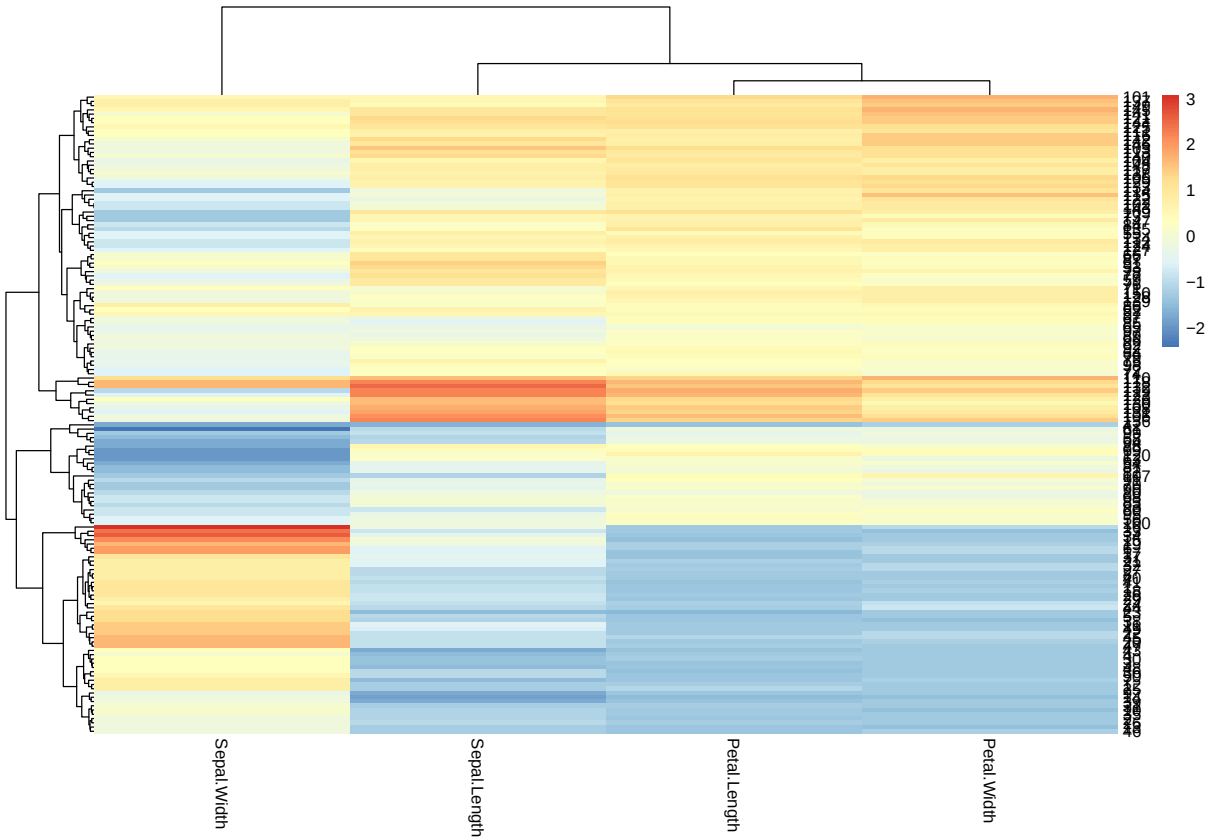
```
pheatmap(iris_scaled, cluster_rows = T, cluster_cols = T)
```



Dendrogram Heatmaps

Lets look at the whole dataset

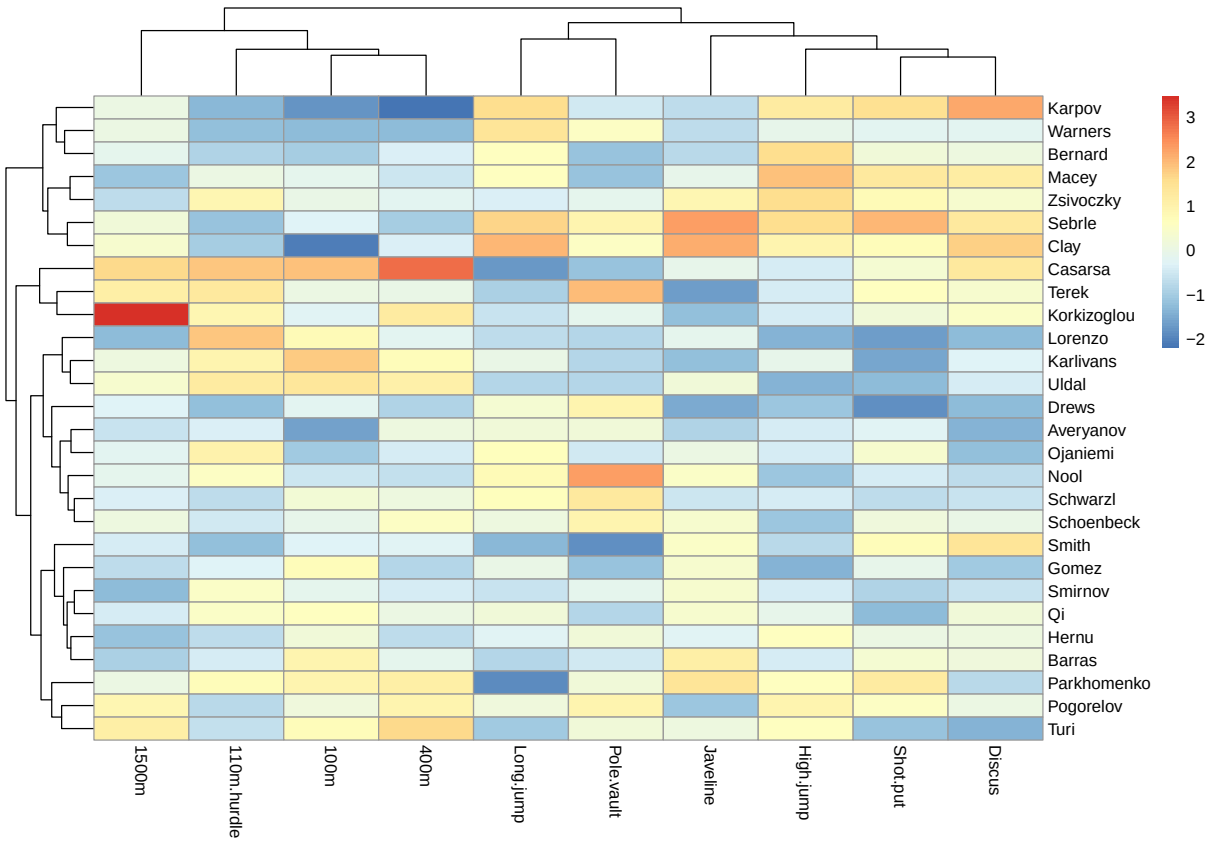
```
iris_scaled = scale(iris[,-5]) %>% as.data.frame()
pheatmap(iris_scaled, cluster_rows = T, cluster_cols = T)
```



Dendrogram Heatmaps

Decathlon dataset

```
decathlon_scaled = scale(decathlon[,1:10])
pheatmap(decathlon_scaled, cluster_rows = T, cluster_cols = T)
```



Section 5

PCA and Cluster Analysis

PCA and Cluster Analysis

Benefits of Using PCA and Clustering Together

- **Simplifies High-Dimensional Data:** PCA reduces the data's complexity, summarizing it into a few principal components, which makes it easier to visualize and interpret.
- **Enhanced Visualization of Clusters:** By reducing the data to two or three dimensions, clusters become more visually distinct, helping reveal group structures that may not be apparent in the original data.
- **Highlights Key Patterns:** PCA brings out the most influential variables, while clustering groups similar observations, making both patterns and outliers more evident.
- **Reduces Visual Clutter:** With PCA simplifying dimensions and clustering organizing the data, the result is a cleaner, more informative visualization.

PCA and Cluster Analysis

The dataset contains credit card usage behavior of about 9000 active credit card holders.

Here is the link to the dataset
https://www.kaggle.com/datasets/arjunbhasin2013/ccdata

```
cc_data = read.csv("Data/CC GENERAL.csv")
str(cc_data)
```

```
## 'data.frame':    8950 obs. of  18 variables:
##  $ CUST_ID                         : chr  "C10001" "C10002" "C10003" "C10004" ...
##  $ BALANCE                         : num  40.9 3202.5 2495.1 1666.7 817.7 ...
##  $ BALANCE_FREQUENCY               : num  0.818 0.909 1 0.636 1 ...
##  $ PURCHASES                       : num  95.4 0 773.2 1499 16 ...
##  $ ONEOFF_PURCHASES                : num  0 0 773 1499 16 ...
##  $ INSTALLMENTS_PURCHASES          : num  95.4 0 0 0 0 ...
##  $ CASH_ADVANCE                    : num  0 6443 0 206 0 ...
##  $ PURCHASES_FREQUENCY             : num  0.1667 0 1 0.0833 0.0833 ...
##  $ ONEOFF_PURCHASES_FREQUENCY      : num  0 0 1 0.0833 0.0833 ...
##  $ PURCHASES_INSTALLMENTS_FREQUENCY: num  0.0833 0 0 0 0 ...
##  $ CASH_ADVANCE_FREQUENCY          : num  0 0.25 0 0.0833 0 ...
##  $ CASH_ADVANCE_TRX                : int  0 4 0 1 0 0 0 0 0 0 ...
##  $ PURCHASES_TRX                   : int  2 0 12 1 1 8 64 12 5 3 ...
##  $ CREDIT_LIMIT                    : num  1000 7000 7500 7500 1200 1800 13500 2300 7000 11000 ...
##  $ PAYMENTS                        : num  202 4103 622 0 678 ...
##  $ MINIMUM_PAYMENTS                : num  140 1072 627 NA 245 ...
##  $ PRC_FULL_PAYMENT                : num  0 0.222 0 0 0 ...
##  $ TENURE                          : int  12 12 12 12 12 12 12 12 12 12 ...
```

PCA and Cluster Analysis

Look at the missing values:

You can:

- remove cases with missing values
- do missing value imputation

```
sapply(cc_data, function(x) sum(is.na(x)))
```

```
##                          CUST_ID                          BALANCE
##                                0                                0
##                BALANCE_FREQUENCY                        PURCHASES
##                                0                                0
##                ONEOFF_PURCHASES            INSTALLMENTS_PURCHASES
##                                0                                0
##                    CASH_ADVANCE               PURCHASES_FREQUENCY
##                                0                                0
##       ONEOFF_PURCHASES_FREQUENCY PURCHASES_INSTALLMENTS_FREQUENCY
##                                0                                0
##           CASH_ADVANCE_FREQUENCY                  CASH_ADVANCE_TRX
##                                0                                0
##                    PURCHASES_TRX                      CREDIT_LIMIT
##                                0                                1
##                         PAYMENTS                  MINIMUM_PAYMENTS
##                                0                              313
##                PRC_FULL_PAYMENT                           TENURE
##                                0                                0
```

PCA and Cluster Analysis

Data cleaning:

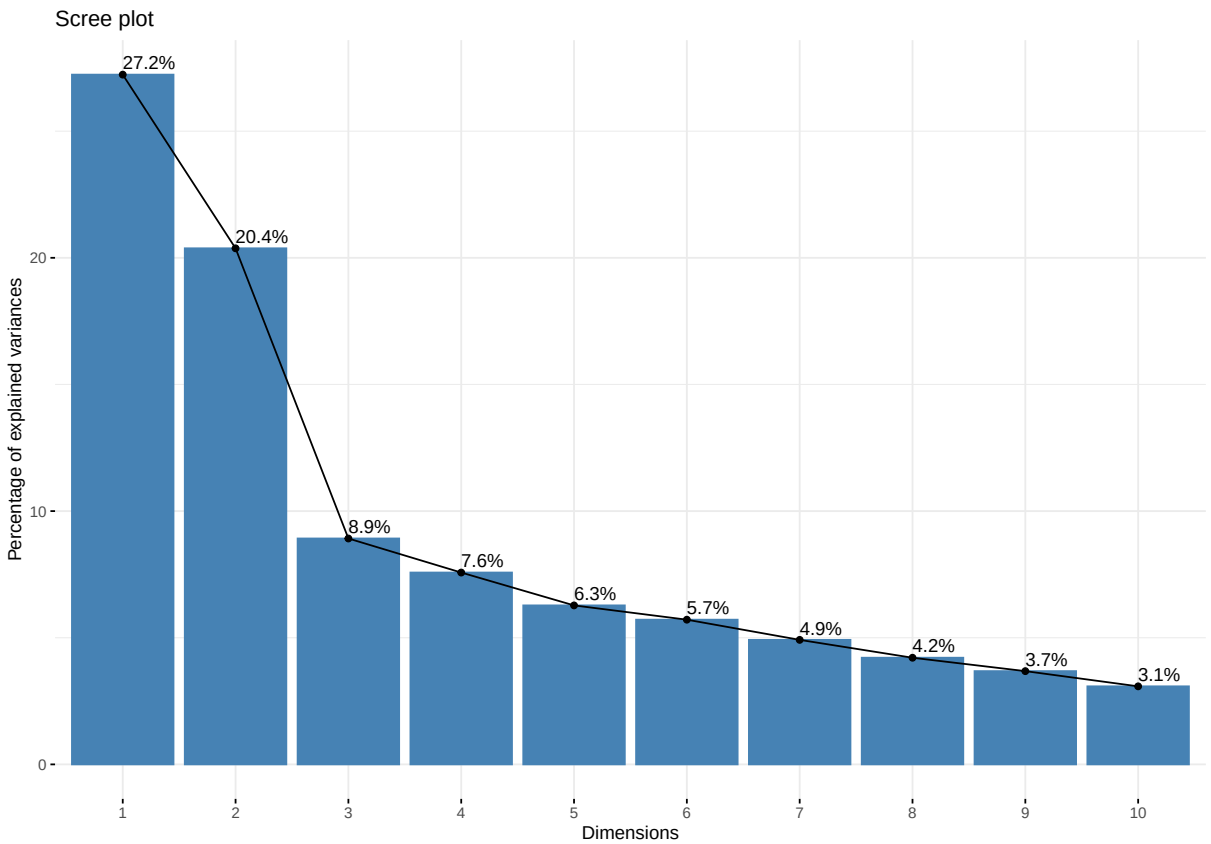
- Remove missing cases with missing values
- Remove variable CUST_ID
- Do z-score scaling

```
cc_data = na.omit(cc_data)
cc_data$CUST_ID = NULL
cc_data_scaled = scale(cc_data) %>% as.data.frame()
```

PCA and Cluster Analysis

Run PCA and visualize variance explained

```
cc_pca = prcomp(cc_data_scaled)
fviz_screeplot(cc_pca, addlabels = T)
```



PCA and Cluster Analysis

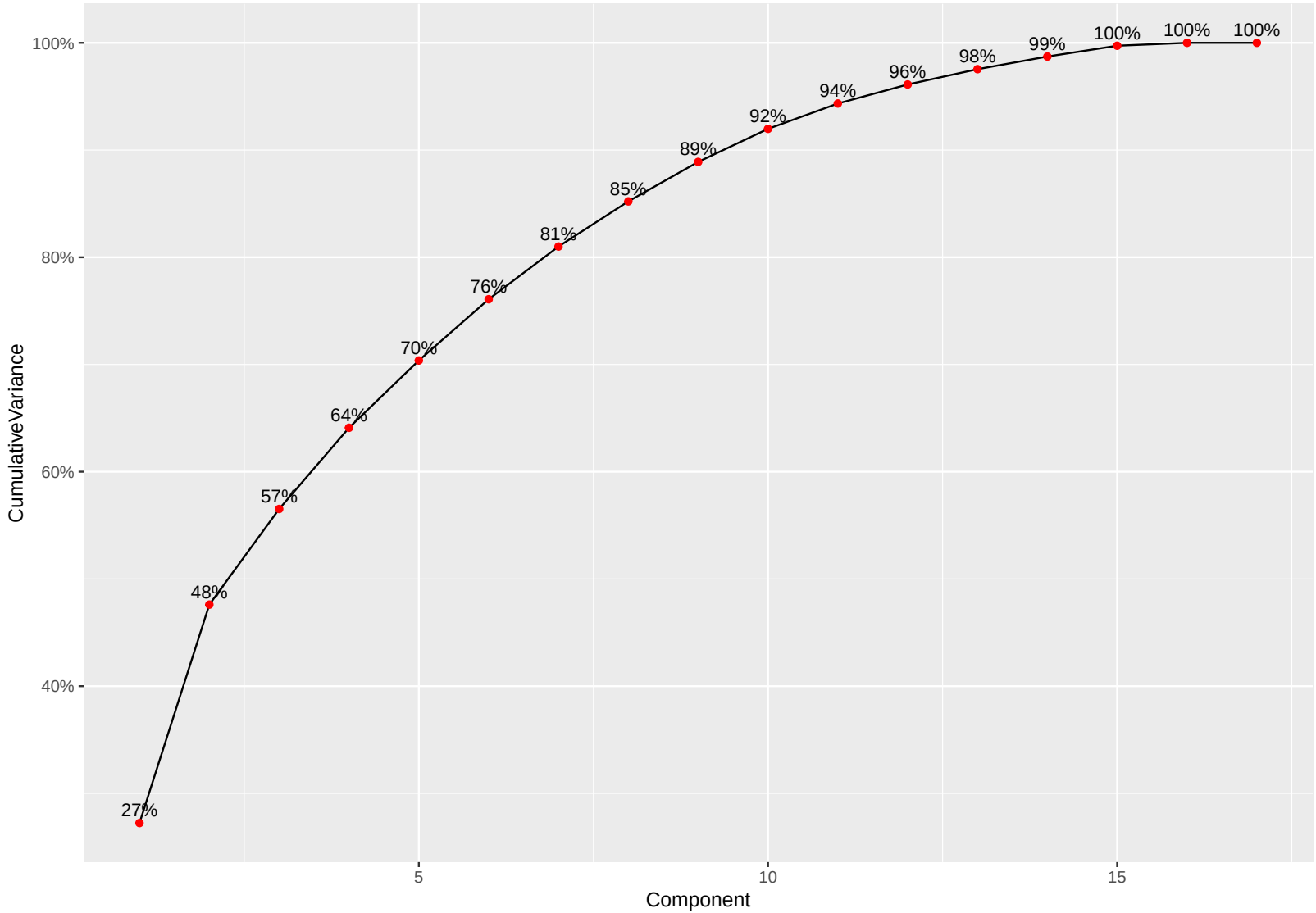
- Extract cumulative variance explained
- Make scree plot for **cumulative** variance explained

```
var_explained <- cc_pca$sdev^2 / sum(cc_pca$sdev^2)
cum_var_explained <- cumsum(var_explained)
var_df <- data.frame(
  Component = 1:length(cum_var_explained),
  CumulativeVariance = cum_var_explained
)
```

PCA and Cluster Analysis

```
ggplot(var_df, aes(x = Component, y = CumulativeVariance)) +
  geom_line() + geom_point(color = "red")+
  geom_text(aes(label = scales::percent(CumulativeVariance,
     accuracy = 1)), vjust = -0.5, color = "black", size = 3.5) +
  scale_y_continuous(labels = scales::percent_format())
```

PCA and Cluster Analysis



PCA and Cluster Analysis

We will chose 7 components and do k-mean clustering on this components

```
cc_pca_data = cc_pca$x[,1:7] %>% as.data.frame()
head(cc_pca_data)
```

```
##          PC1        PC2         PC3         PC4         PC5         PC6
## 1 -1.6962971 -1.1225190  0.49153311  0.71947913  0.07982586 -0.11822694
## 2 -1.2156104  2.4354968  0.69461763 -0.09883702  0.80297229  0.91772406
## 3  0.9357991 -0.3851793 -0.02595178  1.29376862 -1.98717027  0.68209981
## 5 -1.6145448 -0.7245442  0.27234236  1.08605360 -0.42778877 -0.08297746
## 6  0.2236877 -0.7835645 -1.18436576  0.72131105  0.80119651 -0.52584901
## 7  6.2652350 -0.6094139  2.08544348 -0.57775168 -0.96556154  1.09127194
##          PC7
## 1 -0.8089459
## 2  0.3229504
## 3  1.6246273
## 5 -0.6869609
## 6 -0.7888471
## 7  1.3689962
```

PCA and Cluster Analysis

- K-means and let us go with 6 clusters
- Add cluster membership variable to the dataframe
- Calculate cluster quality score

```
cc_kmeans = kmeans(cc_pca_data, centers = 6)
cc_pca_data$kmeans = as.factor(cc_kmeans$cluster)
cc_kmeans$betweenss/cc_kmeans$totss
```

```
## [1] 0.5203228
```

PCA and Cluster Analysis

```
ggpairs(cc_pca_data, columns = 1:6, mapping = aes(color = kmeans),
        progress = F)
```