Vertiefende statistische Verfahren

5. Übungsblatt SS 2024

Stefan Kolb, Joachim Waltl

Allgemeine Information

Alle Aufgaben sind mit R zu lössen, wenn nicht explizit anders angegeben. Die Berechnungen sollen nachvollziehbar und dokumentiert sein. Um die vollständige Punktezahl zu erreichen, müssen alle Ergebnisse und Fragen entsprechend interpretiert bzw. beantwortet werden. Code alleine ist nicht ausreichend! Die Abgabe erfolgt über Moodle entsprechend der Abgaberichtlinien als pdf und Rmd File. Bitte inkludieren Sie namentlich alle beteiligten Gruppenmitglieder sowohl im Bericht als auch im Source Code. Die jeweiligen Datensätze die für diese Übung relevant sind finden Sie ebenfalls in Moodle.

1 Clustering - Diabetes[3P]

Verwenden Sie den Datensatz `diabetes_RM.csv`. Der Datensatz enthält fünf Messungen, die an 145 nicht adipösen erwachsenen Patienten durchgeführt wurden (Beschreibung siehe UE4). Reaven und Miller [ref] wendeten in Anlehnung an Friedman und Rubin (1967) eine Clusteranalyse auf die drei primären Variablen (`insulin`,`glucose` und `sspg`) an und identifizierten drei Cluster: "normal", "chemical" und "overt" diabetische Probanden. Die Variable `group` enthält die Klassifizierungen der Probanden in diese drei Gruppen und dient hier als Ground Truth.

- Führen Sie eine Clusteranalyse durch. Verwenden Sie eine Clusteranzahl von 3 und vergleichen Sie die Genauigkeit (gegenüber Ground Truth) folgender Cluster-Algorithmen:
  - k-means, k-medoids
  - hierarchisches Clustering
  - hierarchischer k-means
  - Modell-basiertes Clustering

```r
load_source()

# Load the data
diabetes <- read.csv("diabetes_RM.csv", header = TRUE, sep = ",")
groups <- c("normal", "chemical", "overt")

# Scale the selected columns
diabetes_scaled_cols <- scale(diabetes[, c("rw", "fpg", "glucose", "insulin", "sspg")])

# Create a new data frame from the scaled columns
diabetes_scaled_df <- as.data.frame(diabetes_scaled_cols)

# Set the row names of the scaled data frame to match the row names of the original data frame
rownames(diabetes_scaled_df) <- rownames(diabetes)
```

```r
# Bind the scaled columns with the unscaled columns
diabetes_scaled <- cbind(diabetes_scaled_df, group = diabetes$group)



# Perform k-means clustering
kmeans_result <- kmeans(diabetes_scaled_cols, centers = 3)

# Perform k-medoids clustering
kmedoids_result <- pam(diabetes_scaled_cols, k = 3)

# Perform hierarchical clustering
hclust_result <- cutree(hclust(dist(diabetes_scaled_cols)), k = 3)

# Perform hierarchical k-means clustering
hkmeans_result <-hkmeans(diabetes_scaled_cols, 3)

# Perform model-based clustering
mclust_result <- Mclust(diabetes_scaled_cols, G = 3)



kmeans_cluster <- kmeans_result$cluster
kmedoids_cluster <- kmedoids_result$cluster
hclust_cluster <- hclust_result
hkmeans_cluster <- hkmeans_result$cluster
mclust_cluster <- mclust_result$classification


# Create a named vector for recoding
recode_vector <- setNames(1:length(groups), groups)

# Recode the 'group' variable to the matching cluster
ground_truth <- recode(diabetes$group, !!!recode_vector)

#ground_truth <- diabetes$group

# Create a list of the cluster vectors
list_of_cluster_vectors <- list(ground_truth = ground_truth,
                     kmeans = kmeans_cluster,
                     kmedoids = kmedoids_cluster,
                     hclust = hclust_cluster,
                     hkmeans = hkmeans_cluster,
                     mclust = mclust_cluster)



kmeans_accuracy <- calculate_accuracy(diabetes_scaled, kmeans_cluster, groups)
```

```
##           [,1]     [,2]     [,3]
## [1,] 0.3621507 4.658219 1.804704
## [2,] 1.9576353 4.285744 0.323201
## [3,] 3.6043677 1.127521 3.027601
## [1] 1 3 2
```

```
kmedoids_accuracy <- calculate_accuracy(diabetes_scaled, kmedoids_cluster, groups)
```

```
##           [,1]      [,2]     [,3]
## [1,] 0.2750971 1.9263863 4.731498
## [2,] 1.8825475 0.4080579 4.383376
## [3,] 3.5838033 2.9547495 1.223532
## [1] 1 2 3
```

```
hclust_accuracy <- calculate_accuracy(diabetes_scaled, hclust_cluster, groups)
```

```
##           [,1]     [,2]      [,3]
## [1,] 0.4881984 4.048269 3.9939866
## [2,] 1.1937538 2.880662 3.6054722
## [3,] 3.3203439 5.118442 0.4326986
## [1] 1 2 3
```

```
hkmeans_accuracy <- calculate_accuracy(diabetes_scaled, hkmeans_cluster, groups)
```

```
##          [,1]      [,2]      [,3]
## [1,] 0.371128 1.7749877 4.1700617
## [2,] 1.977689 0.2475693 3.7832967
## [3,] 3.637292 3.2219678 0.6135777
## [1] 1 2 3
```

```
mclust_accuracy <- calculate_accuracy(diabetes_scaled, mclust_cluster, groups)
```

```
##            [,1]      [,2]      [,3]
## [1,] 0.09247276 1.8287537 4.0804257
## [2,] 1.72876397 0.2333928 3.6906429
## [3,] 3.59758685 3.0974983 0.5223698
## [1] 1 2 3
```

```
# Put the results into a list
accuracy_results <- list(
  ground_truth = 1,
  kmeans = kmeans_accuracy,
  kmedoids = kmedoids_accuracy,
  hclust = hclust_accuracy,
  hkmeans = hkmeans_accuracy,
  mclust = mclust_accuracy
)



# Print the accuracy results
cat("k-means accuracy: ", kmeans_accuracy, "\n")
```

```
## k-means accuracy:  0.7517241
```

```r
cat("k-medoids accuracy: ", kmedoids_accuracy, "\n")
```

```
## k-medoids accuracy:  0.7517241
```

```r
cat("Hierarchical clustering accuracy: ", hclust_accuracy, "\n")
```

```
## Hierarchical clustering accuracy:  0.7448276
```

```r
cat("Hierarchical k-means accuracy: ", hkmeans_accuracy, "\n")
```

```
## Hierarchical k-means accuracy:  0.7862069
```

```r
cat("Model-based clustering accuracy: ", mclust_accuracy, "\n")
```

```
## Model-based clustering accuracy:  0.862069
```

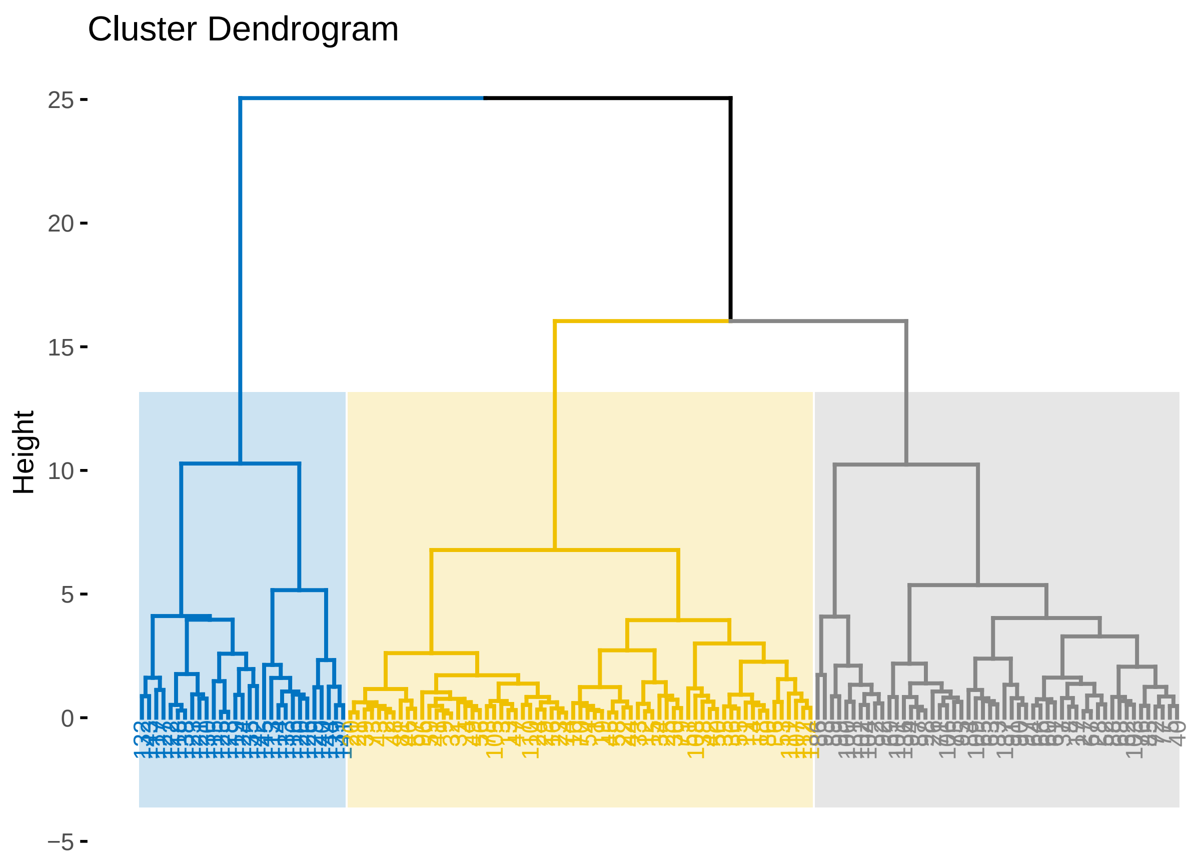
- Welches Verfahren ist am besten geeignet?

Model-based Clustering erreicht mit etwa 86% die höchste Genauigkeit. Auf dem zweiten Platz liegen gleich auf das k-means Clustering und das hierarchische k-means Clustering mit etwa 78% Genauigkeit. Es folgen mit etwa 75% Genauigkeit das k-medoids Clustering und das hierarchische Clustering.

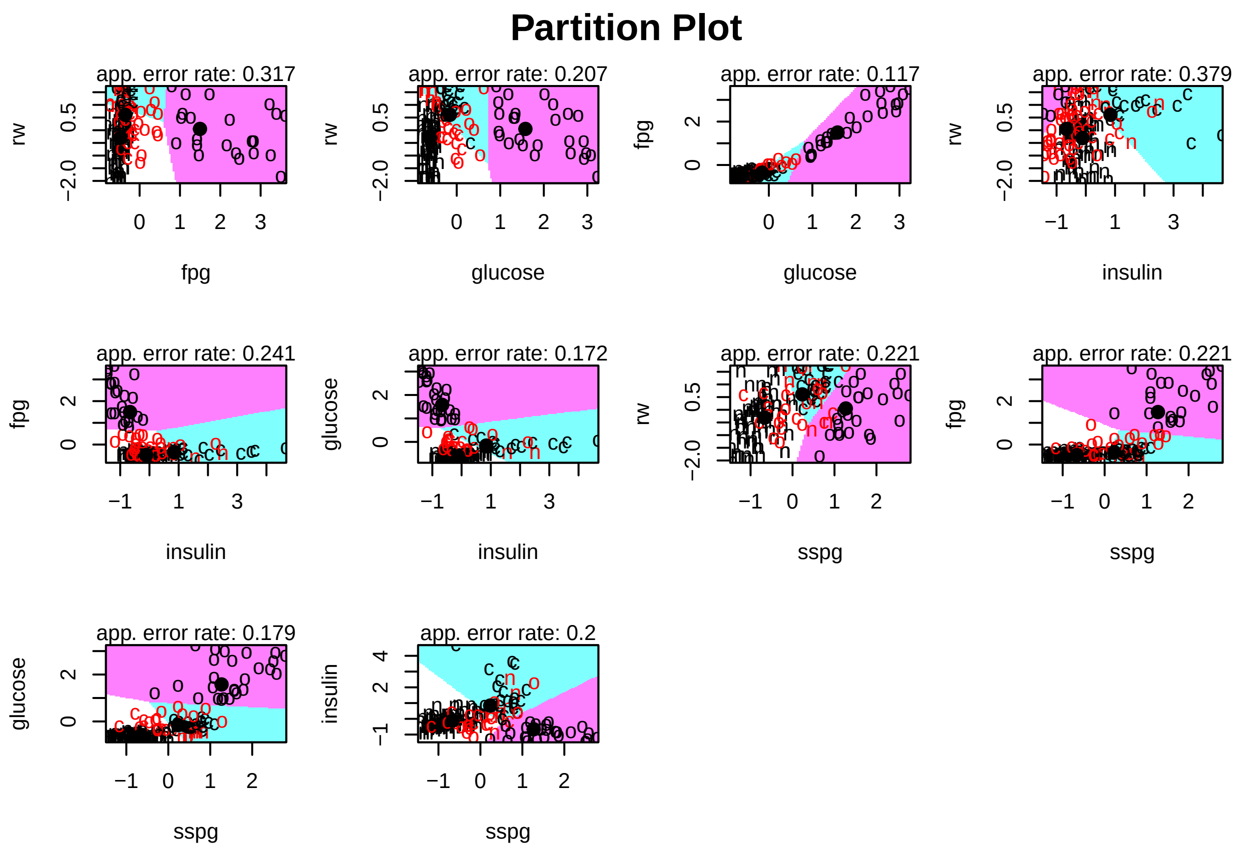
- Stellen Sie die Ergebnisse grafisch dar (Scatter Plot).

```r
# Load the data
diabetes <- read.csv("diabetes_RM.csv", header = TRUE, sep = ",")
groups <- c("normal", "chemical", "overt")



# Scale the selected columns
diabetes_scaled_cols <- scale(diabetes[, c("rw", "fpg", "glucose", "insulin", "sspg")])

# Create a new data frame from the scaled columns
diabetes_scaled_df <- as.data.frame(diabetes_scaled_cols)

# Set the row names of the scaled data frame to match the row names of the original data frame
rownames(diabetes_scaled_df) <- rownames(diabetes)

# Bind the scaled columns with the unscaled columns
diabetes_scaled <- cbind(diabetes_scaled_df, group = diabetes$group)


#-------------------------------------------------------------------------------------------
# Dendrogram

# Visualize the tree
fviz_dend(hkmeans_result, cex = 0.6, palette = "jco",
          rect = TRUE, rect_border = "jco", rect_fill = TRUE)
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## i The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <https://github.com/kassambara/factoextra/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```r
#------------------------------------------------------------------------------------
# Perform LDA

diabetes_scaled$group <- as.factor(diabetes_scaled$group)

# Partition plot
partimat(group ~ ., data = diabetes_scaled, method = "lda")
```



```r
ml <- lda(group ~ ., data = diabetes_scaled)

print(ml)
```

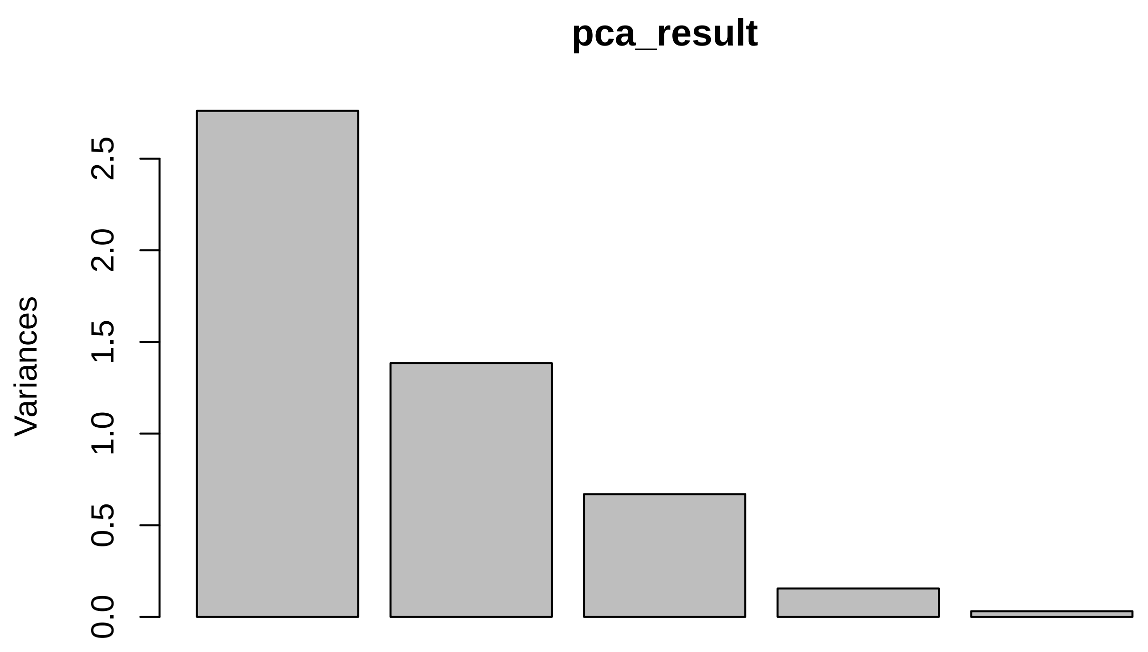
```
## Call:
## lda(group ~ ., data = diabetes_scaled)
##
## Prior probabilities of groups:
##  chemical    normal     overt
## 0.2482759 0.5241379 0.2275862
##
## Group means:
##                   rw        fpg    glucose    insulin       sspg
## chemical  0.60759742 -0.3547709 -0.1567099  0.8424577  0.2335693
## normal   -0.31008189 -0.4818051 -0.6109468 -0.1114027 -0.6621427
## overt     0.05129444  1.4966346  1.5779852 -0.6624810  1.2701317
##
## Coefficients of linear discriminants:
##                 LD1        LD2
## rw       0.17607468  0.4890445
## fpg     -2.15118075 -2.3419829
## glucose  3.98609901  2.2478209
## insulin -0.01236254  0.7465840
## sspg     0.44990449 -0.1202453
##
```

```
## Proportion of trace:
##    LD1    LD2
## 0.8812 0.1188
```

```
#------------------------------------------------------------------------------------------
# Perform PCA
# Using subset
diabetes_scaled <- subset(diabetes_scaled, select = -group)
pca_result <- prcomp(diabetes_scaled, center = TRUE, scale. = TRUE)

# Print summary of the PCA result
summary(pca_result)
```

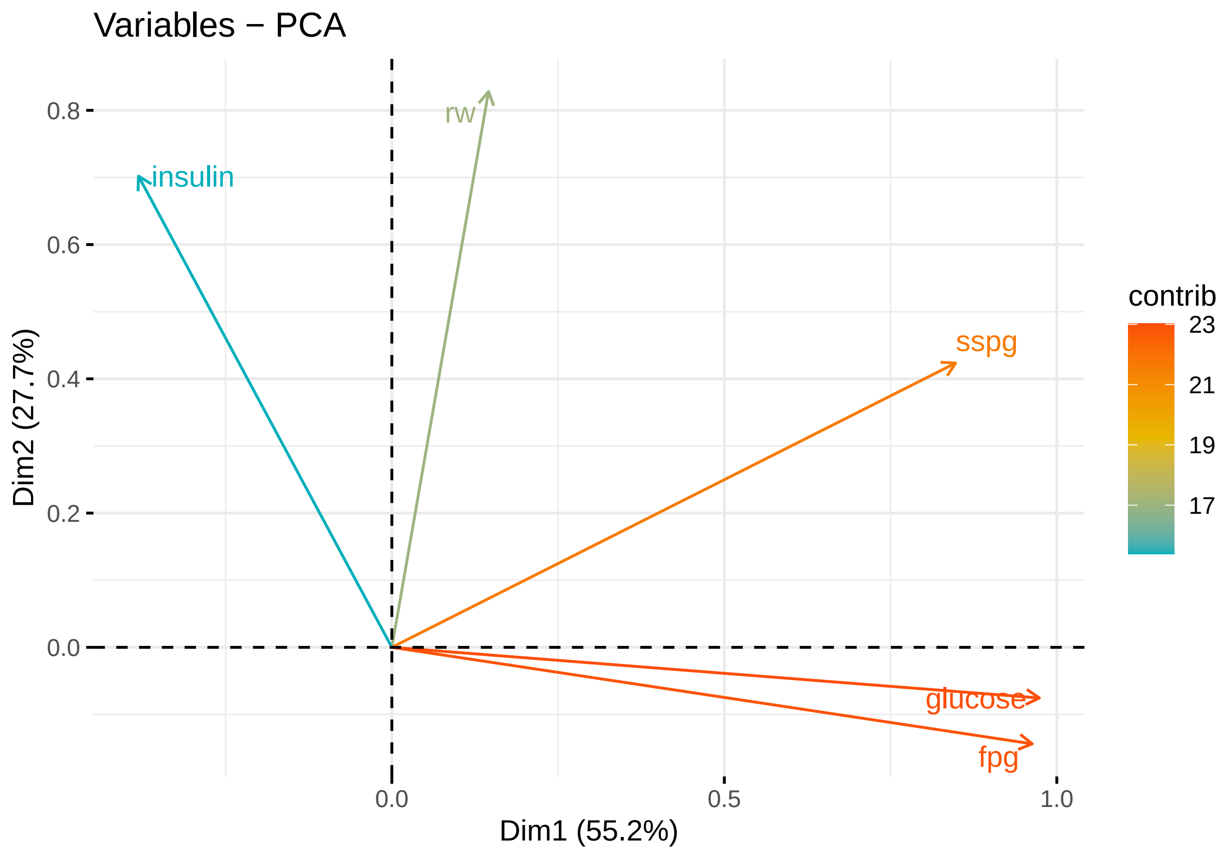
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5
## Standard deviation     1.6615 1.1766 0.8181 0.39341 0.17589
## Proportion of Variance 0.5521 0.2769 0.1339 0.03095 0.00619
## Cumulative Proportion  0.5521 0.8290 0.9629 0.99381 1.00000
```

```
# Plot the variance explained by each principal component
plot(pca_result)
```

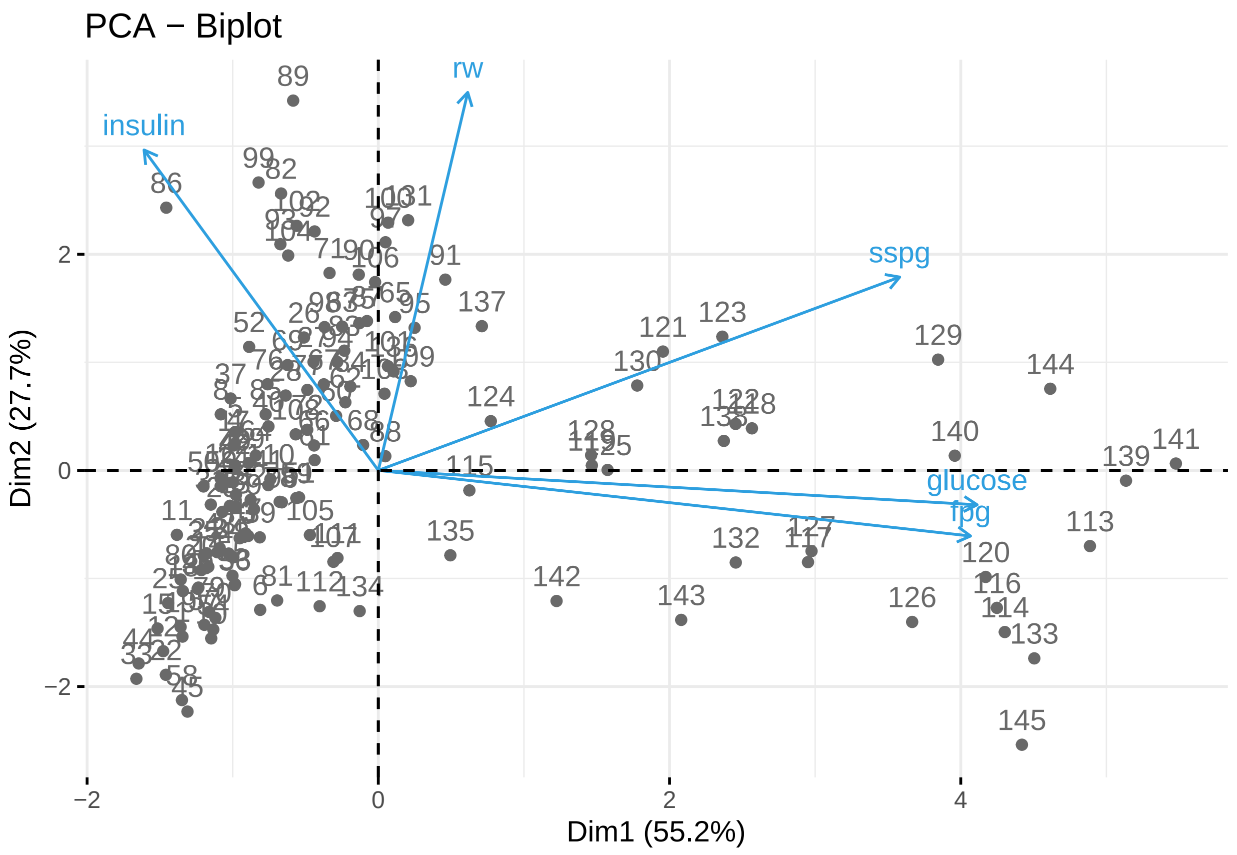


```
# Perform PCA
pca_result <- prcomp(diabetes_scaled[sapply(diabetes_scaled, is.numeric)])
```

```
# Plot the correlation circle
fviz_pca_var(pca_result, col.var="contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel =
```



```
# Plot the biplot
fviz_pca_biplot(pca_result, col.var="#2E9FDF", col.ind="#696969")
```

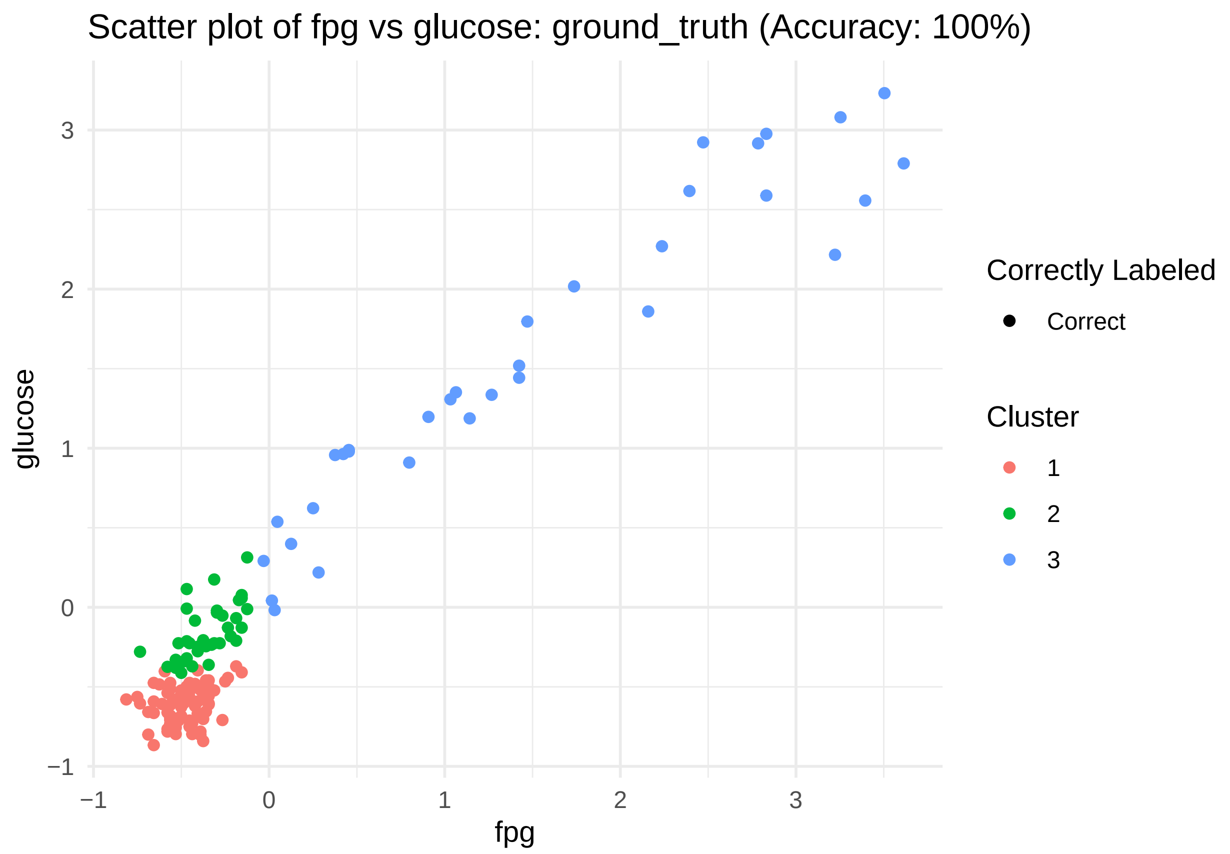


Die LDA zeigt, dass fpg und glucose sowohl zu LD1 und LD2 am stärksten beitragen (d.h. die höchsten Absolutbeträge aufweisen). Die PCA bestätigt dies, da fpg und glucose die stärksten Beiträge zu PC1 und PC2 liefern. Deshalb werden die 2-D Scatter plots *direkt* für fpg und glucose erstellt.

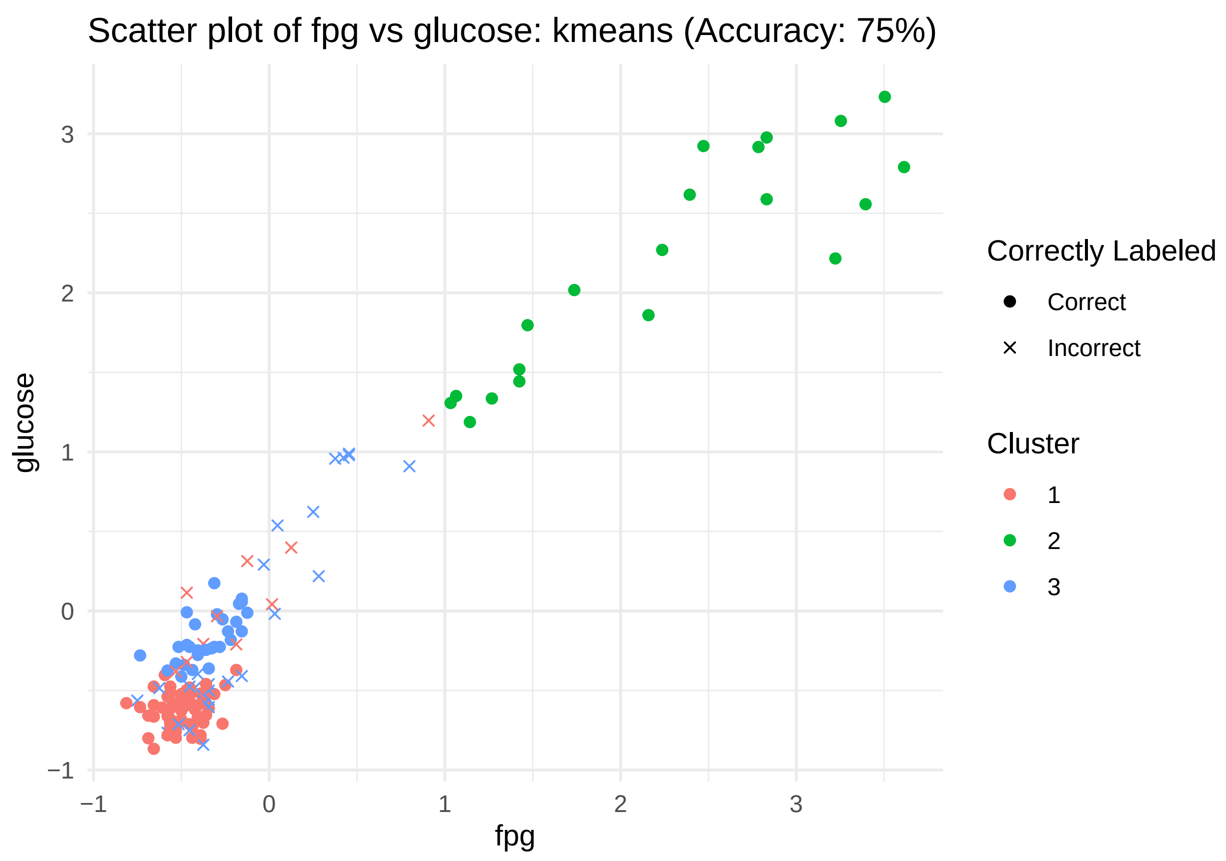
```
load_source()

# Load the data
diabetes <- read.csv("diabetes_RM.csv", header = TRUE, sep = ",")
groups <- c("normal", "chemical", "overt")



# Scale the selected columns
diabetes_scaled_cols <- scale(diabetes[, c("rw", "fpg", "glucose", "insulin", "sspg")], center = TRUE)

# Create a new data frame from the scaled columns
diabetes_scaled_df <- as.data.frame(diabetes_scaled_cols)

# Set the row names of the scaled data frame to match the row names of the original data frame
rownames(diabetes_scaled_df) <- rownames(diabetes)

# Bind the scaled columns with the unscaled columns
diabetes_scaled <- cbind(diabetes_scaled_df, group = diabetes$group)

ignore <- TRUE
# Loop over the list and create a scatter plot for each set of cluster labels
```

```r
for (name in names(list_of_cluster_vectors)) {
  if(name == "hclust") {
    ignore <- FALSE
  }
  plot(create_scatter_plot_with_accuracy(diabetes_scaled,
      list_of_cluster_vectors[[name]], name, accuracy_results[[name]],
      groups,
      ignore))

  ignore <- TRUE
}
```

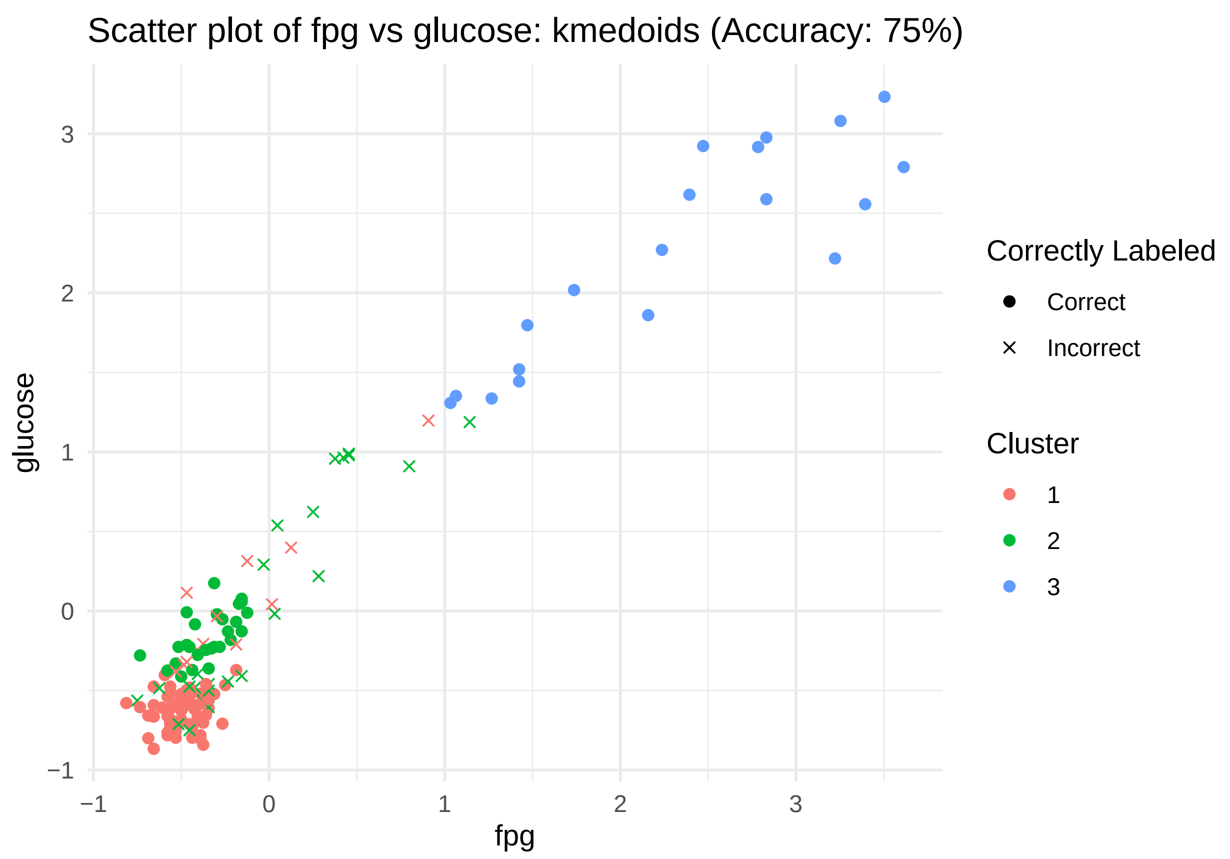
```
##            [,1]     [,2]     [,3]
## [1,] 0.000000 1.666359 3.587982
## [2,] 1.666359 0.000000 3.175790
## [3,] 3.587982 3.175790 0.000000
## [1] 1 2 3
```



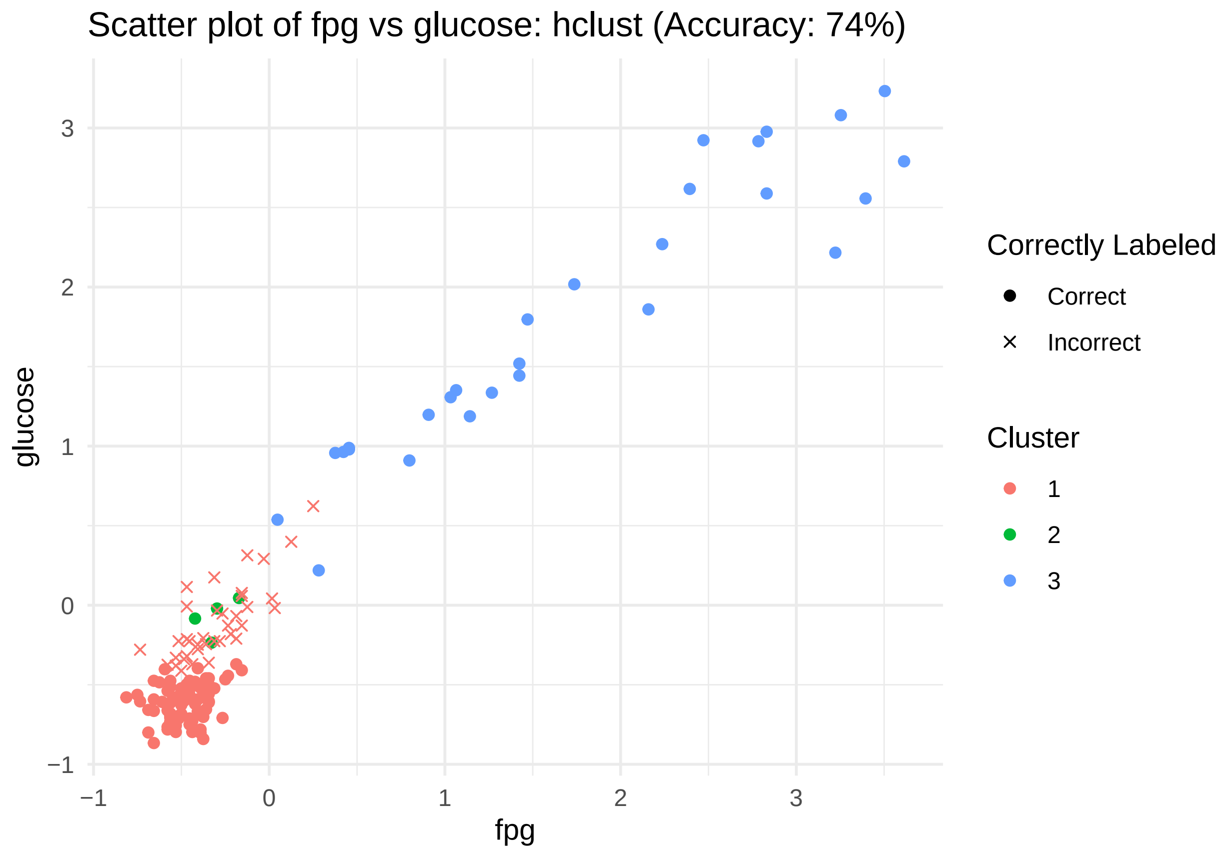
```
##             [,1]     [,2]     [,3]
## [1,] 0.3621507 4.658219 1.804704
## [2,] 1.9576353 4.285744 0.323201
## [3,] 3.6043677 1.127521 3.027601
## [1] 1 3 2
```



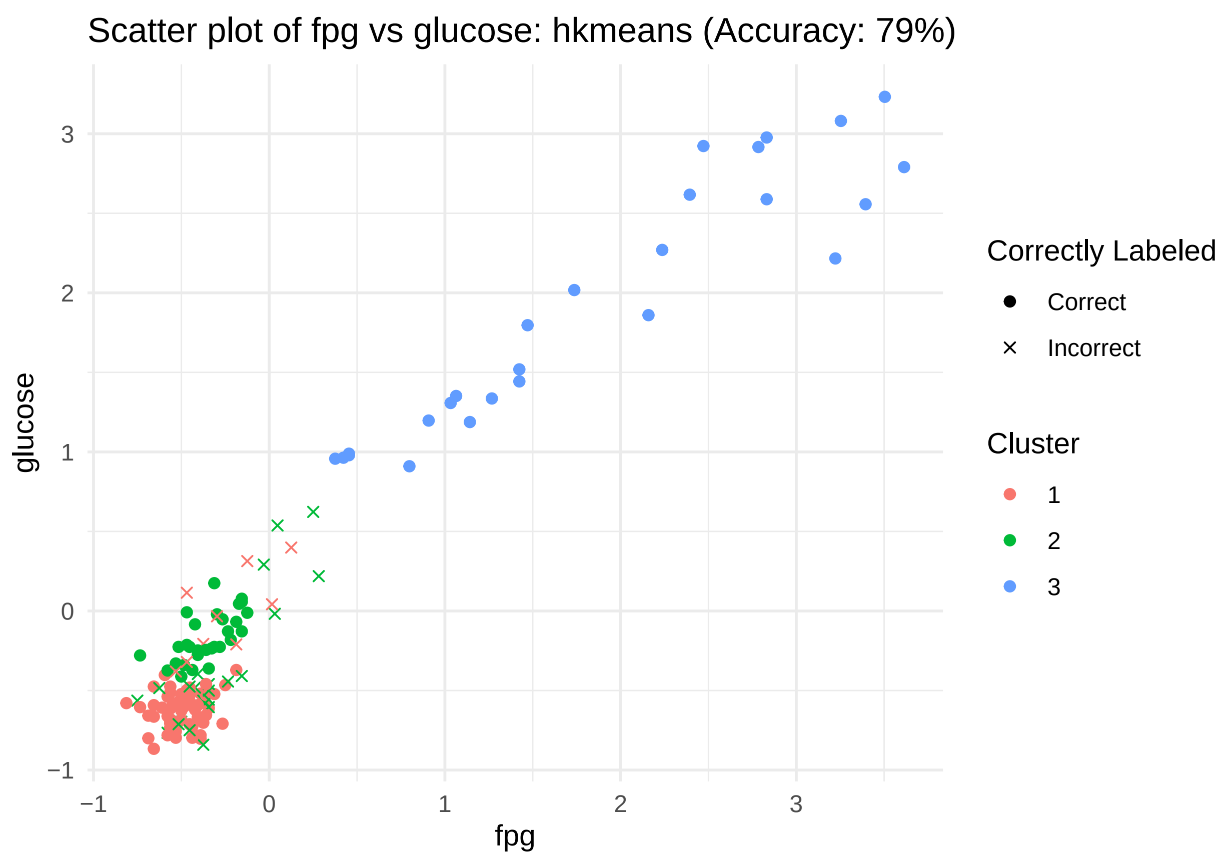
```
##            [,1]      [,2]     [,3]
## [1,] 0.2750971 1.9263863 4.731498
## [2,] 1.8825475 0.4080579 4.383376
## [3,] 3.5838033 2.9547495 1.223532
## [1] 1 2 3
```



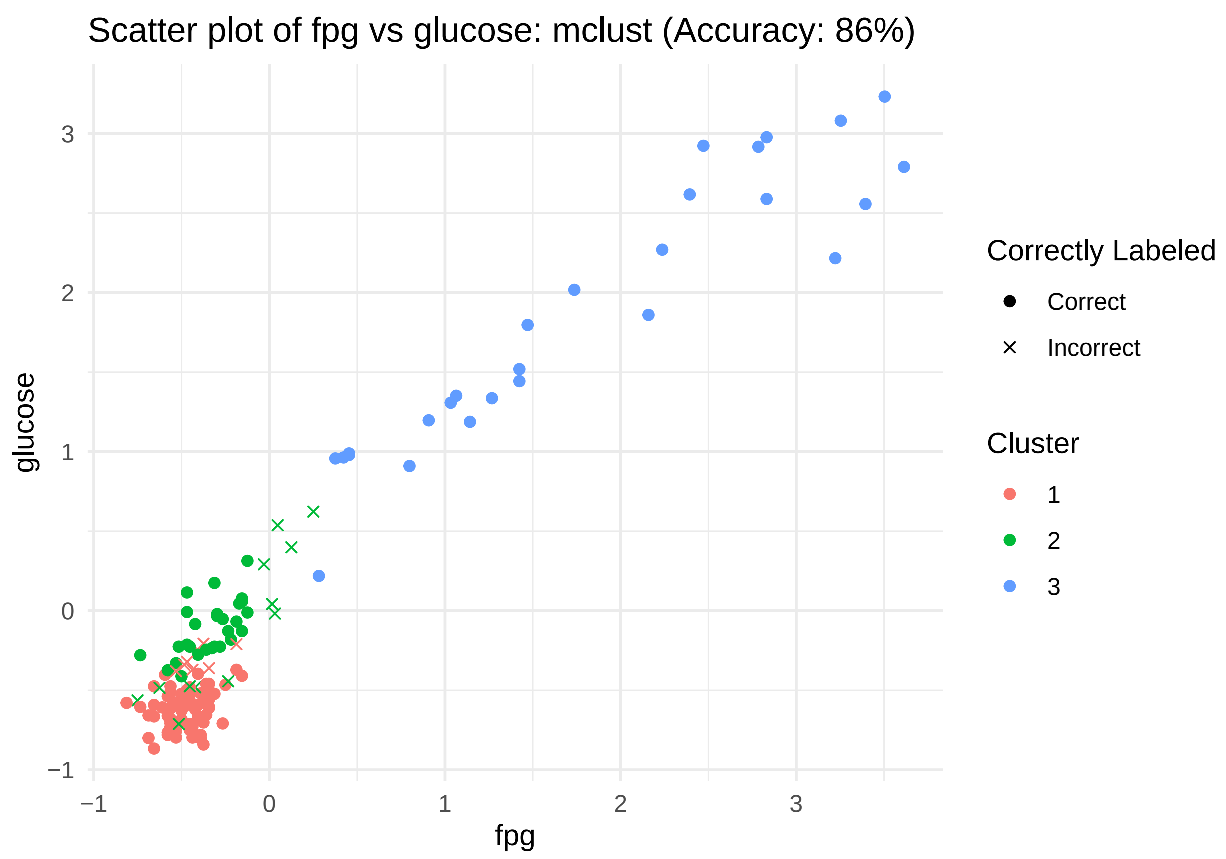
```
##           [,1]     [,2]      [,3]
## [1,] 0.4881984 4.048269 3.9939866
## [2,] 1.1937538 2.880662 3.6054722
## [3,] 3.3203439 5.118442 0.4326986
## [1] 1 2 3
##               rw         fpg      glucose       insulin        sspg group
## 1   -1.29461875 -0.65674862 -0.591933371 -0.5136408814 -1.21858967     1
## 2   -0.21132276 -0.39083447 -0.803322605 -0.5715231391 -1.02053227     1
## 3   -0.28870105 -0.26569840 -0.708670709 -0.3565318964 -0.74702441     1
## 4    0.48508180 -0.50032853 -0.591933371  0.1065261648 -0.71873050     1
## 5    0.17556866 -0.50032853 -0.696050456  0.4455508168 -0.38863482     1
## 6   -1.68151017 -0.56289656 -0.513056792 -0.2407673811 -0.18114610     1
## 7   -0.52083590 -0.34390844 -0.610863750  0.2884418317 -0.61498614     1
## 8    0.94935151 -0.57853857 -0.765461846 -0.0009694565 -0.74702441     1
## 9    0.09819037 -0.39083447 -0.519366918 -0.3648007903 -0.81304355     1
## 10  -1.52675360 -0.39083447 -0.781237162 -0.4557586238 -0.85076877     1
## 11  -0.59821419 -0.48468652 -0.601398561  0.2884418317 -1.23745229     1
## 12  -1.91364503 -0.54725456 -0.749686530 -0.0671206081 -1.11484532     1
## 13  -0.13394448 -0.68803264 -0.800167541 -0.4144141540 -0.39806612     1
## 14  -1.06248389 -0.50032853 -0.544607424  0.1147950588 -0.86020008     1
## 15  -1.83626674 -0.56289656 -0.730756151  0.1809462104 -1.09598271     1
## 16   0.02081209 -0.65674862 -0.475196034  0.1313328467 -0.77531833     1
## 17   0.94935151 -0.50032853 -0.566692866 -0.2821118508 -1.02053227     1
## 18  -0.98510561 -0.35955045 -0.582468182 -0.0092383505 -1.38835317     1
## 19  -1.13986218 -0.57853857 -0.781237162 -0.5797920330 -1.17143315     1
## 20  -0.36607933 -0.50032853 -0.626639066 -0.5219097754 -1.26574620     1
## 21  -0.21132276 -0.50032853 -0.522521981 -0.4144141540 -1.29404012      1
## 22  -1.83626674 -0.53161255 -0.755996657 -0.4309519419 -1.26574620      1
## 23  -0.21132276 -0.42211849 -0.620328940 -0.0175072444 -0.87906269      1
## 24  -0.05656619 -0.50032853 -0.683430204  0.0486439072 -0.56782962      1
## 25  -1.99102331 -0.46904451 -0.497281476  0.7680376808 -1.03939488      1
## 26   1.02672979 -0.75060067 -0.563537803  0.3463240894  0.47904526      1
## 27   1.72313435 -0.37519246 -0.563537803 -0.3399941085 -0.24716524      1
## 28   1.18148636 -0.34390844 -0.604553624 -0.1167339718 -0.41692873      1
## 29   0.17556866 -0.56289656 -0.689740330 -0.0588517142 -0.36977221      1
## 30  -1.52675360 -0.37519246 -0.702360583  0.2967107257 -0.80361225      1
## 31   0.17556866 -0.81316871 -0.579313119 -0.4309519419 -0.88849399      1
## 32   0.17556866 -0.35955045 -0.655034635 -0.3565318964 -0.74702441      1
## 33  -2.06840159 -0.73495866 -0.604553624 -0.1415406537 -1.43550969      1
## 34  -1.68151017 -0.50032853 -0.601398561  0.6357353776 -0.18114610      1
## 35  -0.67559247 -0.57853857 -0.538297297 -0.1001961839 -1.00166966      1
## 36  -0.75297076 -0.35955045 -0.528832108 -0.4309519419 -0.98280704      1
## 37   1.49099950 -0.34390844 -0.557227676 -0.0340450323 -1.22802098      1
## 38  -0.98510561 -0.68803264 -0.658189698  0.4538197107 -0.08683305      1
## 39  -0.05656619 -0.25005639 -0.465730844 -0.4805653056 -0.98280704      1
## 40   0.17556866 -0.37519246 -0.841183363  0.2967107257  0.01691131      1
## 41   0.17556866 -0.31262442 -0.522521981 -0.1746162295 -0.63384875      1
## 42  -0.67559247 -0.50032853 -0.579313119  0.7928443627 -0.22830263      1
## 43   0.02081209 -0.43776049 -0.797012478 -0.7617076999 -1.06768879      1
## 44  -1.52675360 -0.65674862 -0.866423868 -0.5384475633 -1.46380361      1
## 45  -1.83626674 -0.45340250 -0.711825772 -0.9353544729 -1.34119664      1
## 46  -0.52083590 -0.56289656 -0.680275140 -0.6624809725 -1.20915837      1
## 47  -0.21132276 -0.57853857 -0.661344761 -0.5632542451 -0.58669223      1
## 48  -0.21132276 -0.40647648 -0.591933371 -0.6128676088 -1.04882618      1
## 49   0.40770351 -0.53161255 -0.797012478 -0.2407673811 -0.58669223      1
## 50  -0.83034904 -0.54725456 -0.579313119  0.8755333022 -0.53010440      1
## 51  -0.83034904 -0.43776049 -0.727601088  0.1147950588  0.46018265      1
## 52   1.49099950 -0.45340250 -0.749686530  0.2801729378 -0.49237917      1
## 53  -1.13986218 -0.56289656 -0.708670709 -0.3482630024 -0.43579134      1
## 54  -1.21724046 -0.56289656 -0.614018814 -0.6376742907 -0.95451313      1
## 55  -0.90772733 -0.40647648 -0.667654888 -0.2903807448 -0.70929919      1
## 56   0.25294694 -0.56289656 -0.696050456 -0.2324984871 -0.83190616      1
## 57  -0.75297076 -0.51597054 -0.696050456 -0.9353544729 -1.24688359      1
## 58  -1.75888846 -0.60982259 -0.607708687 -0.8692033213 -1.34119664      1
## 59   0.09819037 -0.37519246 -0.207015663 -0.2903807448 -0.58669223      2
## 60   1.10410808 -0.34390844 -0.459420718 -0.5301786693 -0.07740175      1
## 61   0.87197322 -0.18748835 -0.371078948 -0.5715231391 -0.62441745      1
## 62   0.33032523 -0.53161255 -0.330063127  0.1809462104  0.56392701      2
## 63   1.64575607 -0.34390844 -0.361613759  0.1230639527  0.09236175      2
## 64   0.63983837 -0.65674862 -0.664499825 -0.4557586238 -0.45465395      1
## 65   1.72313435 -0.51597054 -0.225946042 -0.1994229113  0.68653397      2
## 66   0.56246008 -0.48468652 -0.339528317 -0.3151874266 -0.16228349      2
## 67   1.56837778 -0.40647648 -0.396319454 -0.4640275177 -0.29432177      1
## 68   0.25294694 -0.42211849 -0.481506160 -0.4061452601  0.60165223      1
## 69  -0.52083590 -0.62546460 -0.484661223  1.5618515000  0.83743486      1
## 70  -1.29461875 -0.59418058 -0.402629580 -0.3317252145 -0.98280704      1
## 71   0.94935151 -0.50032853 -0.412094770  1.3055157876  0.80914094      2
## 72   0.40770351 -0.34390844 -0.500436539  0.0486439072 -0.03967653      1
## 73  -0.05656619 -0.56289656 -0.475196034 -0.5880609270 -0.93565052      1
## 74  -0.13394448 -0.45340250 -0.528832108  0.0734505890 -0.73759311      1
## 75   0.94935151 -0.23441438 -0.443645402  0.6688109534  0.65824006      1
## 76   0.71721665 -0.15620434 -0.408939707  0.7845754687 -0.61498614      1
## 77   0.79459494 -0.43776049 -0.371078948  0.2222906801 -0.06797044      2
## 78  -0.21132276 -0.45340250 -0.566692866 -0.2490362750 -0.23773393      1
## 79  -1.83626674 -0.45340250 -0.481506160  0.2884418317 -0.76588703      1
## 80  -1.06248389 -0.50032853 -0.591933371  0.1065261648 -1.18086445      1
## 81  -0.67559247 -0.35955045 -0.459420718 -0.9105477910 -0.71873050      1
## 82   1.02672979 -0.45340250 -0.475196034  2.5127743043  0.70539659      1
## 83   1.64575607 -0.57853857 -0.374234012 -0.3565318964  0.18667480      2
## 84   1.56837778 -0.51597054 -0.711825772 -0.9353544729  0.33757568      1
## 85   0.63983837 -0.40647648 -0.248031484  0.4207441349 -0.69043658      2
## 86  -0.21132276 -0.17184634  0.045389392  4.6461489434 -0.58669223      2
## 87   0.63983837 -0.23441438 -0.128139083  1.1070623328  0.64880875      2
## 88   0.02081209 -0.12492032 -0.011401746  0.0155683314  0.25269394      2
## 89   1.41362121 -0.32826643 -0.235411232  3.4802348964  0.81857225      2
## 90   1.56837778 -0.21877237 -0.181775157  0.9168777719  0.33757568      2
## 91   1.72313435 -0.15620434  0.076940024  0.3793996652  0.86572877      2
## 92   0.79459494 -0.26569840 -0.052417567  2.4300853648  0.46018265      2
## 93  -0.52083590 -0.29698242 -0.020866935  3.6042683057  0.75255311      2
## 94   0.40770351 -0.35955045 -0.244876421  0.8341888324  0.44132004      2
## 95   0.87197322 -0.31262442  0.174746982  0.6605420595  0.79027833      2
## 96   0.56246008 -0.18748835 -0.210170726 -0.5136408814 -1.17143315      2
## 97   1.72313435 -0.31262442 -0.225946042  0.9168777719  0.82800355      2
## 98   0.56246008 -0.40647648 -0.276427053  1.1566756965  0.47904526      2
## 99   0.94935151 -0.42211849 -0.083968199  3.1246724566  0.20553741      2
## 100  1.10410808 -0.15620434 -0.128139083  1.8347250004  1.09208010      2
## 101 -0.13394448 -0.18748835 -0.068192883  1.1484068025  0.96004183      2
## 102  1.18148636 -0.46904451 -0.213325789  2.0414473491  0.39416351      2
## 103  0.71721665 -0.28134041 -0.225946042 -0.0505828202  0.51677048      2
## 104  0.94935151 -0.73495866 -0.279582116  1.7024226972  0.54506440      2
## 105 -0.28870105 -0.46904451 -0.320597938 -0.6376742907 -0.25659654      2
## 106  1.10410808 -0.46904451 -0.008246682  1.0491800751  0.78084703      2
## 107 -0.75297076 -0.46904451  0.114800782 -0.4474897298 -0.27545916      2
## 108 -0.36607933 -0.45340250 -0.225946042  0.8176510445  0.09236175      2
## 109  1.41362121 -0.15620434  0.058009644 -0.3896074722  0.13008697      2
## 110 -0.28870105 -0.53161255 -0.380544138  0.2140217862 -0.26602785      2
## 111 -0.52083590 -0.12492032  0.313569762 -0.2573051690 -0.79418094      2
## 112 -1.13986218 -0.29698242 -0.033487188 -0.5467164572 -0.46408526      2
## 113 -0.44345762  2.78449332  2.916496888 -1.3074547006  2.55393240      3
## 114 -0.90772733  2.83141934  2.976443089 -1.3487991704  1.34672534      3
## 115 -0.98510561  0.04714178  0.537579248  0.3793996652  0.89402269      3
## 116 -1.13986218  2.47165314  2.922807015 -1.0924634579  1.86544712      3
## 117 -0.98510561  1.47056458  1.796449458 -0.8692033213  1.82772190      3
## 118  0.63983837  1.06387235  1.351585550 -0.8195899576  1.78999668      3
## 119  0.63983837  0.45383401  0.979288094 -0.9105477910  0.71482789      3
## 120 -0.44345762  2.83141934  2.588370317 -1.1916901853  1.52592014      3
## 121  1.72313435  0.79795820  0.909876704 -0.6955565483  1.27127490      3
## 122  0.48508180  1.26721846  1.335810234 -0.3978763661  1.57307666      3
## 123  1.41362121  1.14208239  1.187522264 -0.2159606992  1.62966449      3
## 124  0.79459494  0.28177191  0.218917867 -0.4557586238  0.60165223      3
## 125 -0.21132276  0.45383401  0.988753284 -0.3399941085  1.31843142      3
## 126 -0.90772733  2.39344310  2.616765886 -1.1668835035  1.09208010      3
## 127 -0.59821419  2.15881297  1.859550722 -0.5632542451  1.09208010      3
## 128 -0.05656619  0.42254999  0.963512778 -0.2242295932  1.18639315      3
## 129  1.41362121  1.73647873  2.017303881 -0.9353544729  2.58222632      3
## 130  1.10410808  0.37562396  0.957202652 -0.6872876544  1.45990100      3
## 131  0.71721665  0.03149977 -0.017711872  2.2647074858  1.28070620      3
## 132 -0.36607933  1.42363855  1.443082382 -1.1916901853  1.06378618      3
## 133 -0.98510561  3.25375358  3.080560174 -1.4314881099  1.12037401      3
## 134 -1.29461875  0.01585776  0.042234329 -0.4640275177 -0.30375307      3
## 135  0.02081209  0.12535182  0.398756468 -1.1751523974 -0.16228349      3
## 136  0.25294694 -0.03106827  0.291484320  1.0574489691  0.33757568      3
## 137  1.64575607  0.25048789  0.622765954  0.2719040438  0.23383133      3
## 138  0.48508180  1.03258833  1.307414665 -0.7120943362  1.57307666      3
## 139  0.63983837  3.39453166  2.556819686 -1.4562947917  2.50677588      3
## 140  0.40770351  2.23702301  2.269708936 -0.8526655334  2.15781759      3
## 141  0.56246008  3.61351978  2.790294361 -1.1999590793  2.78971503      3
## 142 -0.52083590  0.90745226  1.196987454 -0.9022788971 -0.32261568      3
## 143 -0.59821419  1.42363855  1.518803898 -1.2991858067  0.23383133      3
## 144  1.02672979  3.22246956  2.216072862 -0.5136408814  2.43132544      3
## 145 -1.83626674  3.50402572  3.232003207 -1.4149503220  0.64880875      3
##     cluster correctly_labeled
## 1         1           Correct
## 2         1           Correct
## 3         1           Correct
## 4         1           Correct
## 5         1           Correct
## 6         1           Correct
## 7         1           Correct
## 8         1           Correct
## 9         1           Correct
## 10        1           Correct
## 11        1           Correct
## 12        1           Correct
## 13        1           Correct
## 14        1           Correct
## 15        1           Correct
## 16        1           Correct
## 17        1           Correct
## 18        1           Correct
## 19        1           Correct
## 20        1           Correct
## 21        1           Correct
## 22        1           Correct
## 23        1           Correct
## 24        1           Correct
## 25        1           Correct
## 26        1           Correct
## 27        1           Correct
## 28        1           Correct
## 29        1           Correct
## 30        1           Correct
## 31        1           Correct
## 32        1           Correct
## 33        1           Correct
## 34        1           Correct
## 35        1           Correct
## 36        1           Correct
## 37         1           Correct
## 38         1           Correct
## 39         1           Correct
## 40         1           Correct
## 41         1           Correct
## 42         1           Correct
## 43         1           Correct
## 44         1           Correct
## 45         1           Correct
## 46         1           Correct
## 47         1           Correct
## 48         1           Correct
## 49         1           Correct
## 50         1           Correct
## 51         1           Correct
## 52         1           Correct
## 53         1           Correct
## 54         1           Correct
## 55         1           Correct
## 56         1           Correct
## 57         1           Correct
## 58         1           Correct
## 59         1         Incorrect
## 60         1           Correct
## 61         1           Correct
## 62         1         Incorrect
## 63         1         Incorrect
## 64         1           Correct
## 65         1         Incorrect
## 66         1         Incorrect
## 67         1           Correct
## 68         1           Correct
## 69         1           Correct
## 70         1           Correct
## 71         1         Incorrect
## 72         1           Correct
## 73         1           Correct
## 74         1           Correct
## 75         1           Correct
## 76         1           Correct
## 77         1         Incorrect
## 78         1           Correct
## 79         1           Correct
## 80         1           Correct
## 81         1           Correct
## 82         1           Correct
## 83         1         Incorrect
## 84         1           Correct
## 85         1         Incorrect
## 86         2           Correct
## 87         1         Incorrect
## 88         1         Incorrect
## 89         2           Correct
## 90         1         Incorrect
## 91        1          Incorrect
## 92        1          Incorrect
## 93        2            Correct
## 94        1          Incorrect
## 95        1          Incorrect
## 96        1          Incorrect
## 97        1          Incorrect
## 98        1          Incorrect
## 99        2            Correct
## 100       1          Incorrect
## 101       1          Incorrect
## 102       1          Incorrect
## 103       1          Incorrect
## 104       1          Incorrect
## 105       1          Incorrect
## 106       1          Incorrect
## 107       1          Incorrect
## 108       1          Incorrect
## 109       1          Incorrect
## 110       1          Incorrect
## 111       1          Incorrect
## 112       1          Incorrect
## 113       3            Correct
## 114       3            Correct
## 115       3            Correct
## 116       3            Correct
## 117       3            Correct
## 118       3            Correct
## 119       3            Correct
## 120       3            Correct
## 121       3            Correct
## 122       3            Correct
## 123       3            Correct
## 124       3            Correct
## 125       3            Correct
## 126       3            Correct
## 127       3            Correct
## 128       3            Correct
## 129       3            Correct
## 130       3            Correct
## 131       1          Incorrect
## 132       3            Correct
## 133       3            Correct
## 134       1          Incorrect
## 135       1          Incorrect
## 136       1          Incorrect
## 137       1          Incorrect
## 138       3            Correct
## 139       3            Correct
## 140       3            Correct
## 141       3            Correct
## 142       3            Correct
## 143       3            Correct
## 144       3            Correct
## 145       3            Correct
```



```
##           [,1]      [,2]      [,3]
## [1,] 0.371128 1.7749877 4.1700617
## [2,] 1.977689 0.2475693 3.7832967
## [3,] 3.637292 3.2219678 0.6135777
## [1] 1 2 3
```



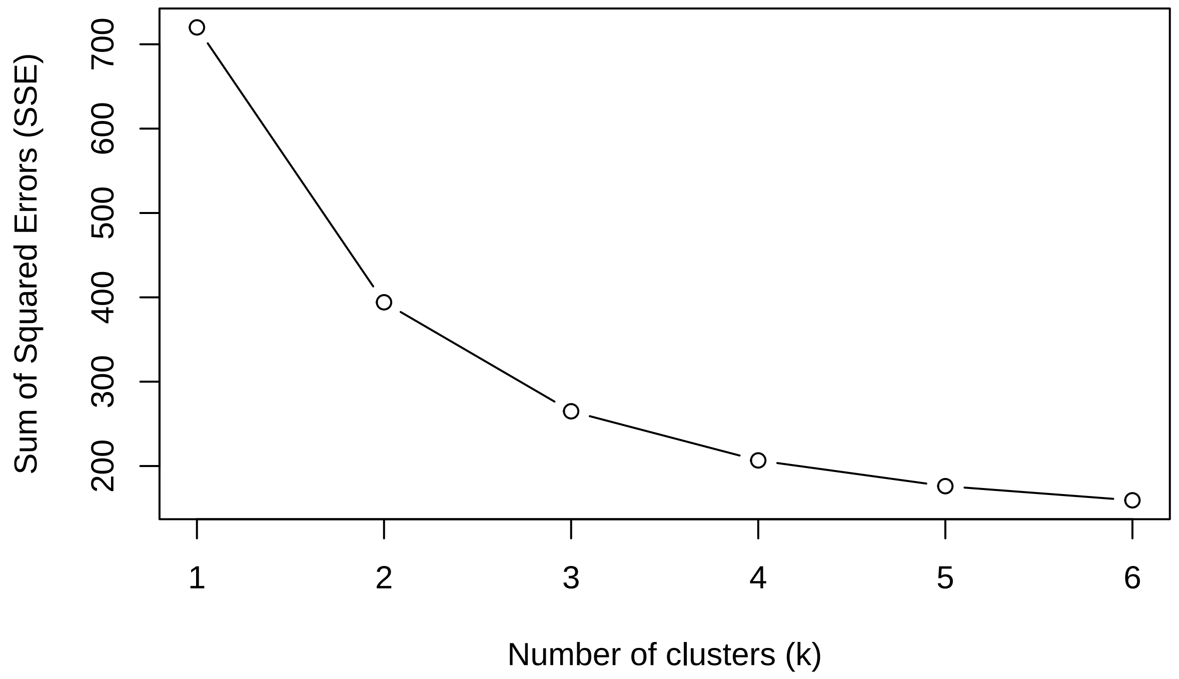
```
##             [,1]      [,2]      [,3]
## [1,] 0.09247276 1.8287537 4.0804257
## [2,] 1.72876397 0.2333928 3.6906429
## [3,] 3.59758685 3.0974983 0.5223698
## [1] 1 2 3
```



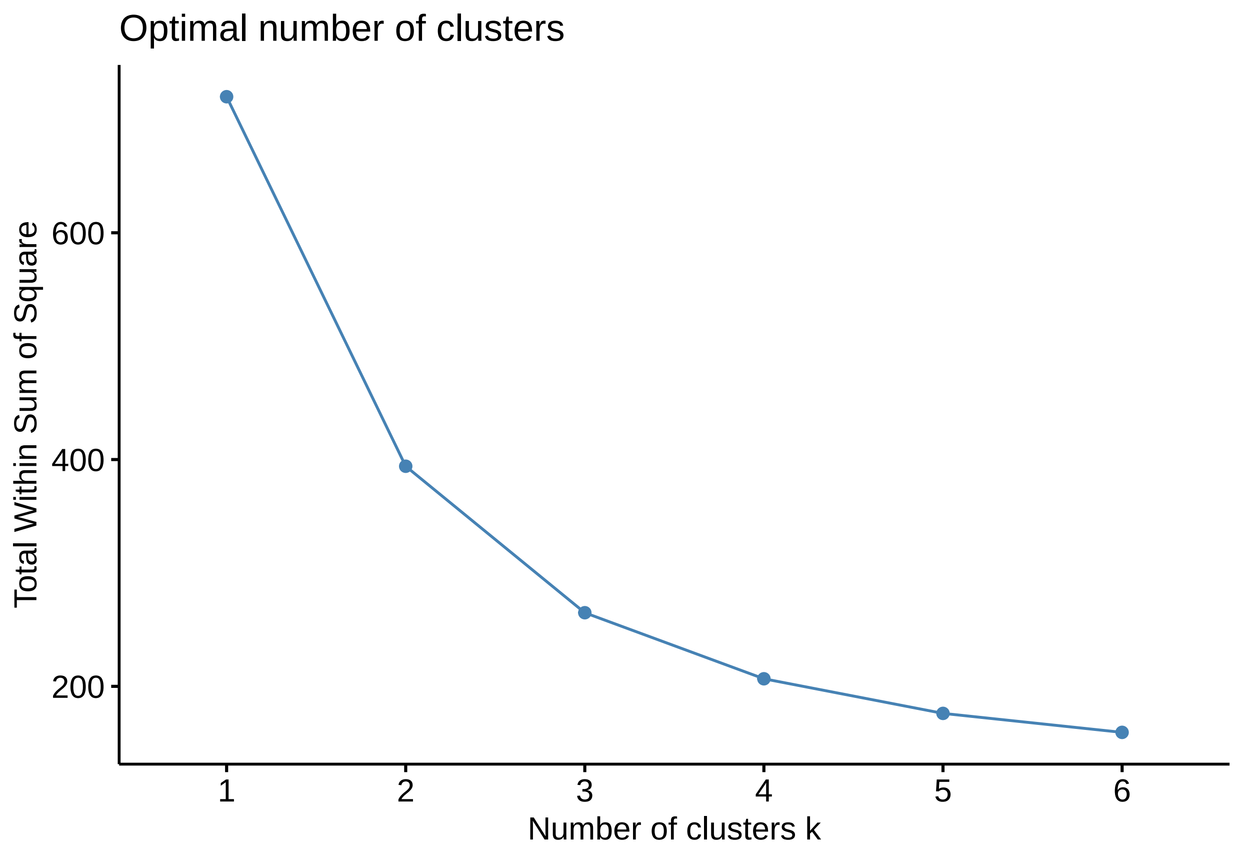
- Finden Sie für k-means die optimale Anzahl an Cluster, und beurteilen Sie ob sich die Ground Truth Clusterstruktur reproduzieren lässt.

```
load_source()

# Load the data
diabetes <- read.csv("diabetes_RM.csv", header = TRUE, sep = ",")
groups <- c("normal", "chemical", "overt")



# Scale the selected columns
diabetes_scaled_cols <- scale(diabetes[, c("rw", "fpg", "glucose", "insulin", "sspg")])

# Create a new data frame from the scaled columns
diabetes_scaled_df <- as.data.frame(diabetes_scaled_cols)

# Set the row names of the scaled data frame to match the row names of the original data frame
rownames(diabetes_scaled_df) <- rownames(diabetes)

# Bind the scaled columns with the unscaled columns
diabetes_scaled <- cbind(diabetes_scaled_df, group = diabetes$group)


#-----------------------------------------------------------------------------------------
```

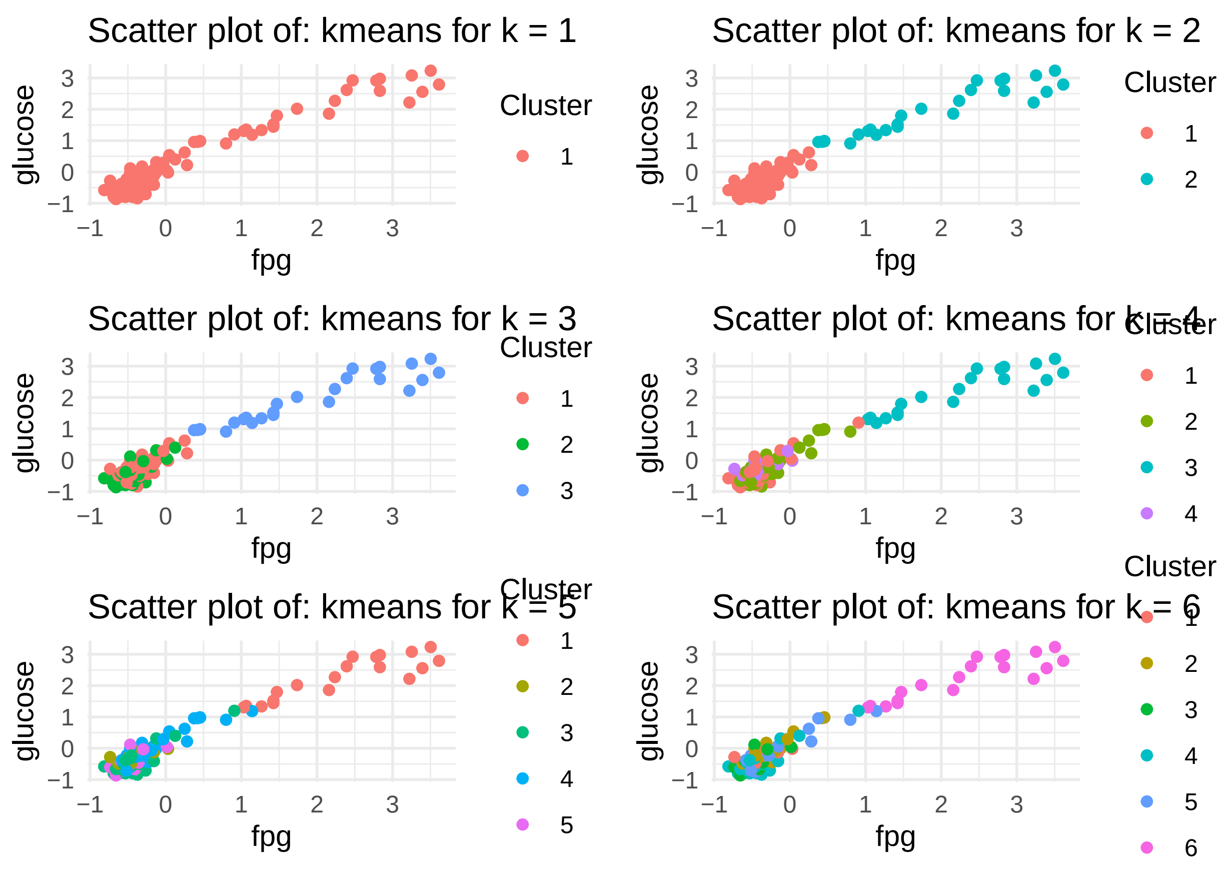
```r
n = 6
set.seed(123)
sse <- numeric(n)
list_of_cluster_vectors <- list()
list_of_cluster_results <- list()

for(k in 1:n) {
  kmeans_result <- kmeans(diabetes_scaled_cols, centers = k)
  sse[k] <- kmeans_result$tot.withinss

  list_of_cluster_vectors[[k]] <- kmeans_result$cluster
  list_of_cluster_results[[k]] <- kmeans_result
}

#-------------------------------------------------------------------------------------------
# Plot the SSE/WSS

# Determine the optimal number of clusters using SSE
plot(1:n, sse, type = "b", xlab = "Number of clusters (k)", ylab = "Sum of Squared Errors (SSE)")
```



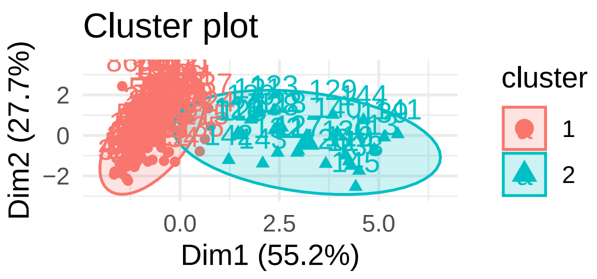
```r
# Alternaitvely, determine the optimal number of clusters using the within-cluster sum of squares (WSS)
# Both methods should give the same result
fviz_nbclust(x = diabetes_scaled_cols, FUNcluster = kmeans, method = "wss", k.max = n)
```

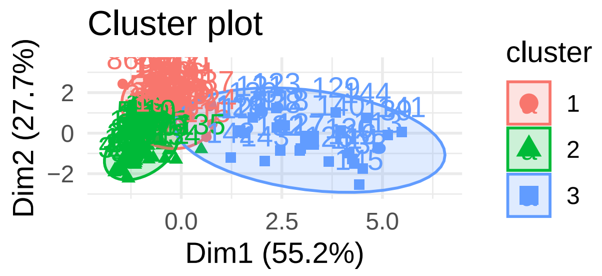


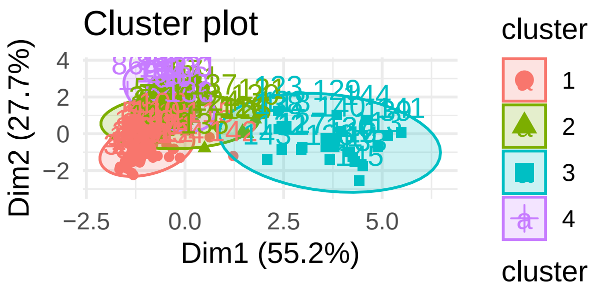
```r
#----------------------------------------------------------------------------------------
# Visualize the k-means clustering results

# Depiction 1
plot_list <- list()

for(k in 1:n) {
  # Generate the plot and store it in the list
  plot_list[[k]] <- create_scatter_plot(diabetes_scaled, list_of_cluster_vectors[[k]])
}

# Combine all the plots into a single plot
grid.arrange(grobs = plot_list, ncol = 2)
```

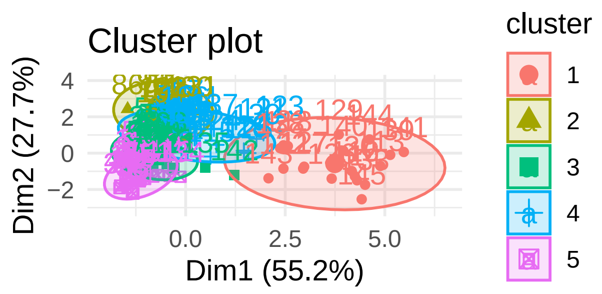


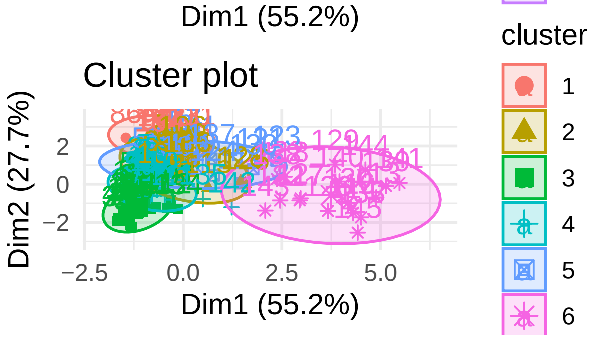
```r
# Depiction 2
plot_list <- list()

for(k in 2:n) {
  # Visualize kmeans clustering
  plot_list[[k]] <- fviz_cluster(list_of_cluster_results[[k]], diabetes_scaled_cols, ellipse.type = "no
    theme_minimal()
}

# Combine all the plots into a single plot
grid.arrange(grobs = plot_list, ncol = 2)
```











Für k = 3 lässt sich die Ground Truth Clusterstruktur reproduzieren. Die Cluster sind klar voneinander getrennt und entsprechen visuell den Ground Truth Klassen “normal”, “chemical” und “overt”.

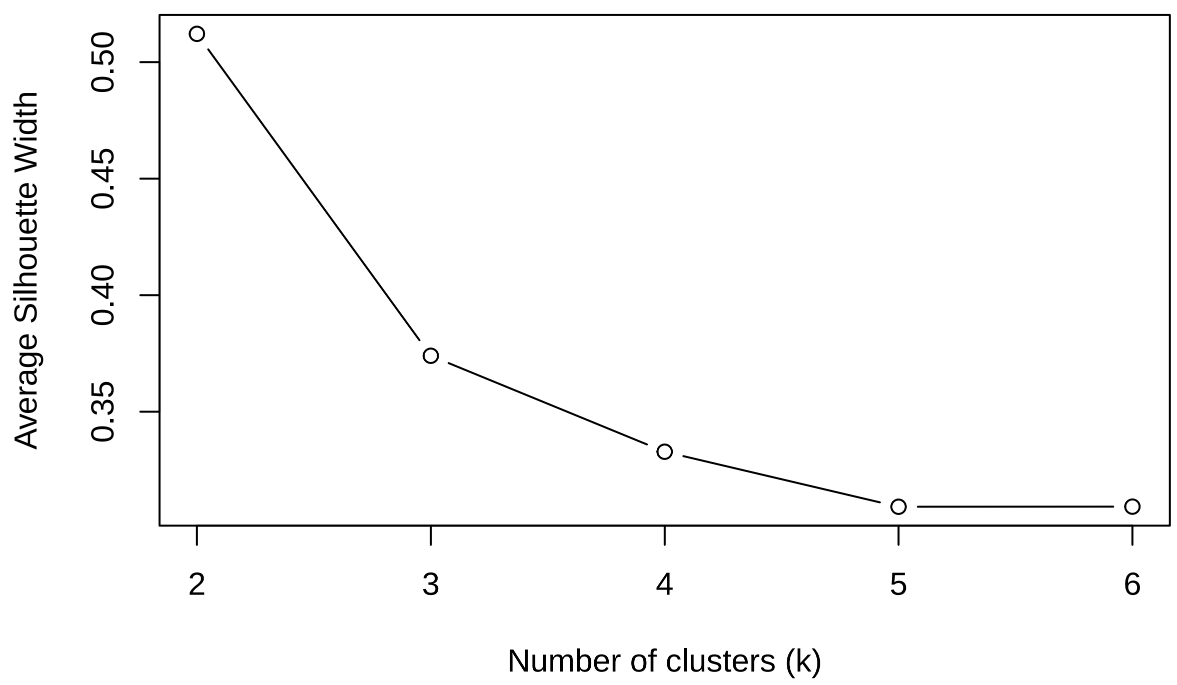
```r
set.seed(123)

n = 6
list_of_cluster_results <- list()
list_of_cluster_vectors <- list()
silhouette_list <- list()
silhouette_score <- numeric(n)

for(k in 2:n) { # silhouette score is undefined for k = 1
  kmeans_result <- kmeans(diabetes_scaled_cols, centers = k)
  cluster_stats <- cluster.stats(dist(diabetes_scaled_cols), kmeans_result$cluster)
  silhouette_values <- silhouette(kmeans_result$cluster, dist(diabetes_scaled_cols))
  silhouette_score[k] <- mean(cluster_stats$avg.silwidth)
  list_of_cluster_results[[k]] <- kmeans_result
  list_of_cluster_vectors[[k]] <- kmeans_result$cluster
  silhouette_list[[k]] <- fviz_silhouette(silhouette_values)
}
```

```
##   cluster size ave.sil.width
## 1       1  119          0.53
## 2       2   26          0.42
##   cluster size ave.sil.width
## 1       1   26          0.38
## 2       2   51          0.23
## 3       3   68          0.48
##   cluster size ave.sil.width
## 1       1   26          0.37
## 2       2   52          0.30
## 3       3   23          0.31
## 4       4   44          0.36
##   cluster size ave.sil.width
## 1       1   31          0.16
## 2       2   19          0.44
## 3       3   11          0.38
## 4       4   38          0.34
## 5       5   46          0.31
##   cluster size ave.sil.width
## 1       1   10          0.35
## 2       2   19          0.46
## 3       3   42          0.34
## 4       4   34          0.33
## 5       5   22          0.17
## 6       6   18          0.20
```

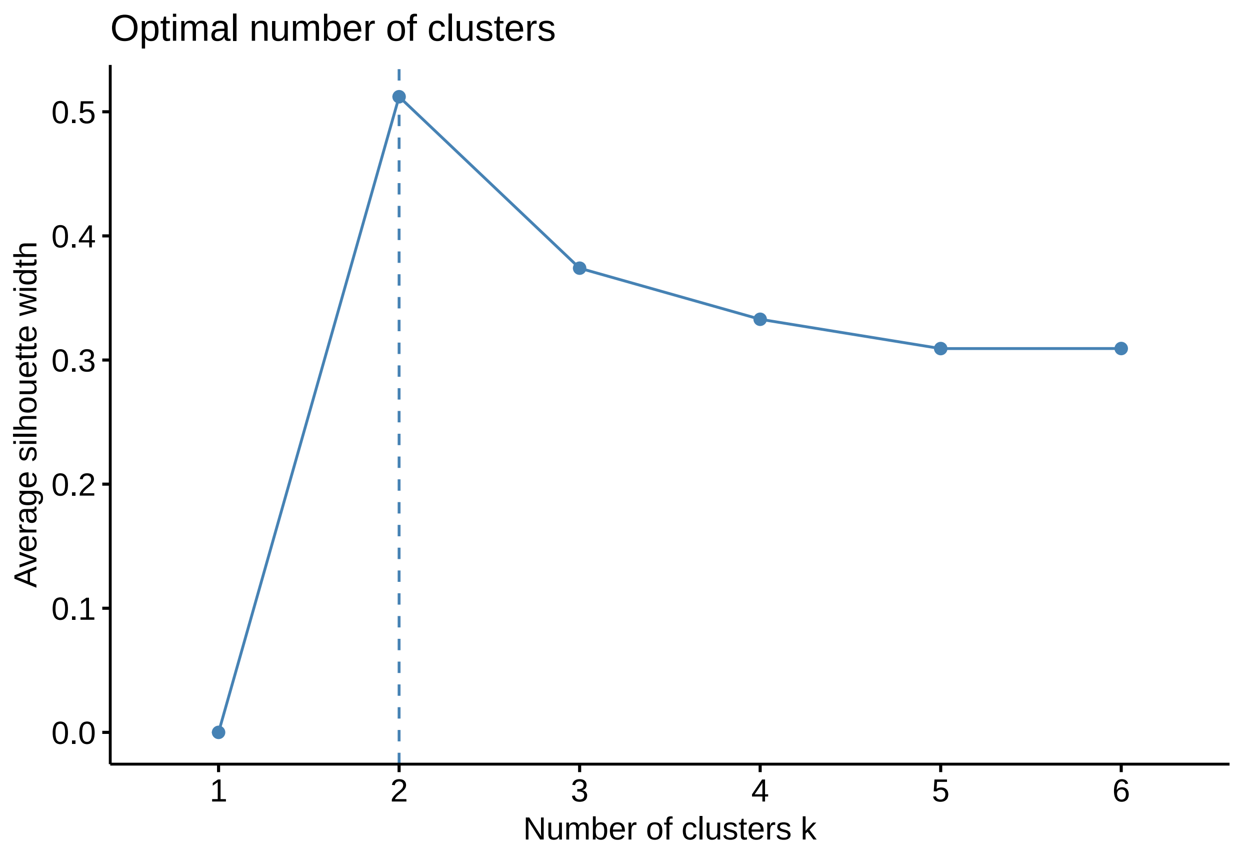
```
# Find the number of clusters that gives the highest silhouette score
best_k <- which.max(silhouette_score)

# Print the best number of clusters
print(paste("Best number of clusters (k):", best_k))
```

```
## [1] "Best number of clusters (k): 2"
```

```
#------------------------------------------------------------------------------------------
# Plot the silhouette scores

# Determine the optimal number of clusters using the silhouette score
plot(2:n, silhouette_score[2:n], type = "b", xlab = "Number of clusters (k)", ylab = "Average Silhouett
```



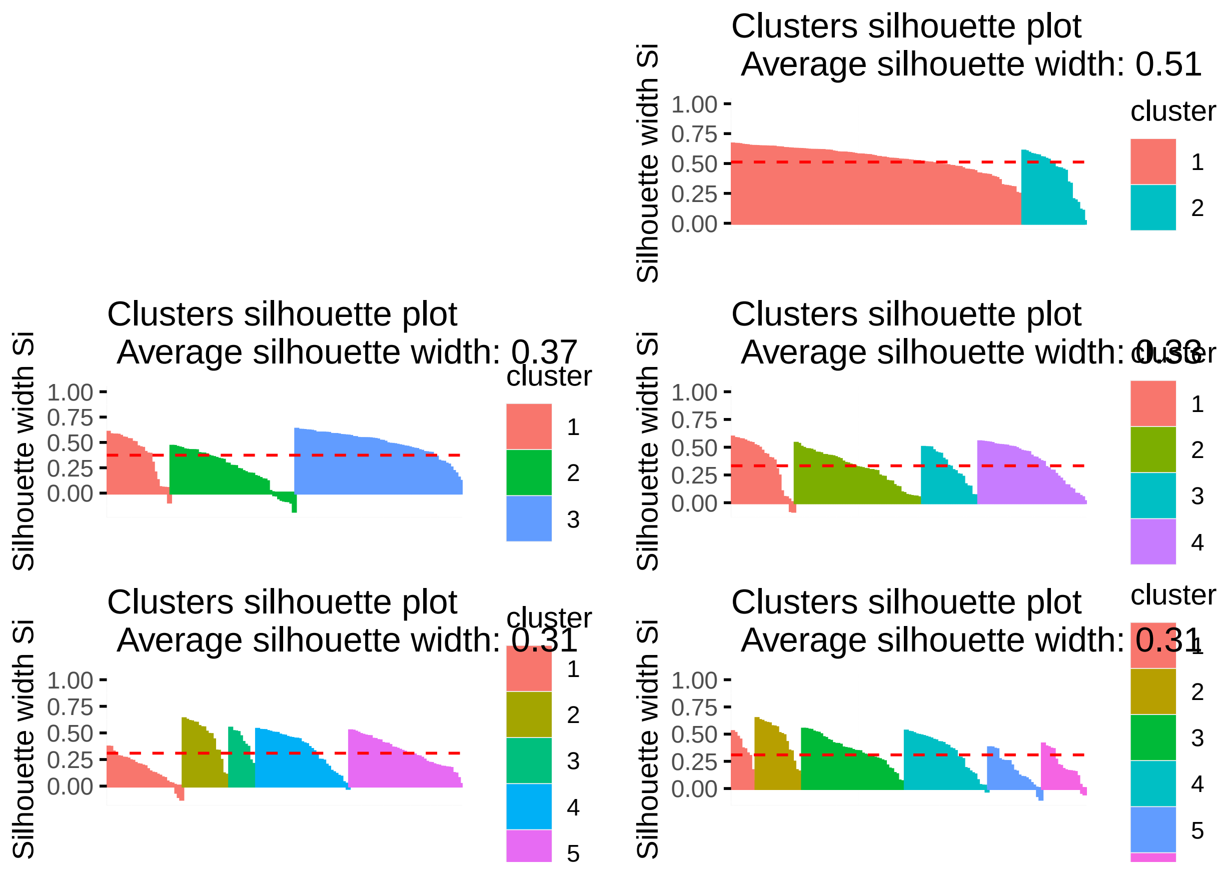
```
# Alternatively, determine the optimal number of clusters using the silhouette method
# Both methods should give the same result
fviz_nbclust(x = diabetes_scaled_cols, FUNcluster = kmeans, method = "silhouette", k.max = n)
```



```r
#-----------------------------------------------------------------------------------------
# Silhouette plots

# Initialize an empty list to store the plots
plot_list <- list()

for(k in 2:n) {
  # Visualize kmeans clustering
  # Print the silhouette plots for each number of clusters
  plot_list[[k]] <- silhouette_list[[k]]
  score <- round(cluster.stats( dist(diabetes_scaled_cols), list_of_cluster_vectors[[k]])$avg.silwidth,
  print(paste("For k = ",k," Silhouette Score: ", score))
}
```

```
## [1] "For k =  2  Silhouette Score:  0.51"
## [1] "For k =  3  Silhouette Score:  0.37"
## [1] "For k =  4  Silhouette Score:  0.33"
## [1] "For k =  5  Silhouette Score:  0.31"
## [1] "For k =  6  Silhouette Score:  0.31"
```

```r
# Combine all the plots into a single plot
grid.arrange(grobs = plot_list, ncol = 2)
```



Bestimmung des optimalen k-Werts für k-Means-Clustering mit Hilfe des des Silhouettenkoeffizienten. Laut dem Silhouettenkoeffizienten ist der optimale k-Wert 2 (0.51). Die Elbow-Methode spricht ebenso, dass der optimale k-Wert eher 2 ist. Hätten wir keine Labelinformation zur Verfügung, würde wir den k-Wert 2 wählen.

2 Clustering - Breast Cancer [4P]

Brustkrebs ist weltweit die häufigste bösartige Erkrankung bei Frauen und eine der Hauptursachen für krebsbedingte Todesfälle sowohl in Entwicklungs- als auch in Industrieländern. Verwenden Sie den Datensatz `breast_cancer.csv`. Die Merkmale werden aus einem digitalisierten Bild eines Feinnadelaspirats einer Brustmasse berechnet. Sie beschreiben Merkmale der im Bild vorhandenen Zellkerne. Das Zielmerkmal erfasst die Prognose gutartig (B) oder bösartig (M) und dient hier als Ground Truth. Achten Sie auf uninformative Features (z.B. ID) und fehlende Daten (Missing Values).

- Führen Sie eine Clusteranalyse durch, um eine etwaige Clusterstruktur zwischen gutartigen und bösartigen Zellen zu identifizieren.
- Vergleichen Sie die Genauigkeit folgender Cluster-Algorithmen:
  - k-means
  - hierarchisches Clustering
  - Modell-basiertes Clustering
  - DBSCAN
- Welches Verfahren ist am besten geeignet?
- Stellen Sie die Ergebnisse grafisch dar (Scatter Plot z.B. radius_mean vs. texture_mean).

- Lässt sich das Ergebnis verbessern, wenn vor dem Clustering der Merkmalsraum mittels PCA reduziert wird? Vergleichen Sie die Ergebnisse mit den vorherigen Resultaten.

```r
# Load the data
breast_cancer <- read.csv("breast_cancer.csv", header = TRUE, sep = ",")
str(breast_cancer)
```

```
## 'data.frame':    569 obs. of  31 variables:
##  $ diagnosis              : chr  "M" "M" "M" "M" ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean              : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst             : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

```r
#summary(breast_cancer)

# check for missing values
sum(is.na(breast_cancer))
```

```
## [1] 0
```

```r
# Preprocess the data
# Remove the diagnosis column (ground truth)
breast_cancer_scaled <- scale(breast_cancer[, -c(1)])
breast_cancer_scaled <- as.data.frame(breast_cancer_scaled)
```

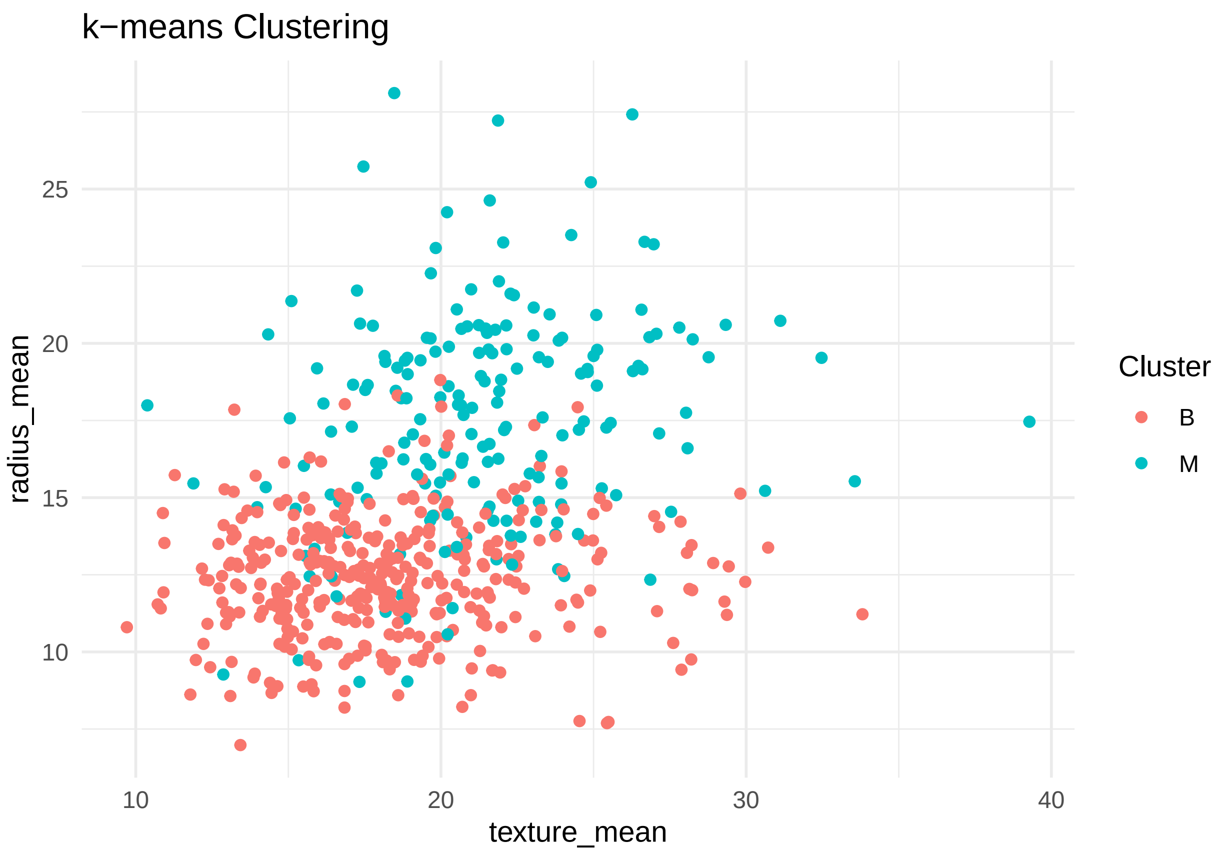
```r
# Perform k-means clustering
kmeans_result <- kmeans(breast_cancer_scaled, centers = 2, nstart = 12)
groups <- kmeans_result$cluster
groups <- car::recode(groups, "1='M'; 2='B'", as.factor = TRUE)

table(breast_cancer$diagnosis, groups)
```

```
##    groups
##       B   M
##   B 343  14
##   M  37 175
```

```r
acc_kmeans <- mean(breast_cancer$diagnosis==groups)

cat("Accuracy of k-means clustering: ", acc_kmeans, "\n")
```

```
## Accuracy of k-means clustering:  0.9103691
```

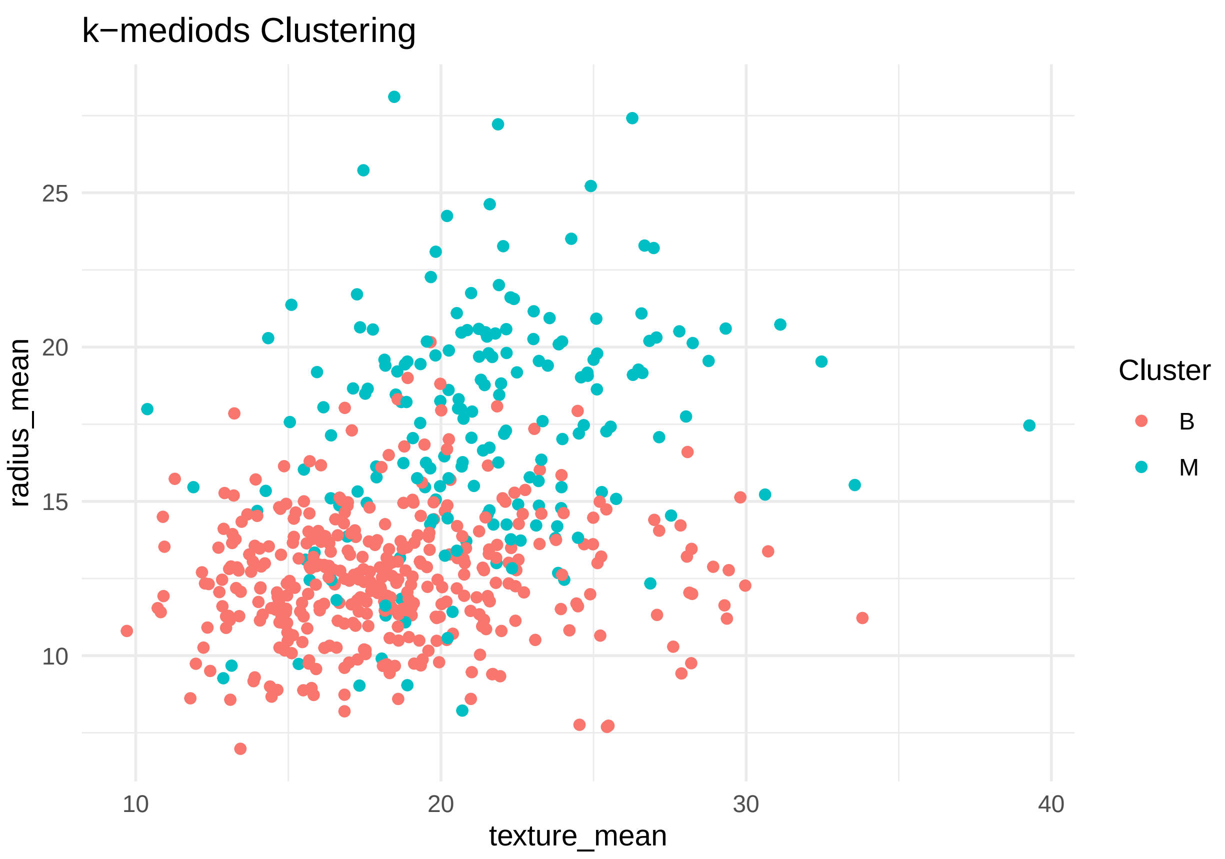
```r
# Visualize the clusters (radius_mean vs. texture_mean)
ggplot(breast_cancer, aes(x = texture_mean, y = radius_mean, color = groups)) +
  geom_point() +
  labs(title = "k-means Clustering", color = "Cluster") +
  theme_minimal()
```



```r
# Perform k-medoids clustering (PAM)
ncl.pam<-cluster::pam(breast_cancer_scaled,k = 2)

groups<-ncl.pam$clustering
groups
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 2 2 2 2 2 1 2 2 1 2 1 2 2 2 2 2 1 2 2 1 1 2 2 2 2 1 2 1 1 2 2 1 2 1 2 1 2
##  [75] 2 1 2 1 1 2 2 1 1 1 2 1 2 1 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2
## [112] 2 1 2 2 2 2 1 1 2 2 1 1 2 2 2 2 2 1 1 2 1 2 2 1 2 2 2 1 2 2 2 2 2 2 2 1 2
## [149] 2 2 2 1 1 2 2 2 1 2 2 2 2 1 1 2 1 2 2 2 1 2 2 2 1 2 2 2 1 1 2 2 1 1 2 2 2
## [186] 2 2 2 2 2 1 2 2 1 1 2 1 2 1 1 2 1 1 1 2 2 2 2 2 2 1 2 1 1 1 1 2 2 1 1 2 2
## [223] 2 1 2 2 2 2 2 1 1 2 2 1 2 2 1 1 2 1 2 2 1 2 1 2 2 2 2 2 1 2 1 2 1 2 1 1 1
## [260] 1 1 2 1 2 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 1 1 2 2 2 2 2 2 1 2 2 2 2 2
## [297] 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 1 2 2 2 2 1 1 1 2 2
## [334] 2 2 1 2 1 2 1 2 2 2 1 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 1
## [371] 1 2 1 1 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2
## [408] 2 1 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 1 2 1 1 2 2 2 2 2 2 2 1 2 2
## [445] 2 2 1 2 2 1 2 1 2 2 2 2 2 2 2 2 1 1 2 2 2 1 2 2 1 1 2 2 2 2 2 2 2 2 2 1 2
## [482] 2 2 2 2 1 2 1 2 2 2 2 1 2 2 2 2 2 1 1 2 1 2 1 1 1 2 2 2 1 2 2 1 2 2 2 1 1
## [519] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [556] 2 2 2 2 2 2 2 1 1 1 1 2 1 2
```

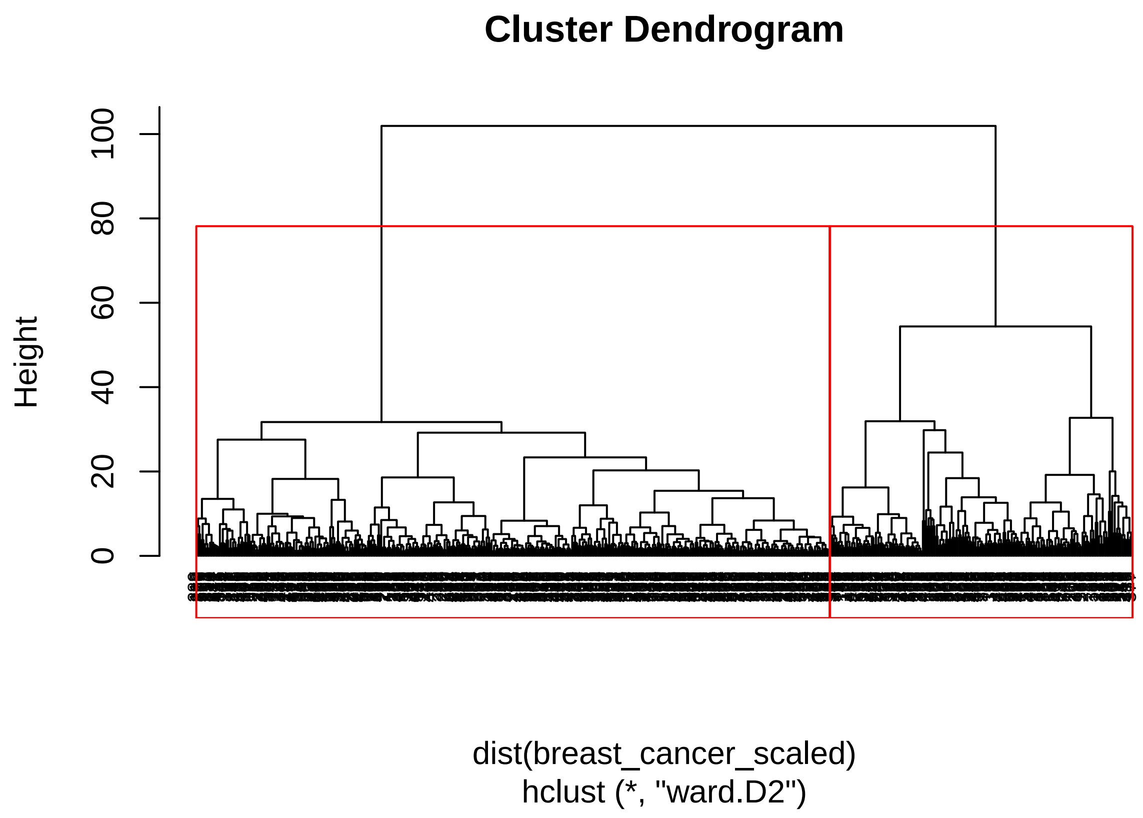
```r
groups<-car::recode(groups,recodes="1='M';2='B'",as.factor = T)
acc_kmediod <- mean(breast_cancer$diagnosis==groups)

# Visualize the clusters (area_mean vs. smoothness_mean)
ggplot(breast_cancer, aes(x = texture_mean, y = radius_mean, color = groups)) +
  geom_point() +
  labs(title = "k-mediods Clustering", color = "Cluster") +
  theme_minimal()
```



Interpretation: Die k-means-Clustering-Methode hat eine Genauigkeit von 0,9103, was bedeutet, dass 91% der Daten dem korrekten Cluster zugeordnet wurden. Die Genauigkeit kann in unserem Fall ermittelt werden, da wir die Ground-Truth-Informationen haben. Die Visualisierung anhand der Merkmale `texture_mean` und `radius_mean` zeigt, dass die Cluster stark überlappen und diese zufällig gewählten Merkmale eine geringe Trennschärfe aufweisen.

Beim k-medoids-Clustering beträgt die Genauigkeit 0,8910. Im Vergleich zu k-means ist die Genauigkeit etwas niedriger, aber immer noch relativ hoch. Die Visualisierung anhand der Merkmale `area_mean` und `smoothness_mean` zeigt ebenfalls eine starke Überlappung der Cluster.

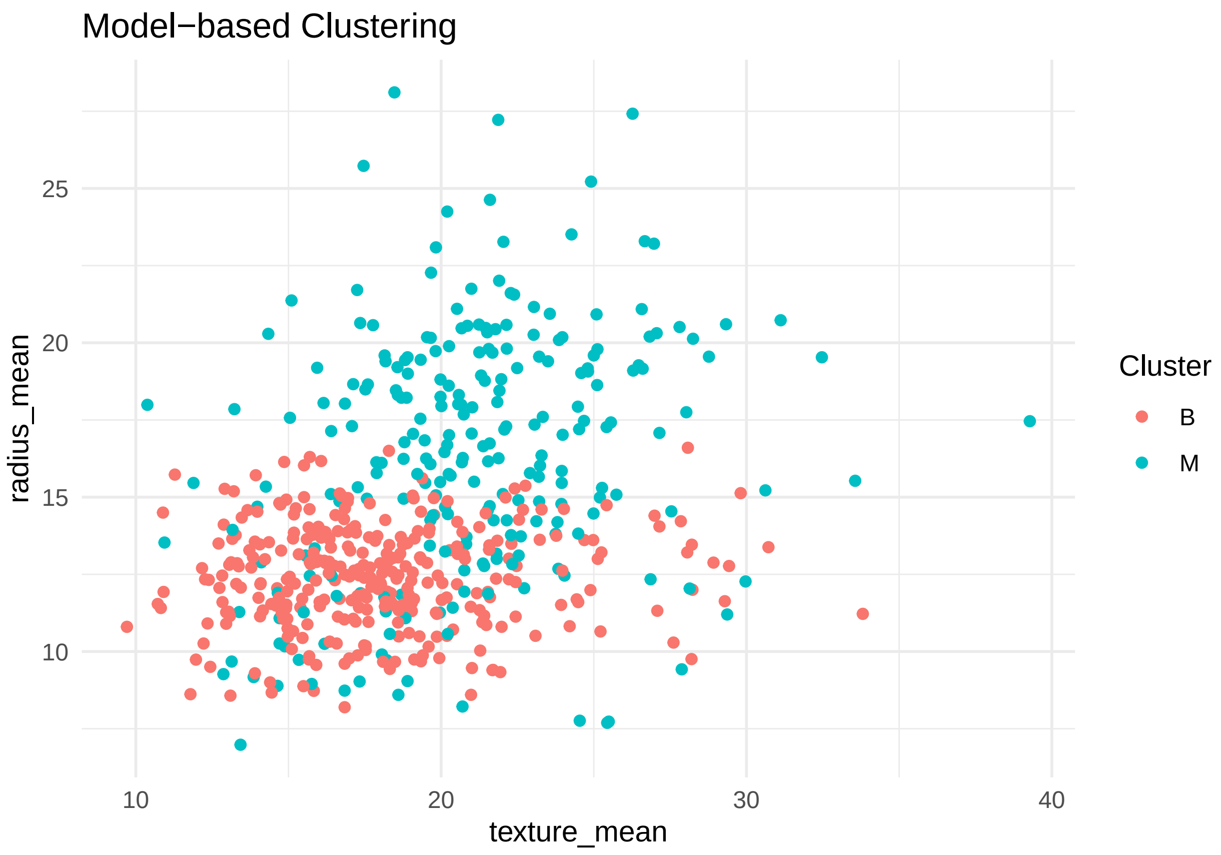
```r
# Perform hierarchical clustering
hclust <- hclust(dist(breast_cancer_scaled), method = "ward.D2")
plot(hclust,cex=0.6, hang = -1)
# Cluster einzeichnen
rect.hclust(hclust, k = 2, border = "red")
```



```r
# Cluster zuweisen
groups <- cutree(hclust, k = 2)
groups <- car::recode(groups, "1='M'; 2='B'", as.factor = TRUE)

acc_hclust <- mean(breast_cancer$diagnosis==groups)
```

Das hierarchische Clustering hat erfolgreich zwei Hauptcluster identifiziert, die mit einer Genauigkeit von 88% den tatsächlichen Diagnosen (benign vs. malignant) entsprechen. Auch durch hierarchisches Clustering ist also eine klare Trennung der Datenpunkte möglich. Die Genauigkeit ist jedoch etwas niedriger als bei k-means und k-medoids.

```r
# Perform model-based clustering
mclust_result <- Mclust(breast_cancer_scaled, G = 2)
groups <- mclust_result$classification
groups <- car::recode(groups, "1='M'; 2='B'", as.factor = TRUE)

acc_mclust <- mean(breast_cancer$diagnosis==groups)

# Visualize the clusters (texture_mean vs. radius_mean)
ggplot(breast_cancer, aes(x = texture_mean, y = radius_mean, color = groups)) +
  geom_point() +
  labs(title = "Model-based Clustering", color = "Cluster") +
  theme_minimal()
```

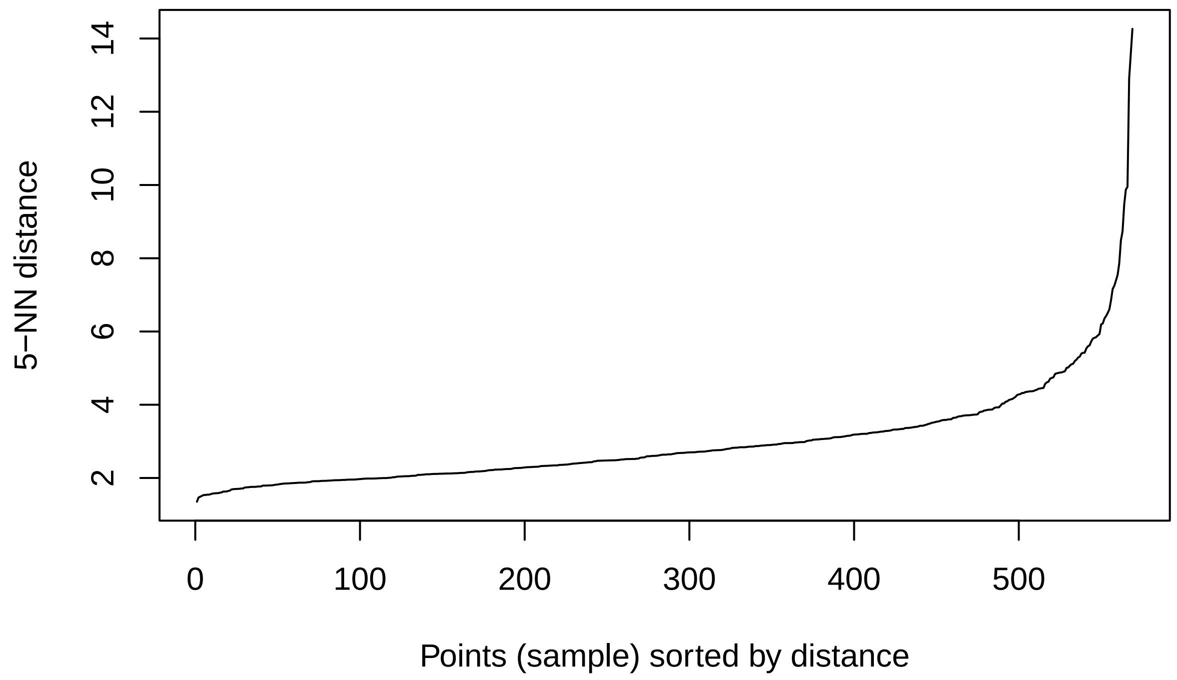


Das modellbasierte Clustering hat eine Genauigkeit von 0,8910, was der Genauigkeit des k-mediod-Clustering entspricht. Dies bedeutet, dass das modellbasierte Clustering in diesem Fall ähnlich effektiv ist wie k-medoids. Die Visualisierung zeigt eine klare Trennung der Cluster anhand der Merkmale `texture_mean` und `radius_mean`.

```r
# Perform DBSCAN clustering
set.seed(123)
dbscan_result <- dbscan(breast_cancer_scaled, eps = 5, MinPts = 50)
```

```
## Warning in dbscan(breast_cancer_scaled, eps = 5, MinPts = 50): converting
## argument MinPts (fpc) to minPts (dbscan)!
```

```r
groups <- dbscan_result$cluster
groups <- car::recode(groups, "0='M'; 1='B'", as.factor = TRUE)

acc_dbscan <- mean(breast_cancer$diagnosis==groups)

# Berechnung der k-nearest Neighbour-Distanz für k = MinPts
kNNdistplot(breast_cancer_scaled, k = 5)  # Hier setzen wir k = MinPts (Startwert)
abline(h = 0.5, col = "red", lty = 2)
```



DBSCAN scheint für diesen Datensatz nicht optimal zu sein, um zwei Cluster (benign vs. malignant) zu identifizieren. Unabhängig von den Parametern (eps, MinPts) konnte immer nur ein Cluster und Rauschen identifiziert werden. Die ermittelte Genauigkeit beträgt demnach nur 0.6432. Alternative Methoden wie k-Means und hierarchisches Clustering könnten somit besser geeignet sein, um die Daten in zwei Hauptcluster zu unterteilen.

```r
# Accuracy comparison
accuracy <- c(acc_kmeans, acc_kmediod, acc_hclust, acc_mclust, acc_dbscan)
method <- c("k-means", "k-medoids", "hierarchical", "model-based", "DBSCAN")
accuracy_df <- data.frame(method, accuracy)
accuracy_df
```

```
##         method  accuracy
## 1      k-means 0.9103691
## 2    k-medoids 0.8910369
## 3 hierarchical 0.8804921
## 4  model-based 0.8629174
## 5       DBSCAN 0.6432337
```

Die Untersuchung unterschiedlicher Clustering-Methoden auf den Brustkrebsdatensatz zeigt, dass das k-Means-Verfahren mit einer Genauigkeit von 91% am besten abschneidet. Das k-Medoids-Verfahren und das modellbasierte Clustering folgen dicht dahinter mit einer Genauigkeit von jeweils etwa 89%. Hierarchisches Clustering zeigt ebenfalls gute Ergebnisse mit einer Genauigkeit von 88%. Im Vergleich dazu zeigt das dichtebasierte DBSCAN-Verfahren eine deutlich geringere Genauigkeit von 64%, was darauf hinweist, dass dieses Verfahren für den gegebenen Datensatz nicht geeignet ist.

Jetzt wird überprüft, ob durch eine vorhergehende PCA eine bessere Trennung der Cluster erreicht werden kann.

```r
# Dimensionality Reduction with PCA
pca_result <- prcomp(breast_cancer_scaled, scale. = TRUE)
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010
##                            PC8    PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     0.69037 0.6457 0.59219 0.5421 0.51104 0.49128 0.39624
## Proportion of Variance 0.01589 0.0139 0.01169 0.0098 0.00871 0.00805 0.00523
## Cumulative Proportion  0.92598 0.9399 0.95157 0.9614 0.97007 0.97812 0.98335
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
## Standard deviation     0.30681 0.28260 0.24372 0.22939 0.22244 0.17652 0.1731
## Proportion of Variance 0.00314 0.00266 0.00198 0.00175 0.00165 0.00104 0.0010
## Cumulative Proportion  0.98649 0.98915 0.99113 0.99288 0.99453 0.99557 0.9966
##                           PC22    PC23   PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.16565 0.15602 0.1344 0.12442 0.09043 0.08307 0.03987
## Proportion of Variance 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023 0.00005
## Cumulative Proportion  0.99749 0.99830 0.9989 0.99942 0.99969 0.99992 0.99997
##                           PC29    PC30
## Standard deviation     0.02736 0.01153
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

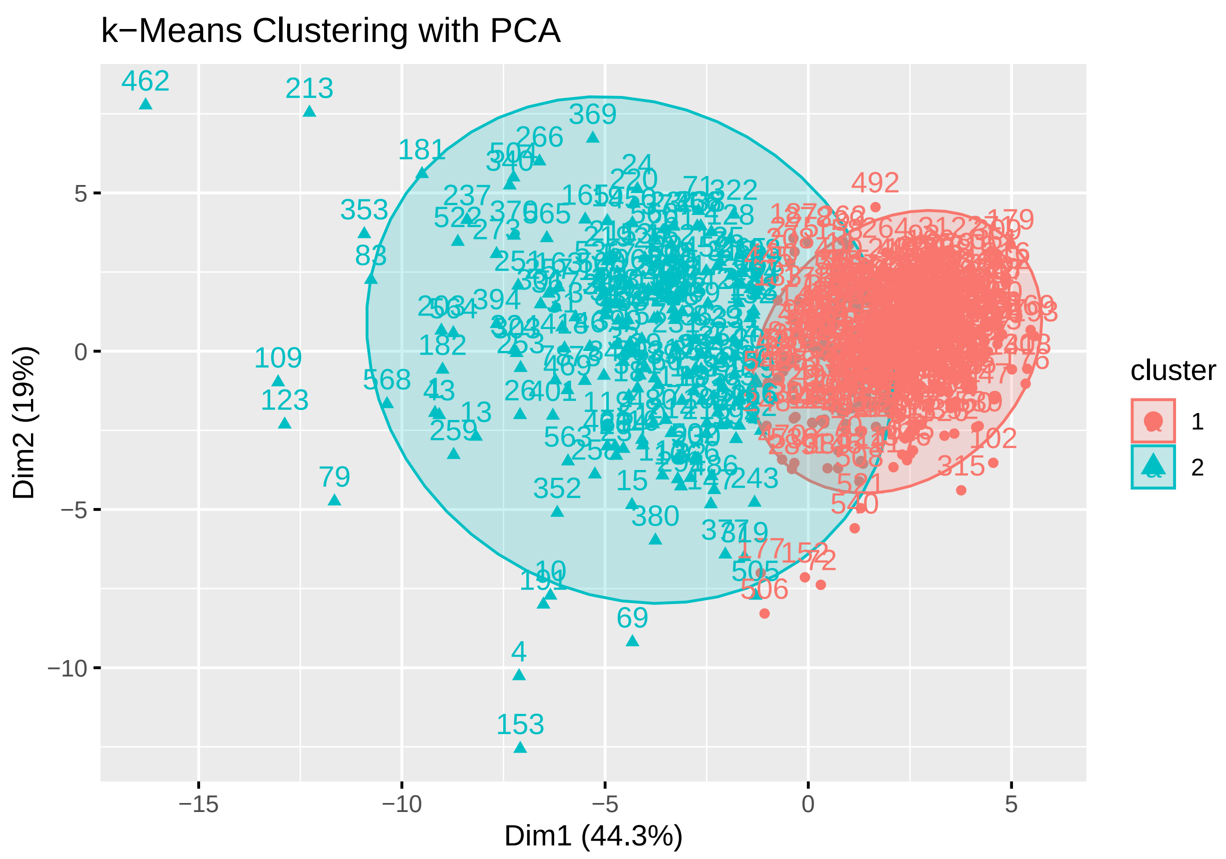
```r
# create a data frame with the first 10 principal components
pca_data <- data.frame(pca_result$x[, 1:10])

# k-Means Clustering
set.seed(123)
kmeans_pca <- kmeans(pca_data, centers = 2, nstart = 20)
groups.kmeans_pca <- kmeans_pca$cluster
groups.kmeans_pca <- car::recode(groups.kmeans_pca, "1='B'; 2='M'", as.factor = TRUE)

# k-Medoids Clustering
kmedoids_pca <- pam(pca_data, k = 2)
groups.kmedoids_pca <- kmedoids_pca$clustering
groups.kmedoids_pca <- car::recode(groups.kmedoids_pca, "1='M'; 2='B'", as.factor = TRUE)

# Hierarchisches Clustering
dist_pca <- dist(pca_data)
hc_pca <- hclust(dist_pca, method = "ward.D2")
groups.hc_pca <- cutree(hc_pca, k = 2)
groups.hc_pca <- car::recode(groups.hc_pca, "1='M'; 2='B'", as.factor = TRUE)


# Modell-basiertes Clustering
mbc_pca <- Mclust(pca_data, G = 2)
groups.mbc_pca <- mbc_pca$classification
groups.mbc_pca <- car::recode(groups.mbc_pca, "1='M'; 2='B'", as.factor = TRUE)

# DBSCAN Clustering
dbscan_pca <- dbscan(pca_data, eps = 1, MinPts = 5)
```

```
## Warning in dbscan(pca_data, eps = 1, MinPts = 5): converting argument MinPts
## (fpc) to minPts (dbscan)!
```

```r
groups.dbscan_pca <- dbscan_pca$cluster
groups.dbscan_pca <- car::recode(groups.dbscan_pca, "0='B'; 1='M'", as.factor = TRUE)

# Berechnung der Genauigkeiten
acc_kmeans_pca <- mean(breast_cancer$diagnosis == groups.kmeans_pca)
acc_kmedoids_pca <- mean(breast_cancer$diagnosis == groups.kmedoids_pca)
acc_hc_pca <- mean(breast_cancer$diagnosis == groups.hc_pca)
acc_mbc_pca <- mean(breast_cancer$diagnosis == groups.mbc_pca)
acc_dbscan_pca <- mean(breast_cancer$diagnosis == groups.dbscan_pca)

# Accuracy comparison
accuracy_pca <- c(acc_kmeans_pca, acc_kmedoids_pca, acc_hc_pca, acc_mbc_pca, acc_dbscan_pca)
method_pca <- c("k-means", "k-medoids", "hierarchical", "model-based", "DBSCAN")
accuracy_df_pca <- data.frame(method_pca, accuracy_pca)
accuracy_df_pca
```

```
##     method_pca accuracy_pca
## 1      k-means    0.9103691
## 2    k-medoids    0.9138840
## 3 hierarchical    0.9191564
## 4  model-based    0.6362039
## 5       DBSCAN    0.6274165
```

```r
# visualize the clusters of kmeans with PCA - Dimensionality Reduction
fviz_cluster(object=kmeans_pca, data=breast_cancer_scaled,ellipse.type = "norm",repel = F) + ggtitle("k
```



Die Auswirkung der PCA auf die Genauigkeit der Clusteranalyse zeigt deutliche Unterschiede bei den einzelnen Methoden. Während sich die Genauigkeit des k-Means-Verfahrens nicht verbessert hat, zeigt die k-Medoids eine Verbesserung von 89.1% auf 91.38%. Die hierarchische Clusteranalyse zeigt die stärkste Verbesserung, mit einem Sprung von 88% auf 91.92%. Im Gegensatz dazu zeigt das modellbasierte Clustering eine deutliche Verschlechterung der Genauigkeit von 89.1% auf 63.62%. Offenbar ist das modellbasierte Clustering weniger gut geeignet, um die Daten in zwei Hauptcluster zu unterteilen, nachdem die Dimensionalität reduziert wurde. Das DBSCAN-Verfahren zeigt ebenfalls eine Verschlechterung der Genauigkeit von 64% auf nur 36%, wobei hier wieder nur ein Cluster und Rauschen iden

3 Clustering - Heart Disease Patients [3P]

Verwenden Sie den Datensatz `heart_disease_patients.csv`. Der Datensatz enthält anonymisierte Daten von Patienten, bei denen eine Herzerkrankung diagnostiziert wurde. Patienten mit ähnlichen Merkmalen könnten auf die gleichen Behandlungen ansprechen, und Ärzte könnten davon profitieren, etwas über die Behandlungsergebnisse von Patienten zu erfahren, die denen ähneln, die sie behandeln. Zu diesem Zweck führen Sie bitte eine Clusteranalyse durch. Vergleichen Sie unterschiedliche Algorithmen und versuchen Sie ein bestmögliches Ergebnis zu erreichen. Begründen Sie ihre Entscheidungen. Verwenden Sie nur numerische Merkmale und achten Sie auf uninformative Features und fehlende Daten. Versuchen Sie die resultierenden Cluster zu interpretieren.

PCA and Summary Stats

```r
load_source()
# Data Preparation
# Load the data
heart_disease_patients_tot <- read.csv("heart_disease_patients.csv")

str(heart_disease_patients_tot)
```

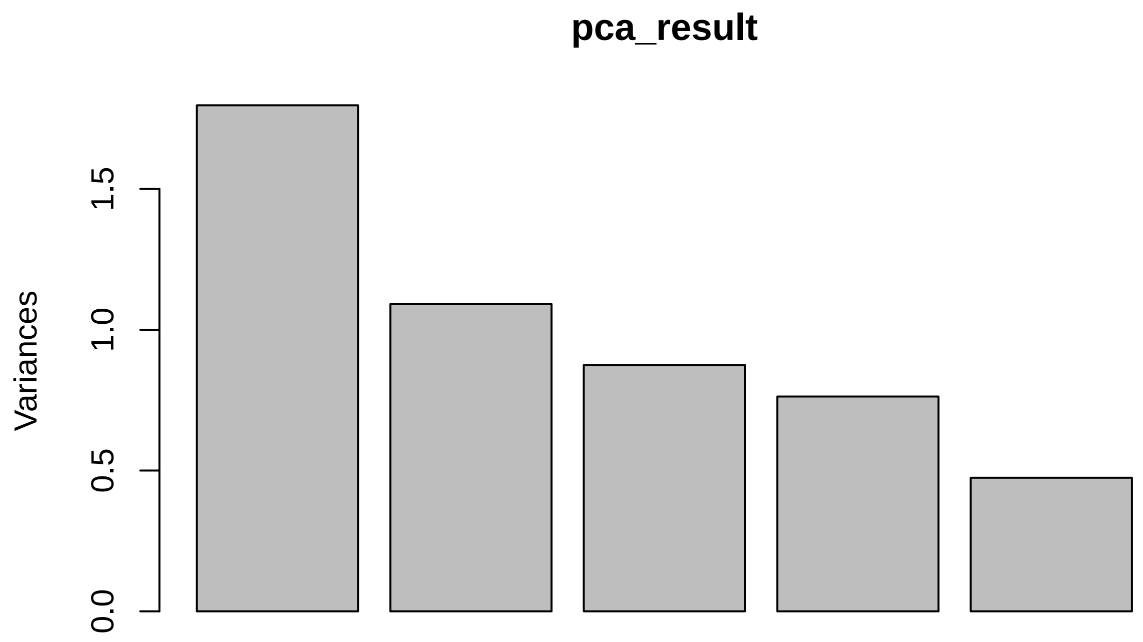
```
## 'data.frame':    303 obs. of  12 variables:
##  $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : int  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : int  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : int  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : int  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : int  3 2 2 3 1 1 3 1 2 3 ...
```

```r
# Remove uninformativ non-numeric features
heart_disease_patients <- subset(heart_disease_patients_tot, select= -c(id,sex,cp,fbs,restecg,exang,slo
# print(heart_disease_patients)

# Perform exploratory data analysis
# summary(heart_disease_patients)

# Handle missing data
heart_disease_patients <- na.omit(heart_disease_patients)

# Normalize the numerical features (assuming all features are numerical)
heart_disease_patients_scaled <- scale(heart_disease_patients, center = TRUE)

#-----------------------------------------------------------------------------------------
# Perform PCA
# Using subset
pca_result <- prcomp(heart_disease_patients_scaled)

# Print summary of the PCA result
summary(pca_result)
```

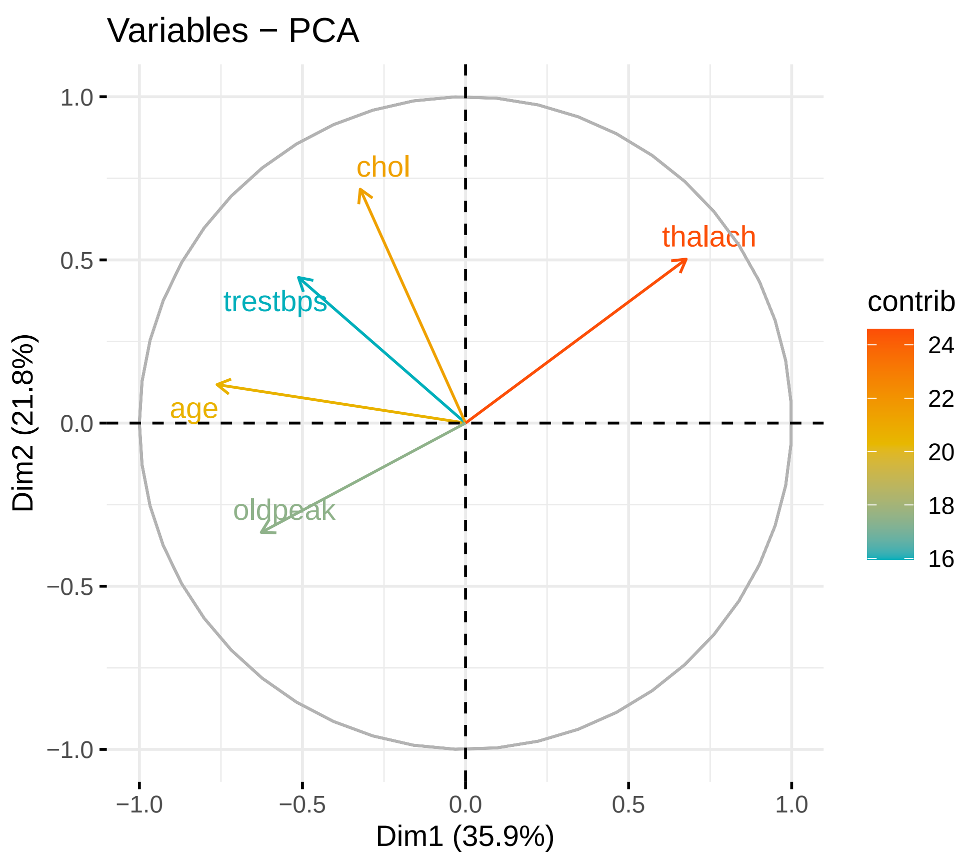
```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5
## Standard deviation     1.3406 1.0446 0.9352 0.8734 0.68865
## Proportion of Variance 0.3594 0.2182 0.1749 0.1525 0.09485
## Cumulative Proportion  0.3594 0.5777 0.7526 0.9052 1.00000
```

```r
# Get the data in the PC space
data <- pca_result$x

# Max number of clusters
n = 6
```

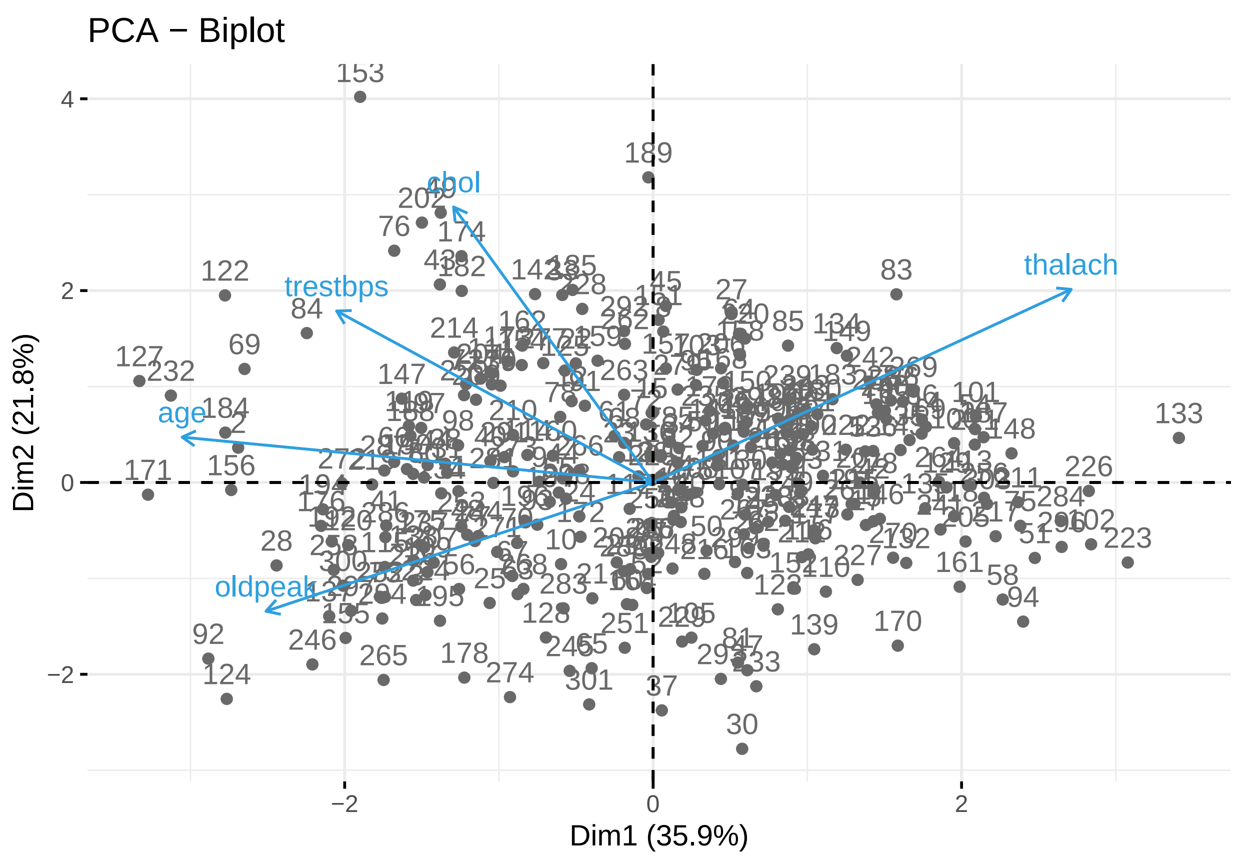
```r
#-----------------------------------------------------------------------------------------
# PCA Plots

# Plot the variance explained by each principal component
plot(pca_result)
```



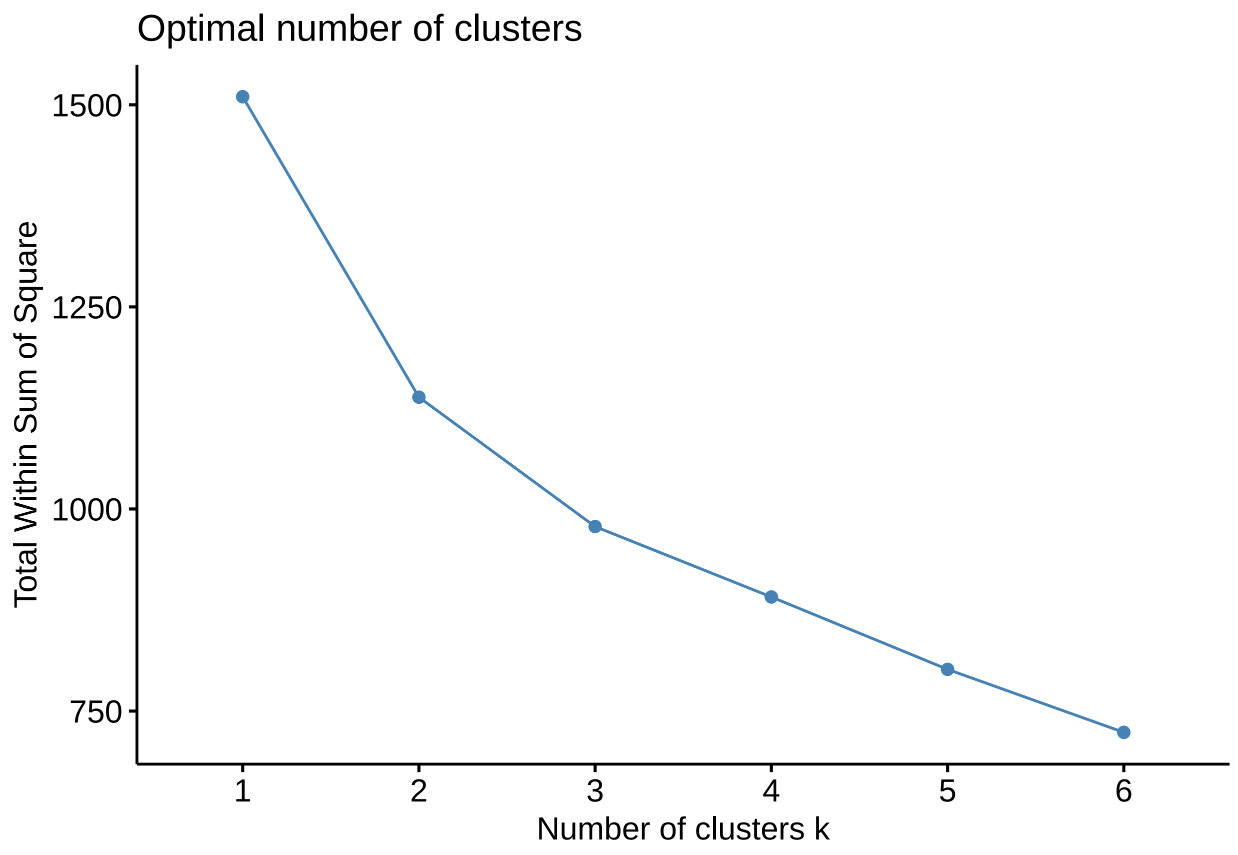
```r
# Plot the correlation circle
fviz_pca_var(pca_result, col.var="contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel =
```



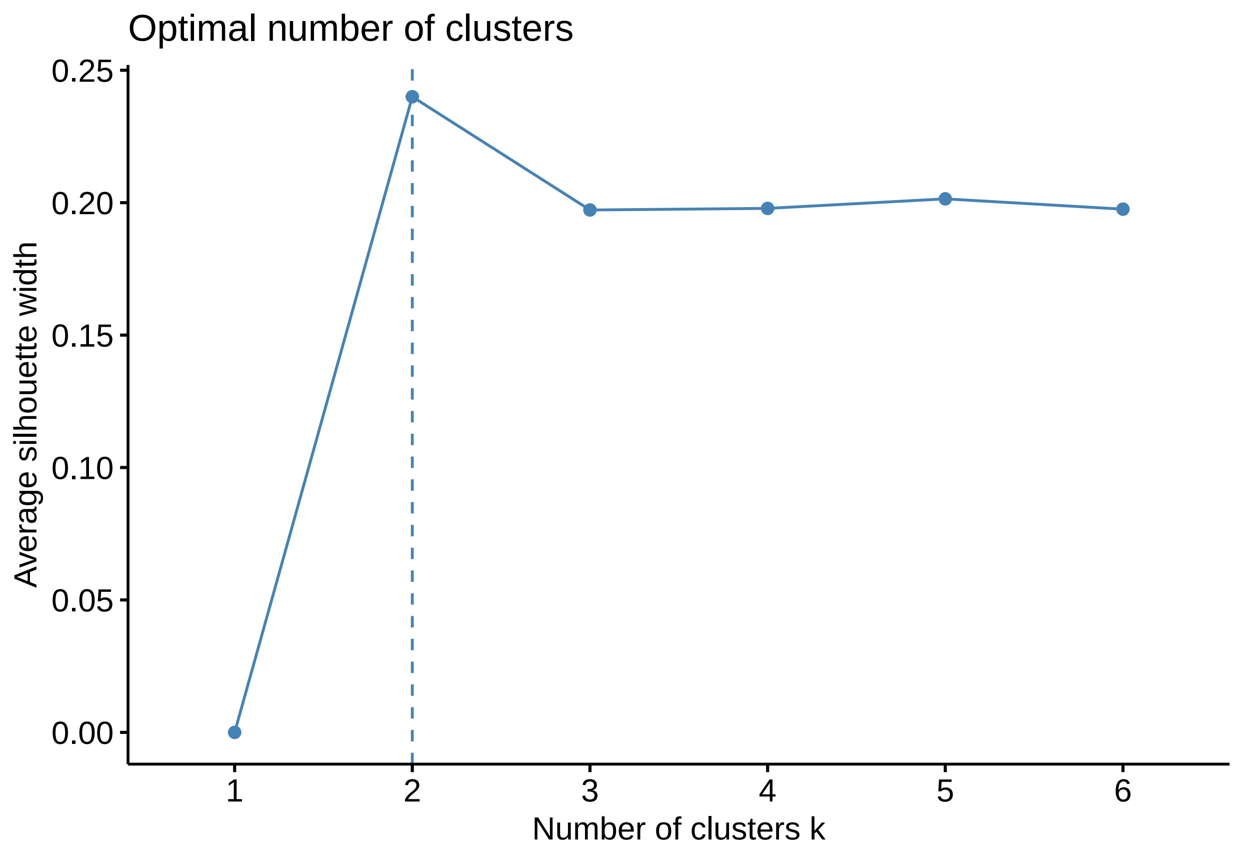
```
# Plot the biplot
fviz_pca_biplot(pca_result, col.var="#2E9FDF", col.ind="#696969")
```



```
#------------------------------------------------------------------------------------------
# Plots for Number of Clusters

# Determine the optimal number of clusters using the within-cluster sum of squares (WSS)
fviz_nbclust(x = data, FUNcluster = kmeans, method = "wss", k.max = n)
```



```r
# Determine the optimal number of clusters using the silhouette method
fviz_nbclust(x = data, FUNcluster = kmeans, method = "silhouette", k.max = n)
```



```
#-----------------------------------------------------------------------------------------
# Silhouette Scores

n = 6 # max number of clusters

# Cluster vectors are generated by different clustering methods for a given k
# They are stored in a list which is generated for each k
# These lists are stored inside another list
lists_of_cluster_vectors <- list()

# Create a list of lists of cluster vectors for different k
for (k in 1:n) {
  lists_of_cluster_vectors[[k]] <- get_list_of_cluster_vectors(data, k) # for m clustering methods and
  # Name each list of cluster vectors 'k'
  names(lists_of_cluster_vectors)[k] <- as.character(k)
}

m = length(lists_of_cluster_vectors[[1]])
# Loop over k lists of cluster vectors
# and pick the i-th cluster vector from that list
for( i in 1:m) {

  print(paste("Method Used: ", names(lists_of_cluster_vectors[[1]])[i]))
  for(k in 2:n) {
    score <- round(cluster.stats( dist(data), lists_of_cluster_vectors[[k]][[i]])$avg.silwidth,2)
    print(paste("For k = ",names(lists_of_cluster_vectors)[k]," Silhouette Score: ", score))
  }

}
```

```
## [1] "Method Used:  kmeans"
## [1] "For k =  2  Silhouette Score:  0.24"
## [1] "For k =  3  Silhouette Score:  0.19"
## [1] "For k =  4  Silhouette Score:  0.2"
## [1] "For k =  5  Silhouette Score:  0.21"
## [1] "For k =  6  Silhouette Score:  0.19"
## [1] "Method Used:  kmedoids"
## [1] "For k =  2  Silhouette Score:  0.19"
## [1] "For k =  3  Silhouette Score:  0.17"
## [1] "For k =  4  Silhouette Score:  0.16"
## [1] "For k =  5  Silhouette Score:  0.13"
## [1] "For k =  6  Silhouette Score:  0.14"
## [1] "Method Used:  hkmeans"
## [1] "For k =  2  Silhouette Score:  0.23"
## [1] "For k =  3  Silhouette Score:  0.2"
## [1] "For k =  4  Silhouette Score:  0.2"
## [1] "For k =  5  Silhouette Score:  0.21"
## [1] "For k =  6  Silhouette Score:  0.17"
## [1] "Method Used:  mclust"
## [1] "For k =  2  Silhouette Score:  0.17"
## [1] "For k =  3  Silhouette Score:  0.09"
## [1] "For k =  4  Silhouette Score:  0.05"
## [1] "For k =  5  Silhouette Score:  0.15"
## [1] "For k =  6  Silhouette Score:  0.11"
## [1] "Method Used:  hclust"
## [1] "For k =  2  Silhouette Score:  0.36"
## [1] "For k =  3  Silhouette Score:  0.22"
## [1] "For k =  4  Silhouette Score:  0.19"
## [1] "For k =  5  Silhouette Score:  0.19"
## [1] "For k =  6  Silhouette Score:  0.14"
```

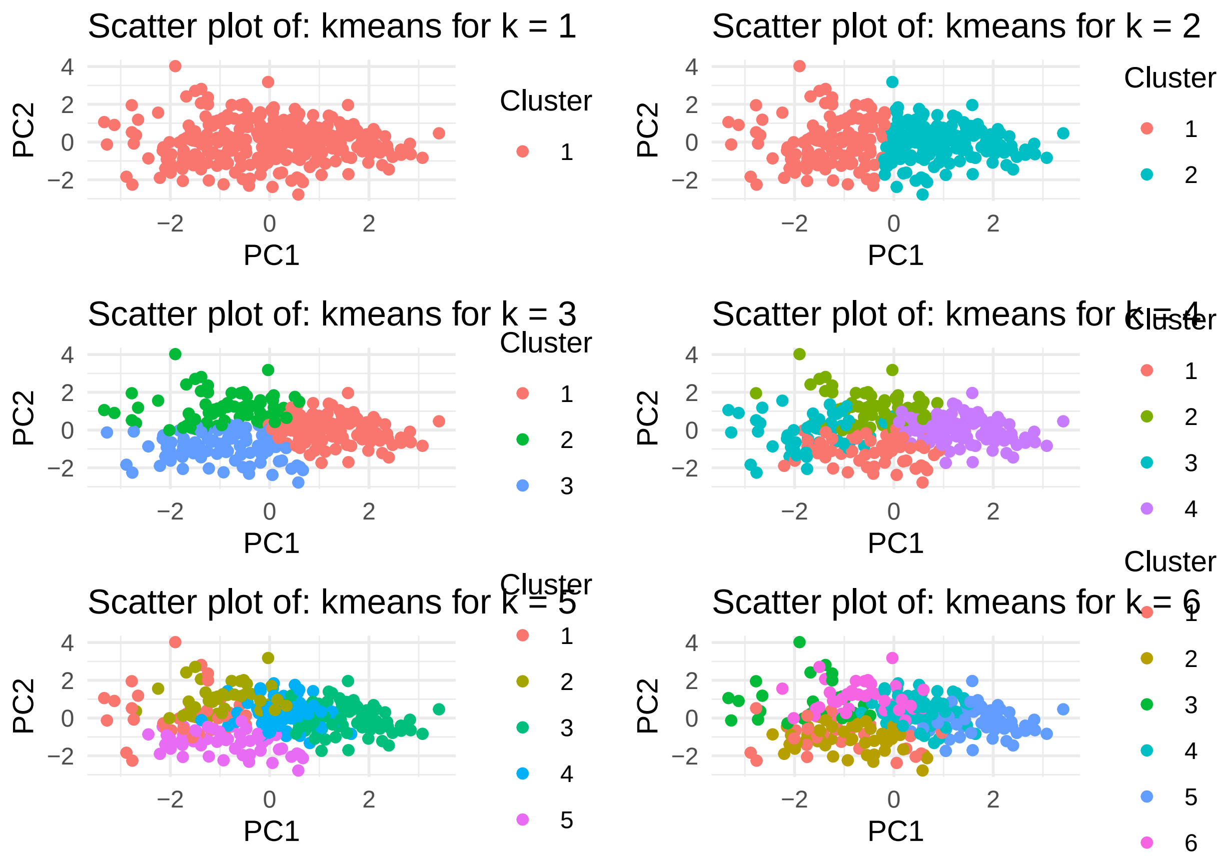
Die Methode get_list_of_cluster_vectors wird verwendet, um Clusterzuweisungen für verschiedene Clustering-Methoden zu berechnen und diese in einer Liste zu speichern. Diese Liste enthält die Clusterzuweisungen für eine gegebene Anzahl von Clustern k. Der Outcome wird verwendet, um die Silhouettenwerte für verschiedene Clustering-Methoden und Clusteranzahlen zu berechnen und zu vergleichen. Dies hilft dabei, die optimale Anzahl von Clustern für die gegebenen Daten zu identifizieren.

Die Ergebnisse deuten darauf hin, dass k=2 die beste Clusteranzahl zu sein scheint, da die meisten Methoden die höchsten Silhouettenwerte. Auch der WSS-Plot sowie der Silhouette-Plot lassen auf 2 als optimale Clusteranzahl schließen.

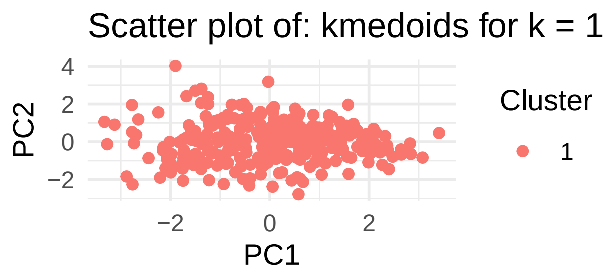
Grouped Scatter/Cluster Plots for 1 to n Clusters for Each (Suitable) Clustering Method

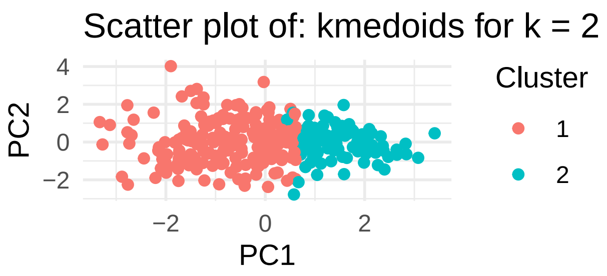
```r
load_source()
# Data Preparation

# Load the data
heart_disease_patients_tot <- read.csv("heart_disease_patients.csv")

# Remove uninformativ non-numeric features
```

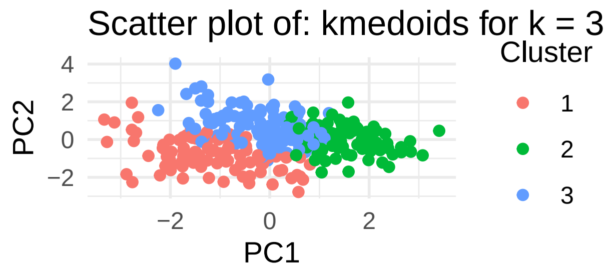
```r
heart_disease_patients <- subset(heart_disease_patients_tot, select= -c(id,sex,cp,fbs,restecg,exang,slop

# Handle missing data
heart_disease_patients <- na.omit(heart_disease_patients)

# Normalize the numerical features (assuming all features are numerical)
heart_disease_patients_scaled <- scale(heart_disease_patients, center = TRUE)

#-----------------------------------------------------------------------------------------
# Perform PCA

# Perform PCA on the scaled data
pca_result <- prcomp(heart_disease_patients_scaled)

# Get the data in the PC space
data <- pca_result$x

#-----------------------------------------------------------------------------------------
# Perform different clustering methods
n = 6 # max number of clusters

# Cluster vectors are generated by different clustering methods for a given k
# They are stored in a list which is generated for each k
# These lists are stored inside another list
lists_of_cluster_vectors <- list()

# Create a list of lists of cluster vectors for different k
for (k in 1:n) {
  lists_of_cluster_vectors[[k]] <- get_list_of_cluster_vectors(data, k) # for m methods of clustering a
  # Name each list of cluster vectors 'k'
  names(lists_of_cluster_vectors)[k] <- as.character(k)
}

# Cluster results are generated by different clustering methods for a given k
# They are stored in a list which is generated for each k
# These lists are stored inside another list
lists_of_cluster_results <- list()

# Create a list of lists of cluster results for different k
for (k in 1:n) {
  lists_of_cluster_results[[k]] <- get_list_of_cluster_results(data, k) # for m methods of clustering a
  # Name each list of cluster results 'k'
  names(lists_of_cluster_results)[k] <- as.character(k)
}

#-----------------------------------------------------------------------------------------
# Visualize the cluster vectors and results

#--- Methods ---#

create_grid_plot_vectors <- function(data, lists_of_cluster_vectors, i, n) {
  # Depiction 1
  plot_list <- list()
  for(k in 1:n) {
    # Generate the plot and store it in the list
    plot_list[[k]] <- create_scatter_plot_pca( data, lists_of_cluster_vectors[[k]][[i]], names(lists_of
  }

  # Combine all the plots into a single plot
  grid.arrange(grobs = plot_list, ncol = 2)
}

create_grid_plot_results <- function(data, lists_of_cluster_results, i, n) {
  # Depiction 2
  plot_list <- list()

  for(k in 2:n) {
    # Visualize kmeans clustering
    plot_list[[k]] <- fviz_cluster(lists_of_cluster_results[[k]][[i]], data = data[, c(1,2)], ellipse.ty
      theme_minimal() +
      ggtitle(paste("Cluster plot of:", names(lists_of_cluster_results[[k]])[i],"for k = ", names(lists
      xlab("PC1") +
      ylab("PC2")
  }

  # Combine all the plots into a single plot
  grid.arrange(grobs = plot_list, ncol = 2)
}

#--- End of Methods --#

# All cluster vectors of different clustering methods have the same length independently of k
m = length(lists_of_cluster_vectors[[1]])
# Loop over k lists of cluster vectors
# and pick the i-th cluster vector from that list
for( i in 1:m) {
  create_grid_plot_vectors(data,lists_of_cluster_vectors,i,n)
}
```

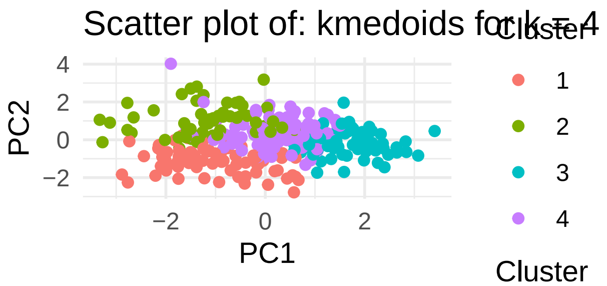


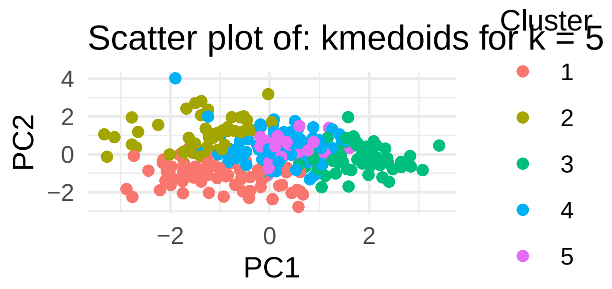
Scatter plot of: kmeans for k = 1
Scatter plot of: kmeans for k = 2
Scatter plot of: kmeans for k = 3
Scatter plot of: kmeans for k = 4
Scatter plot of: kmeans for k = 5
Scatter plot of: kmeans for k = 6
PC1
PC2
−2
0
2
4
Cluster
1
2
3
4
5
6

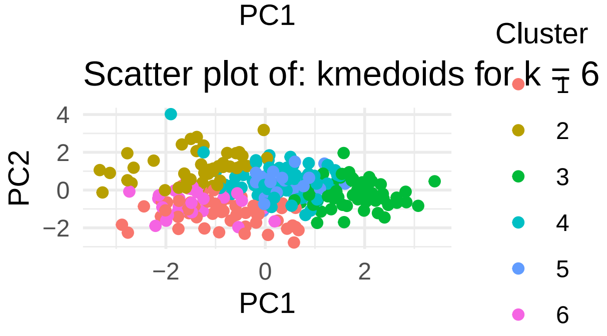

Scatter plot of: kmedoids for k = 1
PC2
PC1
Cluster
1

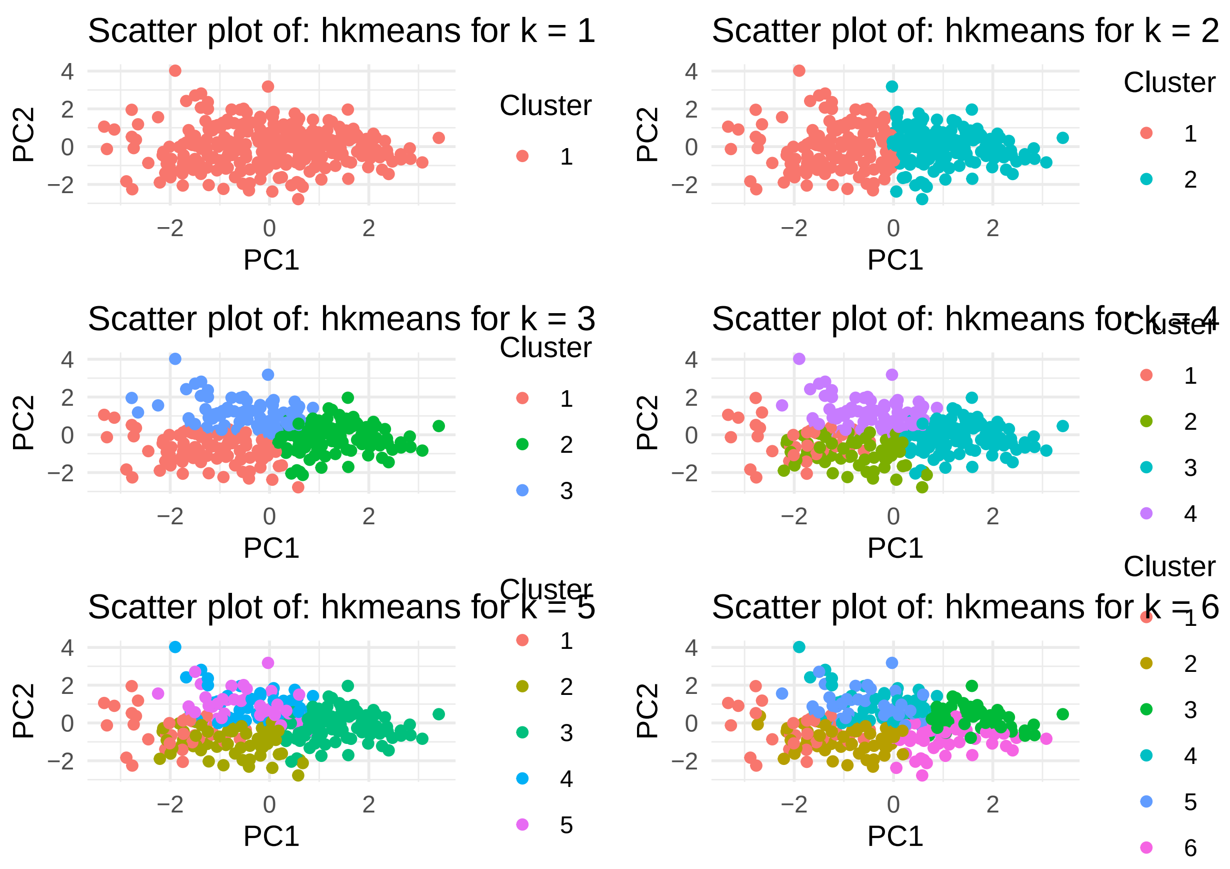

Scatter plot of: kmedoids for k = 2
PC2
PC1
Cluster
1
2

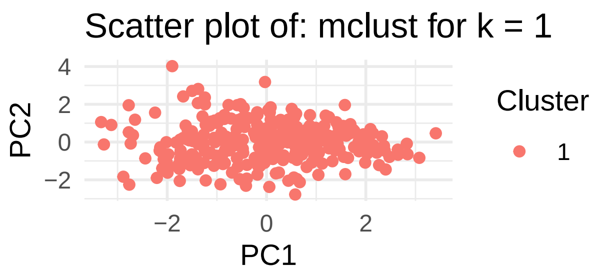

Scatter plot of: kmedoids for k = 3
PC2
PC1
Cluster
1
2
3

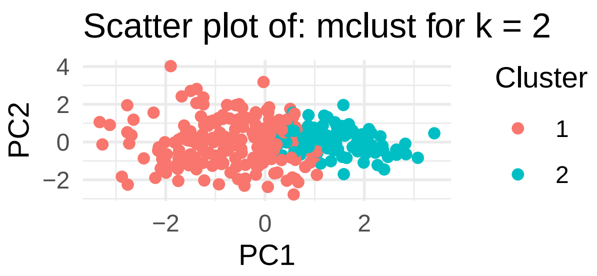

Scatter plot of: kmedoids for k = 4
PC2
PC1
Cluster
1
2
3
4

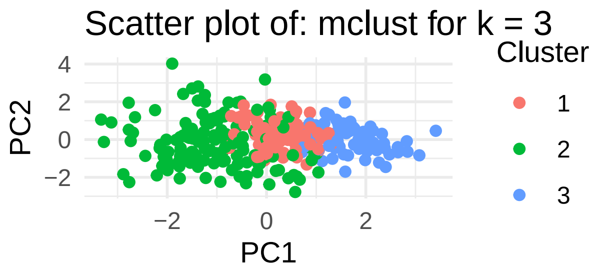

Scatter plot of: kmedoids for k = 5
PC2
PC1
Cluster
1
2
3
4
5

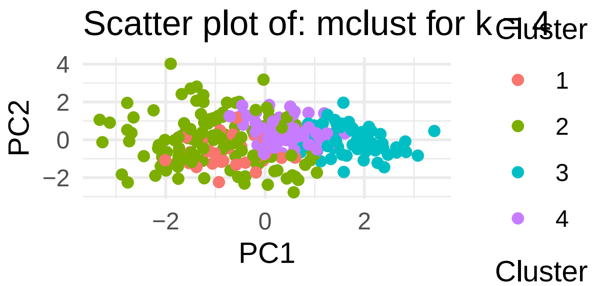

Scatter plot of: kmedoids for k = 6
PC2
PC1
Cluster
1
2
3
4
5
6

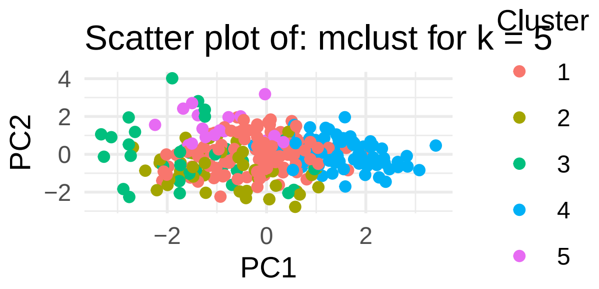

Scatter plot of: hkmeans for k = 1
Scatter plot of: hkmeans for k = 2
Scatter plot of: hkmeans for k = 3
Scatter plot of: hkmeans for k = 4
Scatter plot of: hkmeans for k = 5
Scatter plot of: hkmeans for k = 6
PC1
PC2
Cluster
1
2
3
4
5
6

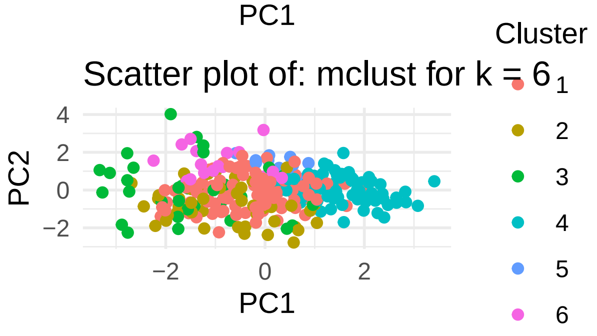

Scatter plot of: mclust for k = 1
PC2
PC1
Cluster
1

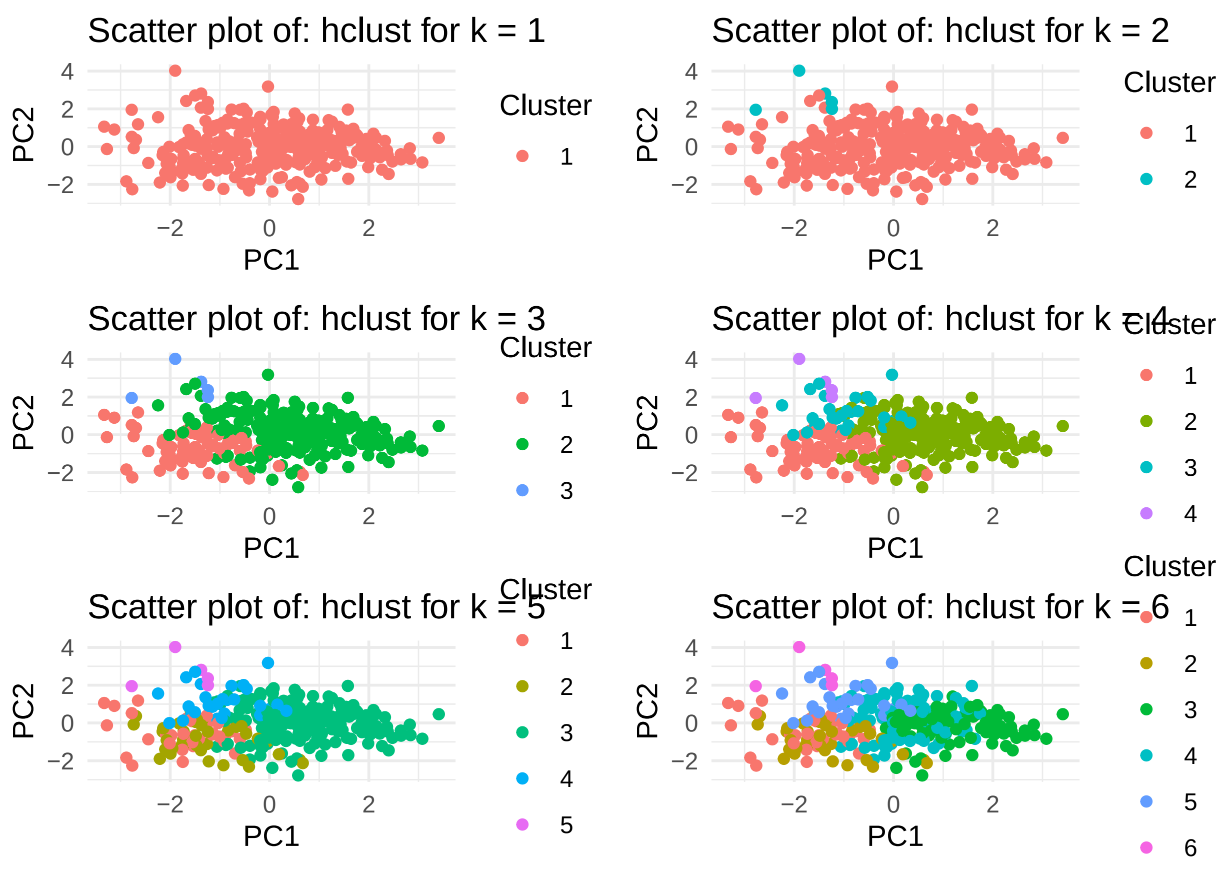

Scatter plot of: mclust for k = 2
PC2
PC1
Cluster
1
2


Scatter plot of: mclust for k = 3
PC2
PC1
Cluster
1
2
3

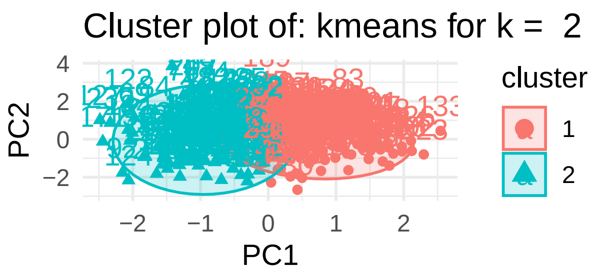

Scatter plot of: mclust for k = 4
PC2
PC1
Cluster
1
2
3
4

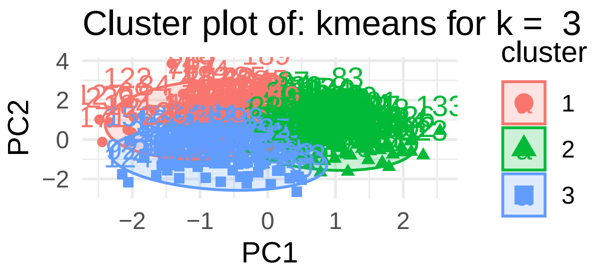

Scatter plot of: mclust for k = 5
PC2
PC1
Cluster
1
2
3
4
5

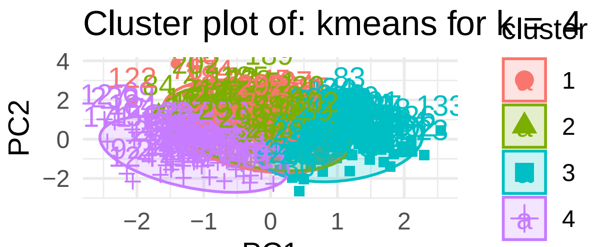

Scatter plot of: mclust for k = 6
PC2
PC1
Cluster
1
2
3
4
5
6

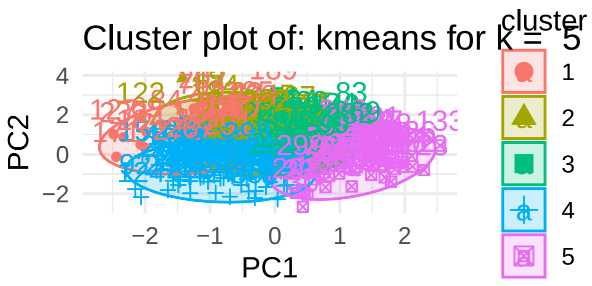


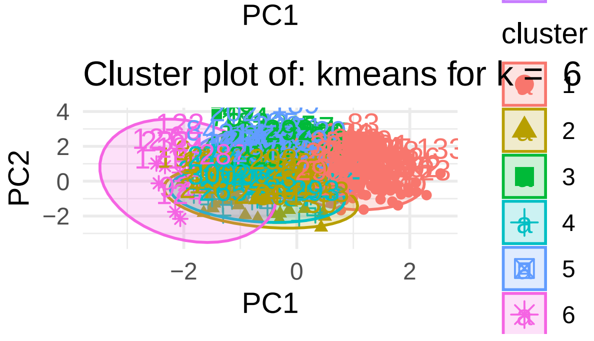
```r
for( i in 1:(m-1)) {
  create_grid_plot_results(data,lists_of_cluster_results,i,n)
}
```

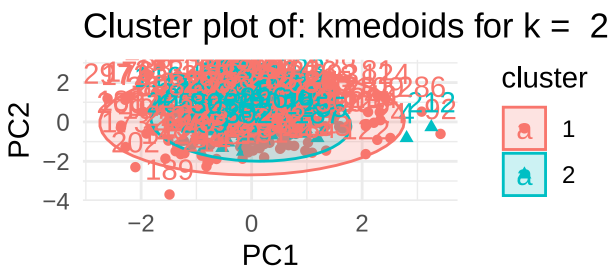

Cluster plot of: kmeans for k = 2
PC2
PC1
cluster
1
2

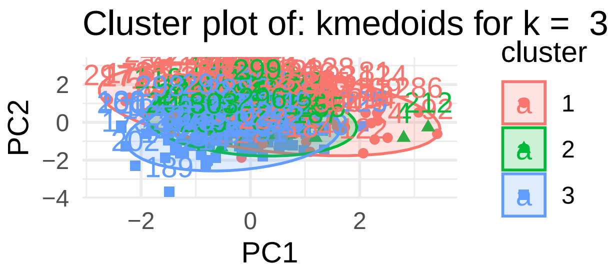

Cluster plot of: kmeans for k = 3
PC2
PC1
cluster
1
2
3

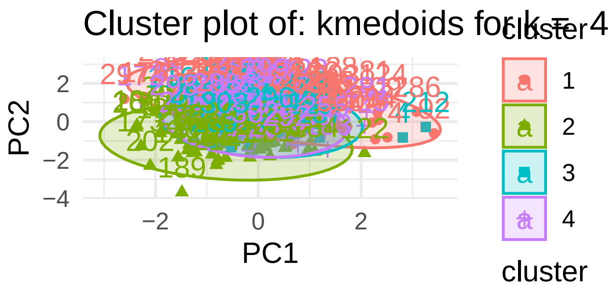

Cluster plot of: kmeans for k = 4
PC2
PC1
cluster
1
2
3
4

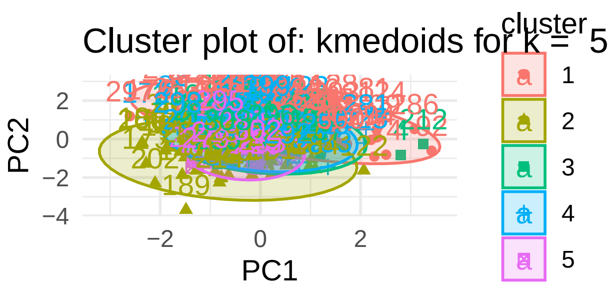

Cluster plot of: kmeans for k = 5
PC2
PC1
cluster
1
2
3
4
5

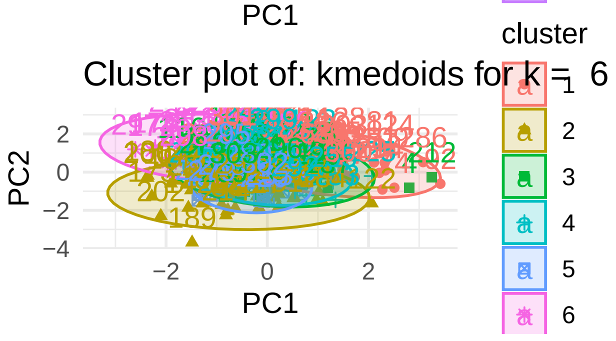

Cluster plot of: kmeans for k = 6
PC2
PC1
cluster
1
2
3
4
5
6

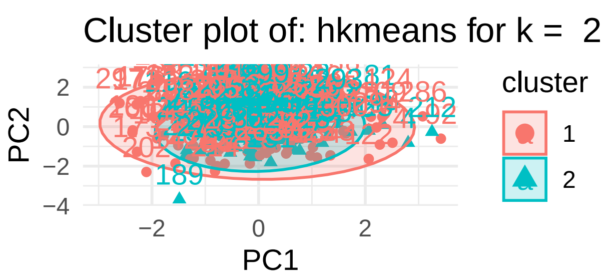

Cluster plot of: kmedoids for k = 2
PC2
PC1
cluster
1
2
189
202
286
212

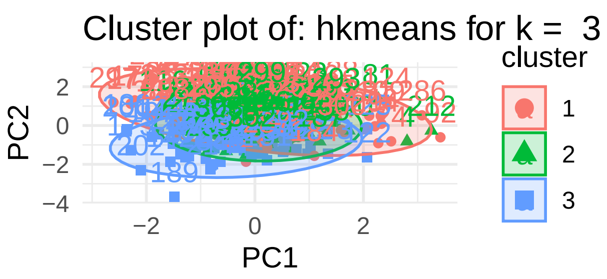

Cluster plot of: kmedoids for k = 3
PC2
PC1
cluster
1
2
3
189
202
299
286
212

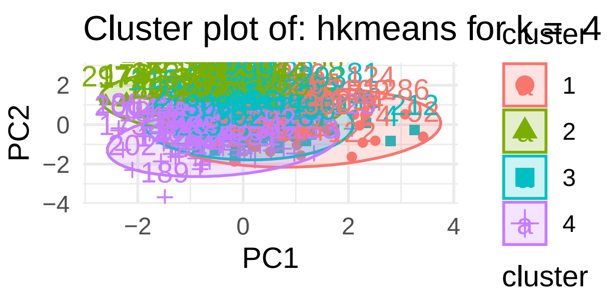

Cluster plot of: kmedoids for k = 4
PC2
PC1
cluster
1
2
3
4
189
202
286
212

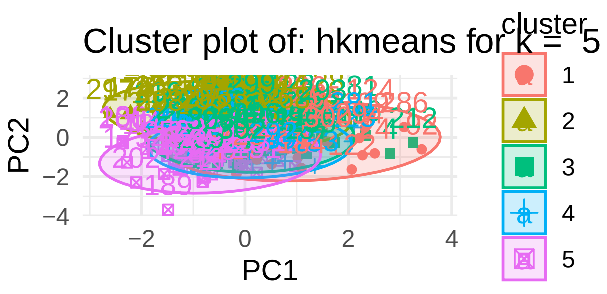

Cluster plot of: kmedoids for k = 5
PC2
PC1
cluster
1
2
3
4
5
189
202
286
212

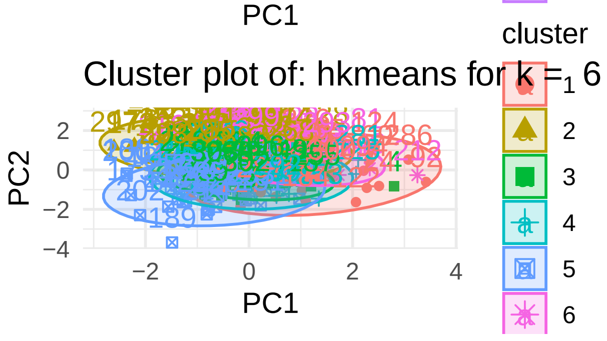

Cluster plot of: kmedoids for k = 6
PC2
PC1
cluster
1
2
3
4
5
6
189
202
286
212

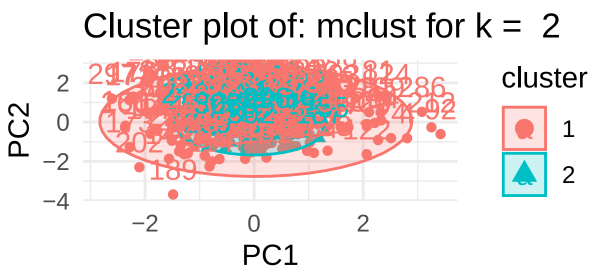

Cluster plot of: hkmeans for k = 2
PC2
PC1
cluster
1
2
189
202

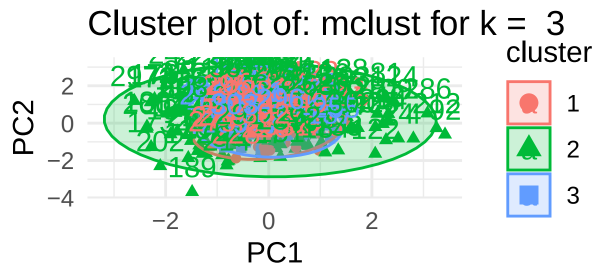

Cluster plot of: hkmeans for k = 3
PC2
PC1
cluster
1
2
3
189
202

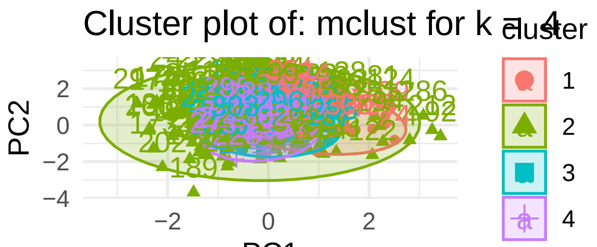

Cluster plot of: hkmeans for k = 4
PC2
PC1
cluster
1
2
3
4
189
202

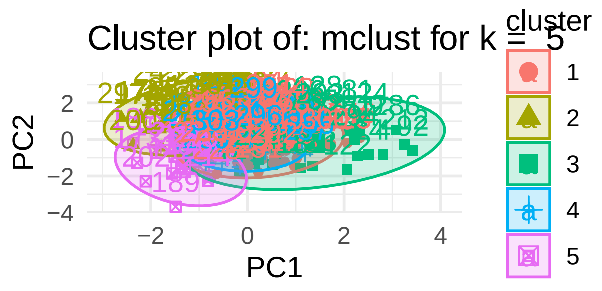

Cluster plot of: hkmeans for k = 5
PC2
PC1
cluster
1
2
3
4
5
189
202

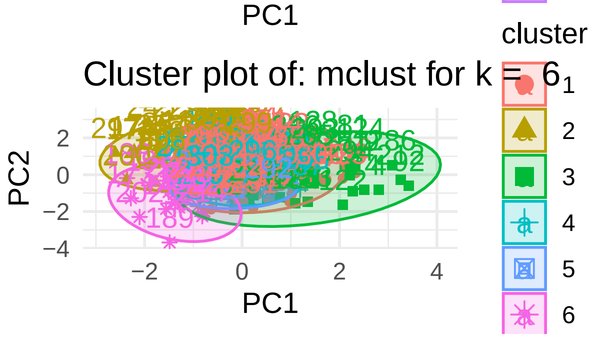

Cluster plot of: hkmeans for k = 6
PC2
PC1
cluster
1
2
3
4
5
6
189
202







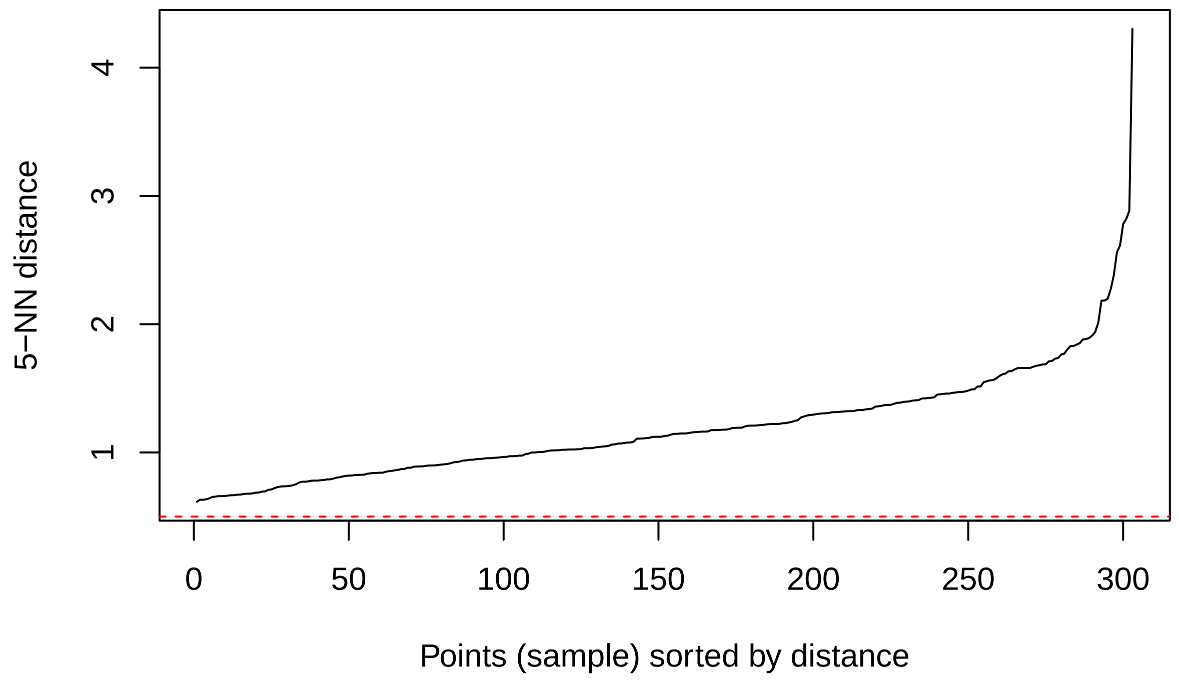




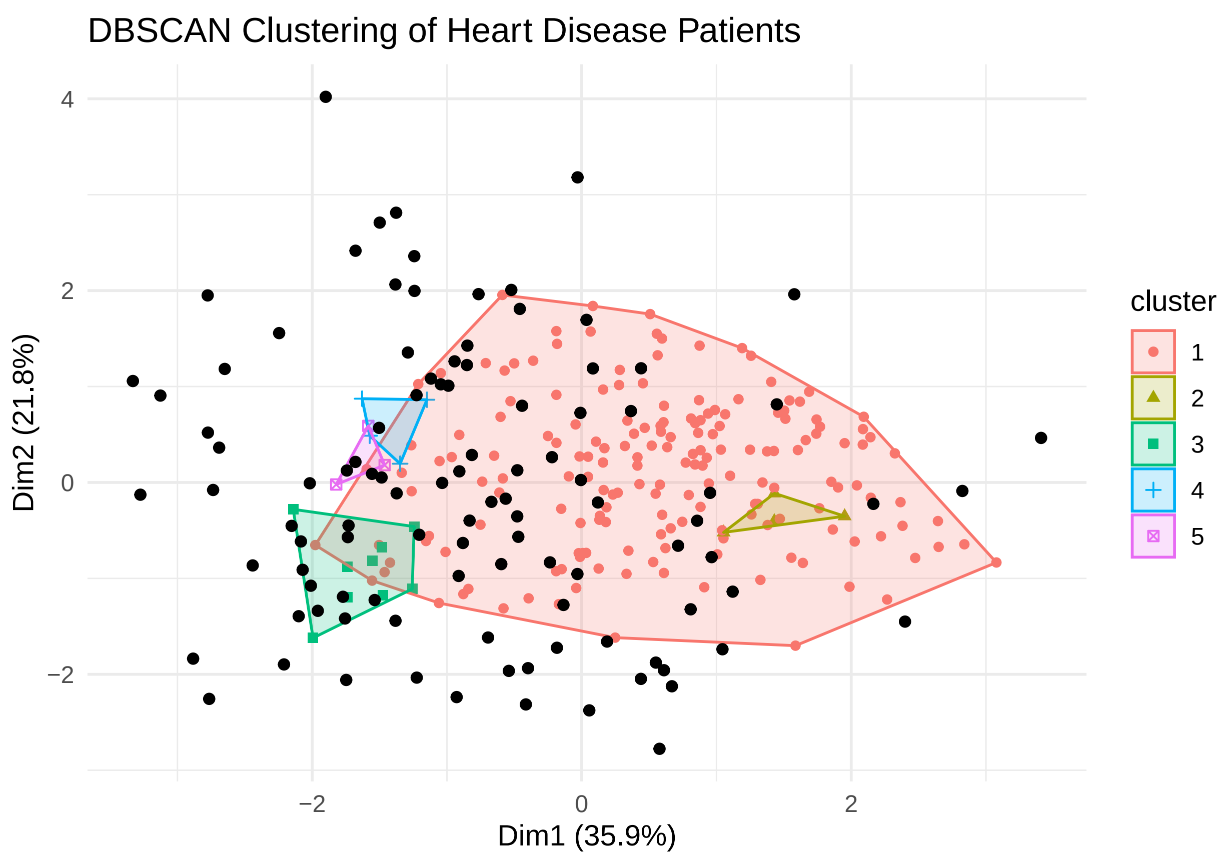
```
# DBSCAN Clustering

# Perform DBSCAN clustering
dbscan_result <- dbscan(data, eps = 1, minPts = 4)
dbscan_result$cluster
```

```
##   [1] 1 0 0 0 2 1 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0
##  [38] 4 0 1 0 2 0 1 1 1 0 0 0 1 1 1 1 1 1 3 1 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 3 1
##  [75] 1 0 1 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1
## [112] 1 1 0 3 1 1 2 5 1 1 0 0 0 1 1 0 0 1 1 1 1 0 1 1 1 0 3 0 1 1 0 1 5 1 1 4 1
## [149] 1 1 0 1 0 0 3 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 0 0
## [186] 1 1 4 0 1 1 0 1 3 0 0 0 1 1 4 1 0 0 0 1 0 1 0 1 1 1 0 2 0 1 1 1 1 5 1 1 1
## [223] 1 3 0 0 1 0 0 1 1 0 0 0 1 1 3 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 3 1 0 1 1 0 0
## [260] 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1
## [297] 0 0 1 0 0 1 0
```

```
# Parameteroptimierung für DBSCAN
kNNdistplot(data, k = 5)
abline(h = 0.5, col = "red", lty = 2)
```



```
# DBSCAN Cluster Visualisierung
fviz_cluster(dbscan_result, data = data, geom = "point", stand = FALSE, show.clust.cent = FALSE) +
  theme_minimal() +
  ggtitle("DBSCAN Clustering of Heart Disease Patients")
```



```
# Anzahl der Cluster und Rauschpunkte
cat("Anzahl der ermittelten cluster: ", length(unique(dbscan_result$cluster[dbscan_result$cluster > 0]))
```

```
## Anzahl der ermittelten cluster:  5
```

```
cat("Anzahl der als Rauschen erkannten Punkte: ", sum(dbscan_result$cluster == 0), "\n")
```

```
## Anzahl der als Rauschen erkannten Punkte:  104
```

```
# Clusterzuweisungen in Dataframe umwandeln
dbscan_clusters <- data.frame(Cluster = factor(dbscan_result$cluster))

# Originaldaten hinzufügen
dbscan_clusters <- cbind(dbscan_clusters, heart_disease_patients)

# Clusterzusammenfassung erstellen
cluster_summary <- aggregate(. ~ Cluster, data = dbscan_clusters, mean)
print(cluster_summary)
```

```
##   Cluster      age trestbps     chol  thalach  oldpeak
## 1       0 56.74038 138.9904 252.0288 140.2596 1.685577
## 2       1 52.89385 127.1285 241.7095 157.9050 0.624581
## 3       2 39.25000 134.5000 200.0000 175.0000 1.550000
## 4       3 59.88889 125.8889 271.6667 101.4444 1.611111
## 5       4 59.75000 158.7500 271.0000 120.2500 0.400000
## 6       5 63.66667 128.3333 314.0000 128.3333 1.866667
```

In diesem Abschnitt wird das DBSCAN-Clustering durchgeführt. Die Parameter wurden so gewählt, dass die Anzahl der Cluster als sinnvoll erscheinten und möglichst wenig Rauschen in den Daten erkannt wurde. Die Visualisierung zeigt, dass die Cluster mittels DBSCAN Verfahren nicht wirklich gut voneinander getrennt sind. Die Clusterzusammenfassung zeigt die Mittelwerte der Cluster für jede numerische Variable.