Project

Group 33

Preparing the data

```
columnNames <- c("id","diagnosis","radius_mean","texture_mean","perimeter_mean",
                 "area_mean","smoothness_mean","compactness_mean","concavity_mean",
                 "concave_points_mean","symmetry_mean","fractal_dimension_mean",
                 "radius_se","texture_se","perimeter_se","area_se","smoothness_se",
                 "compactness_se","concavity_se","concave_points_se","symmetry_se",
                 "fractal_dimension_se","radius_worst","texture_worst","perimeter_worst",
                 "area_worst","smoothness_worst","compactness_worst","concavity_worst",
                 "concave_points_worst","symmetry_worst","fractal_dimension_worst")
```

```
wisc.df <- fread("wdbc.data", col.names = columnNames)
head(wisc.df)
```

```
##          id diagnosis radius_mean texture_mean perimeter_mean area_mean
##       <int>    <char>       <num>        <num>          <num>     <num>
## 1:   842302         M       17.99        10.38         122.80    1001.0
## 2:   842517         M       20.57        17.77         132.90    1326.0
## 3: 84300903         M       19.69        21.25         130.00    1203.0
## 4: 84348301         M       11.42        20.38          77.58     386.1
## 5: 84358402         M       20.29        14.34         135.10    1297.0
## 6:   843786         M       12.45        15.70          82.57     477.1
##    smoothness_mean compactness_mean concavity_mean concave_points_mean
##              <num>            <num>          <num>               <num>
## 1:         0.11840          0.27760         0.3001             0.14710
## 2:         0.08474          0.07864         0.0869             0.07017
## 3:         0.10960          0.15990         0.1974             0.12790
## 4:         0.14250          0.28390         0.2414             0.10520
## 5:         0.10030          0.13280         0.1980             0.10430
## 6:         0.12780          0.17000         0.1578             0.08089
##    symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
##            <num>                 <num>     <num>      <num>        <num>
## 1:        0.2419               0.07871    1.0950     0.9053        8.589
## 2:        0.1812               0.05667    0.5435     0.7339        3.398
## 3:        0.2069               0.05999    0.7456     0.7869        4.585
## 4:        0.2597               0.09744    0.4956     1.1560        3.445
## 5:        0.1809               0.05883    0.7572     0.7813        5.438
## 6:        0.2087               0.07613    0.3345     0.8902        2.217
##    area_se smoothness_se compactness_se concavity_se concave_points_se
##      <num>         <num>          <num>        <num>             <num>
## 1:  153.40      0.006399        0.04904      0.05373           0.01587
```

```
## 2:   74.08      0.005225        0.01308      0.01860           0.01340
## 3:   94.03      0.006150        0.04006      0.03832           0.02058
## 4:   27.23      0.009110        0.07458      0.05661           0.01867
## 5:   94.44      0.011490        0.02461      0.05688           0.01885
## 6:   27.19      0.007510        0.03345      0.03672           0.01137
##    symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
##          <num>                <num>        <num>         <num>           <num>
## 1:     0.03003             0.006193        25.38         17.33          184.60
## 2:     0.01389             0.003532        24.99         23.41          158.80
## 3:     0.02250             0.004571        23.57         25.53          152.50
## 4:     0.05963             0.009208        14.91         26.50           98.87
## 5:     0.01756             0.005115        22.54         16.67          152.20
## 6:     0.02165             0.005082        15.47         23.75          103.40
##    area_worst smoothness_worst compactness_worst concavity_worst
##         <num>            <num>             <num>           <num>
## 1:     2019.0           0.1622            0.6656          0.7119
## 2:     1956.0           0.1238            0.1866          0.2416
## 3:     1709.0           0.1444            0.4245          0.4504
## 4:      567.7           0.2098            0.8663          0.6869
## 5:     1575.0           0.1374            0.2050          0.4000
## 6:      741.6           0.1791            0.5249          0.5355
##    concave_points_worst symmetry_worst fractal_dimension_worst
##                   <num>          <num>                  <num>
## 1:               0.2654         0.4601                0.11890
## 2:               0.1860         0.2750                0.08902
## 3:               0.2430         0.3613                0.08758
## 4:               0.2575         0.6638                0.17300
## 5:               0.1625         0.2364                0.07678
## 6:               0.1741         0.3985                0.12440
```

Check for missing data

```
# Sum of NA values in each column
colSums(is.na(wisc.df))
```

```
##                     id              diagnosis            radius_mean
##                      0                      0                      0
##           texture_mean         perimeter_mean              area_mean
##                      0                      0                      0
##        smoothness_mean        compactness_mean          concavity_mean
##                      0                      0                      0
##     concave_points_mean          symmetry_mean  fractal_dimension_mean
##                      0                      0                      0
##              radius_se             texture_se           perimeter_se
##                      0                      0                      0
##                area_se          smoothness_se         compactness_se
##                      0                      0                      0
##           concavity_se       concave_points_se            symmetry_se
##                      0                      0                      0
##    fractal_dimension_se            radius_worst           texture_worst
##                      0                      0                      0
```

```
##         perimeter_worst              area_worst        smoothness_worst
##                       0                       0                       0
##       compactness_worst         concavity_worst    concave_points_worst
##                       0                       0                       0
##          symmetry_worst fractal_dimension_worst
##                       0                       0
```

Exploratory Data Analysis

Our response variable is diagnosis: Benign (B) or Malignant (M). We have 3 sets of 10 numeric variables: mean, se, worst Let's first collect all the 30 numeric variables into a matrix.

```
# Convert the features of the data: wisc.data
wisc.data <- as.matrix(wisc.df[, .SD, .SDcols = 3:32])  # Selects only the feature columns
```

```
# Set the row names of wisc.data
row.names(wisc.data) <- wisc.df$id
```

```
# Create diagnosis vector
diagnosis <- as.numeric(wisc.df$diagnosis == "M")
```

Number of observations in this dataset.

```
nrow(wisc.data)
```

```
## [1] 569
```

Number of features in the data are suffixed with _mean, _se, _worst?

```
sum(endsWith(colnames(wisc.data), "_mean"))
```

```
## [1] 10
```

```
sum(endsWith(colnames(wisc.data), "_se"))
```

```
## [1] 10
```

```
sum(endsWith(colnames(wisc.data), "_worst"))
```

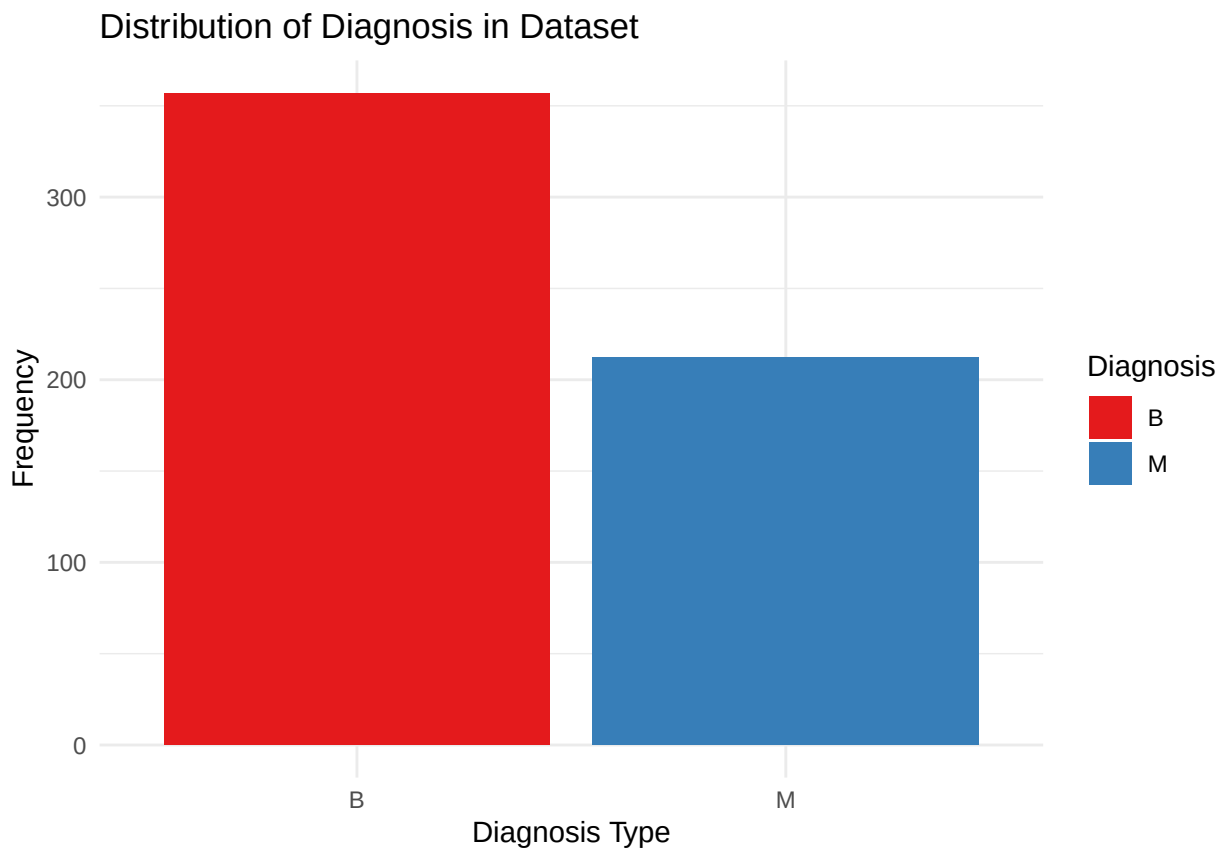
```
## [1] 10
```

Number of observations have benign or malignant diagnosis.

```r
library(ggplot2)
# Create a frequency table of the diagnosis
diagnosis_count <- table(wisc.df$diagnosis)
diagnosis_count
```

```
##
##   B   M
## 357 212
```

```r
# Convert the table to a data frame for ggplot2
diagnosis_df <- as.data.frame(diagnosis_count)
names(diagnosis_df) <- c("Diagnosis", "Count")

# Create a bar plot using ggplot2
ggplot(diagnosis_df, aes(x = Diagnosis, y = Count, fill = Diagnosis)) +
  geom_bar(stat = "identity") +  # Use identity to use the actual counts
  theme_minimal() +  # Use a minimal theme for a cleaner look
  labs(x = "Diagnosis Type", y = "Frequency", title = "Distribution of Diagnosis in Dataset") +
  scale_fill_brewer(palette = "Set1")  # Color palette for visual distinction
```



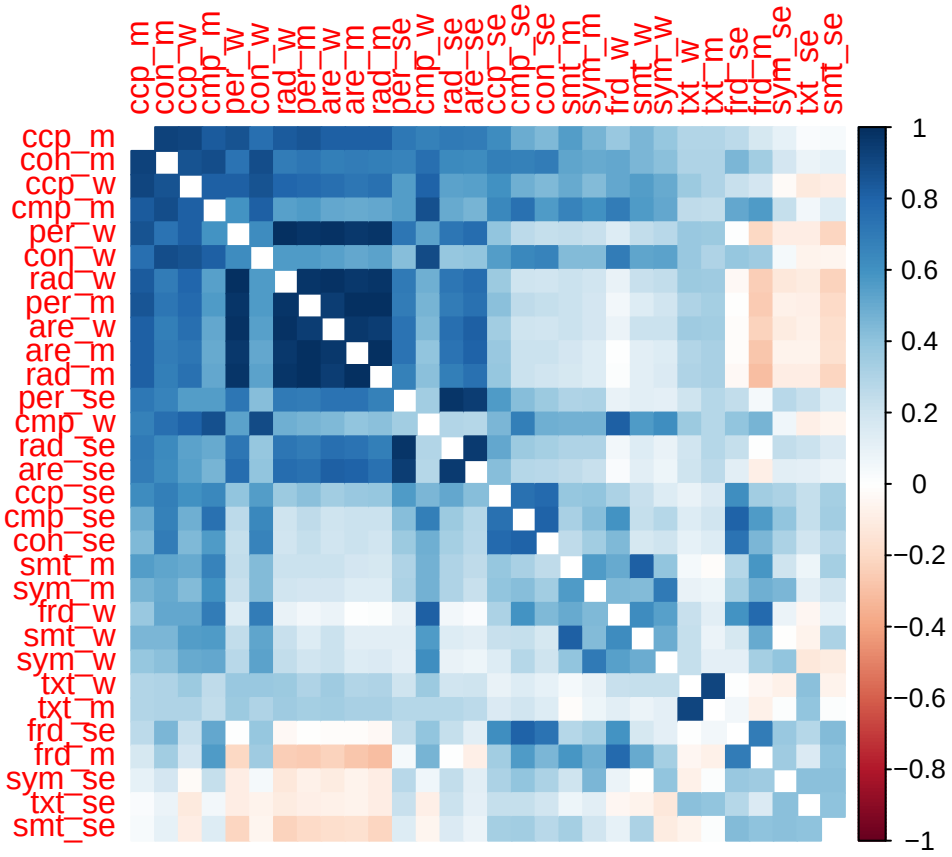
Correlation

```
corMatrix <- wisc.df[,c(3:32)]

# Rename the colnames
cNames <- c("rad_m","txt_m","per_m",
                  "are_m","smt_m","cmp_m","con_m",
                  "ccp_m","sym_m","frd_m",
                  "rad_se","txt_se","per_se","are_se","smt_se",
                  "cmp_se","con_se","ccp_se","sym_se",
                  "frd_se","rad_w","txt_w","per_w",
                  "are_w","smt_w","cmp_w","con_w",
                  "ccp_w","sym_w","frd_w")

colnames(corMatrix) <- cNames

# Create the correlation matrix
M <- round(cor(corMatrix), 2)

# Create corrplot
library(corrplot)
```

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

```
## corrplot 0.92 loaded
```

```
corrplot(M, diag = FALSE, method="color", order="FPC", tl.srt = 90)
```



Performing PCA

Check column means and standard deviations

```
colMeans(wisc.data)
```

```
##             radius_mean            texture_mean          perimeter_mean
##            1.412729e+01            1.928965e+01            9.196903e+01
##               area_mean         smoothness_mean        compactness_mean
##            6.548891e+02            9.636028e-02            1.043410e-01
##          concavity_mean     concave_points_mean          symmetry_mean
##            8.879932e-02            4.891915e-02            1.811619e-01
##  fractal_dimension_mean               radius_se              texture_se
##            6.279761e-02            4.051721e-01            1.216853e+00
##            perimeter_se                 area_se         smoothness_se
##            2.866059e+00            4.033708e+01            7.040979e-03
##          compactness_se           concavity_se      concave_points_se
##            2.547814e-02            3.189372e-02            1.179614e-02
##             symmetry_se    fractal_dimension_se            radius_worst
##            2.054230e-02            3.794904e-03            1.626919e+01
##           texture_worst         perimeter_worst              area_worst
##            2.567722e+01            1.072612e+02            8.805831e+02
```

```
##        smoothness_worst       compactness_worst         concavity_worst
##            1.323686e-01            2.542650e-01            2.721885e-01
##    concave_points_worst          symmetry_worst fractal_dimension_worst
##            1.146062e-01            2.900756e-01            8.394582e-02
```

```
apply(wisc.data, 2, sd)
```

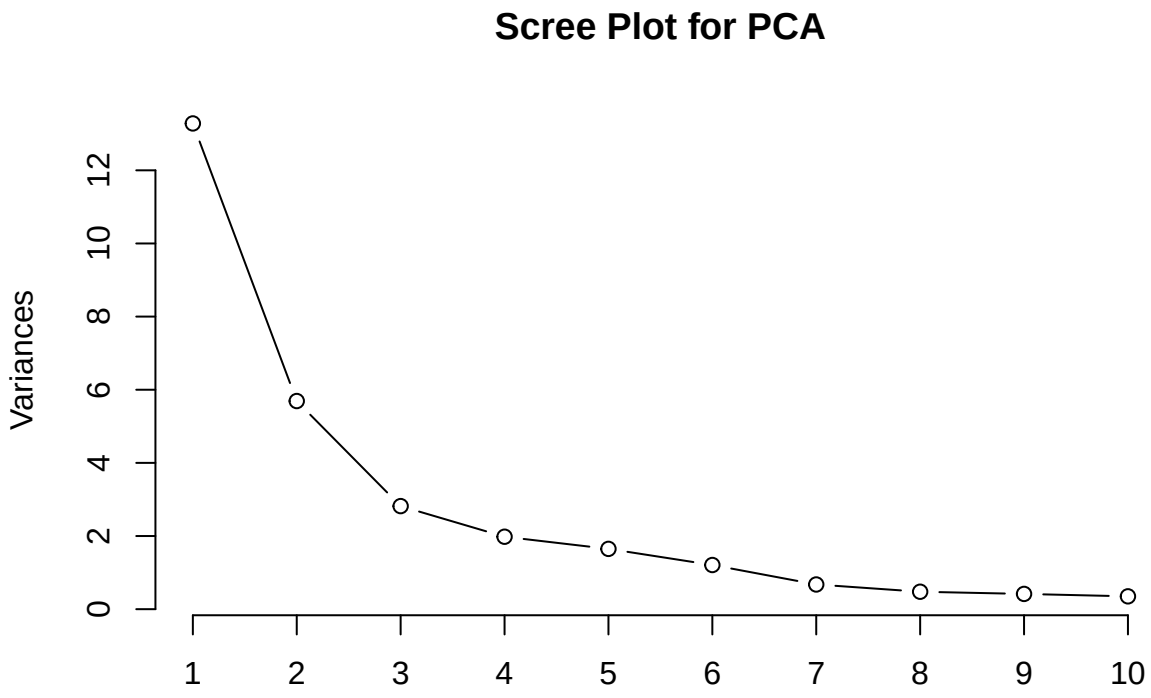
```
##             radius_mean            texture_mean          perimeter_mean
##            3.524049e+00            4.301036e+00            2.429898e+01
##               area_mean         smoothness_mean        compactness_mean
##            3.519141e+02            1.406413e-02            5.281276e-02
##          concavity_mean     concave_points_mean           symmetry_mean
##            7.971981e-02            3.880284e-02            2.741428e-02
##  fractal_dimension_mean               radius_se              texture_se
##            7.060363e-03            2.773127e-01            5.516484e-01
##            perimeter_se                 area_se           smoothness_se
##            2.021855e+00            4.549101e+01            3.002518e-03
##          compactness_se            concavity_se       concave_points_se
##            1.790818e-02            3.018606e-02            6.170285e-03
##             symmetry_se    fractal_dimension_se            radius_worst
##            8.266372e-03            2.646071e-03            4.833242e+00
##           texture_worst         perimeter_worst              area_worst
##            6.146258e+00            3.360254e+01            5.693570e+02
##        smoothness_worst       compactness_worst         concavity_worst
##            2.283243e-02            1.573365e-01            2.086243e-01
##    concave_points_worst          symmetry_worst fractal_dimension_worst
##            6.573234e-02            6.186747e-02            1.806127e-02
```

```
# Execute PCA, scaling if appropriate: wisc.pr
wisc.pr <- prcomp(x = wisc.data, scale = TRUE)

# Look at summary of results
summary(wisc.pr)
```

```
## 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
```

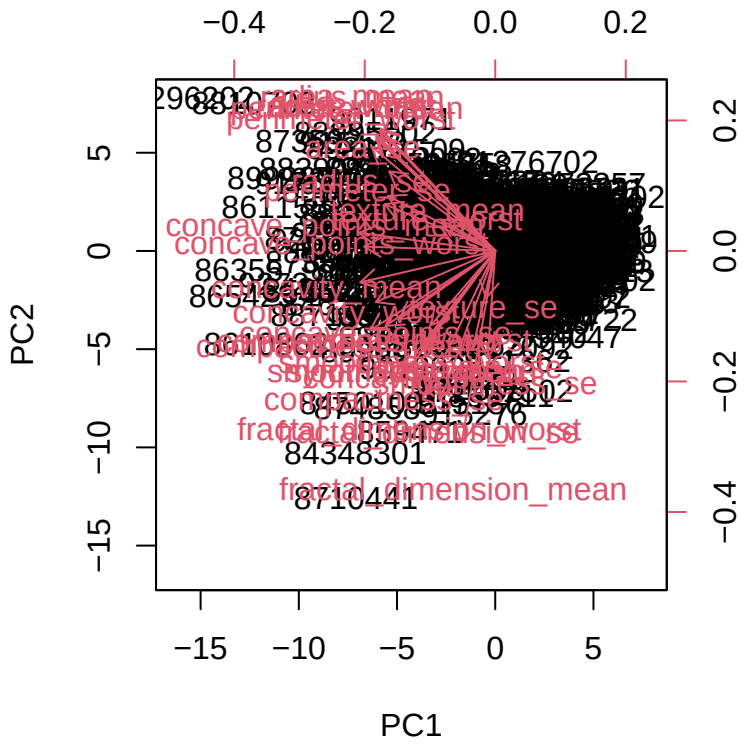
```
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

```r
screeplot(wisc.pr, type = "line", main = "Scree Plot for PCA")
```

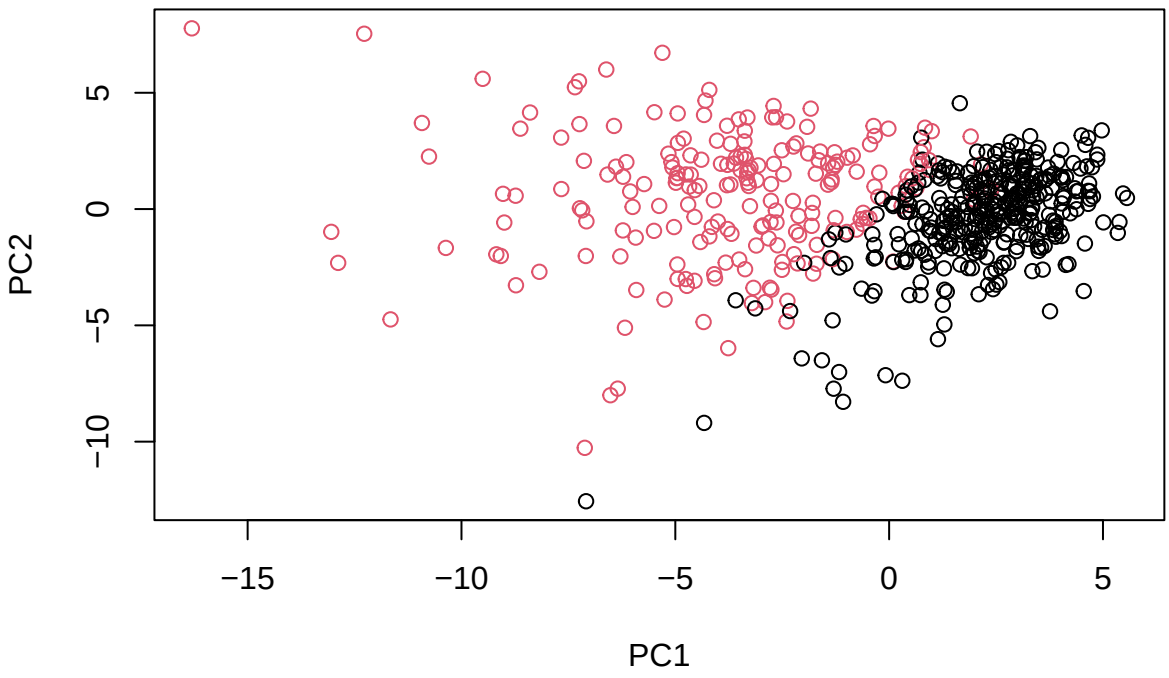


Interpreting PCA results

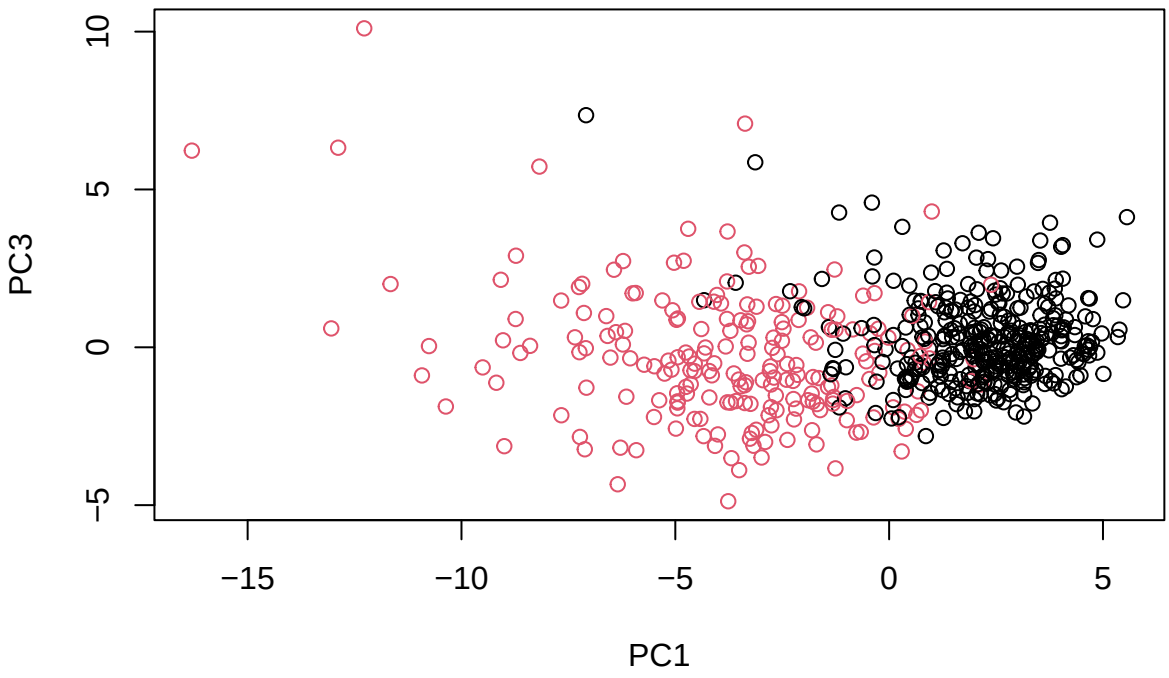
```r
# Create a biplot of wisc.pr
biplot(wisc.pr, scale = 0)
```



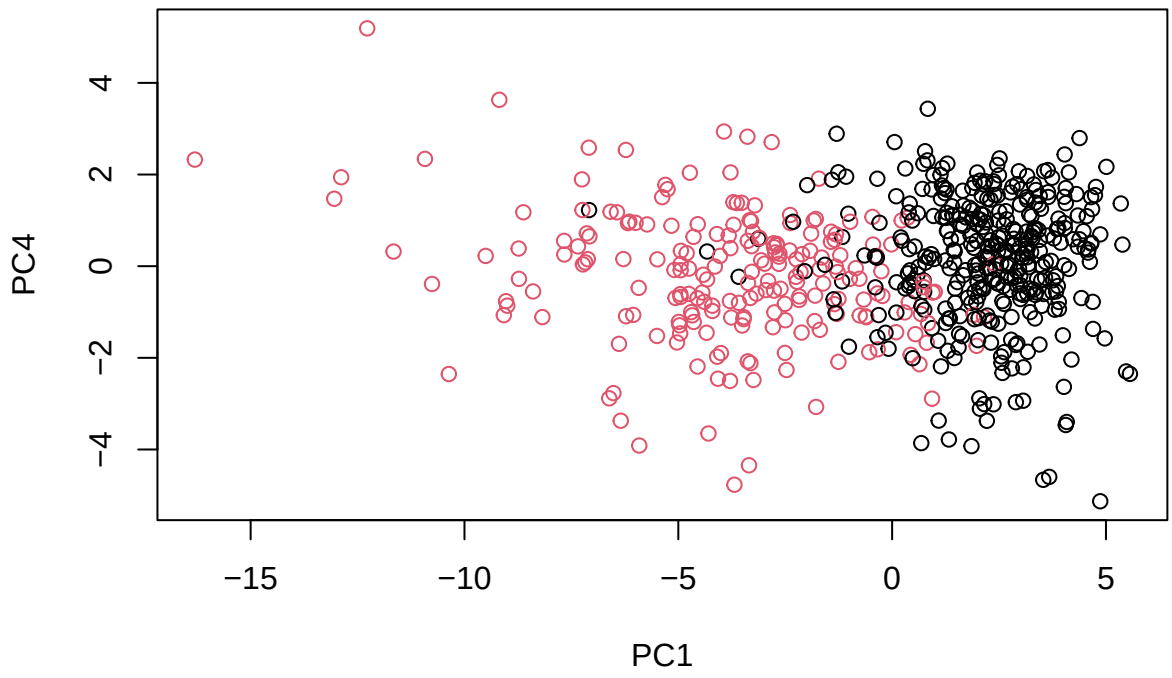

```
#Scatter plot observations by components 1 and 2
plot(wisc.pr$x[, c(1, 2)], col = (diagnosis+1),
     xlab = "PC1", ylab = "PC2")
```




```
# Repeat for components 1 and 3
plot(wisc.pr$x[, c(1, 3)], col = (diagnosis + 1),
     xlab = "PC1", ylab = "PC3")
```

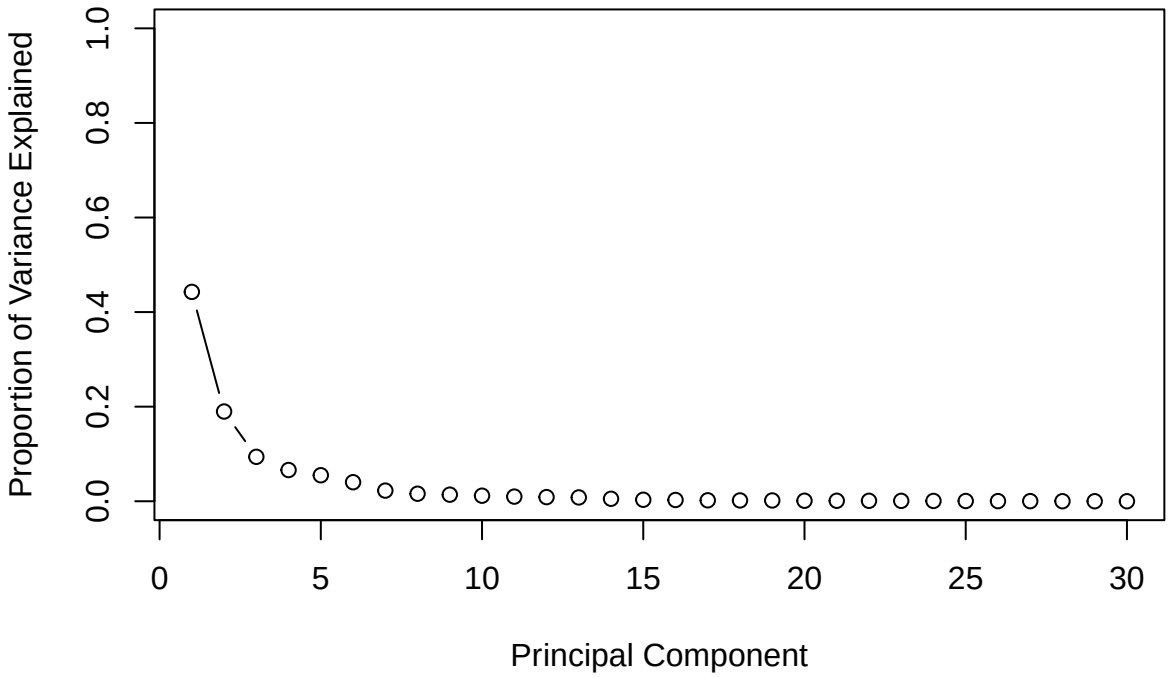



```
plot(wisc.pr$x[, c(1, 4)], col = (diagnosis + 1),
     xlab = "PC1", ylab = "PC4")
```

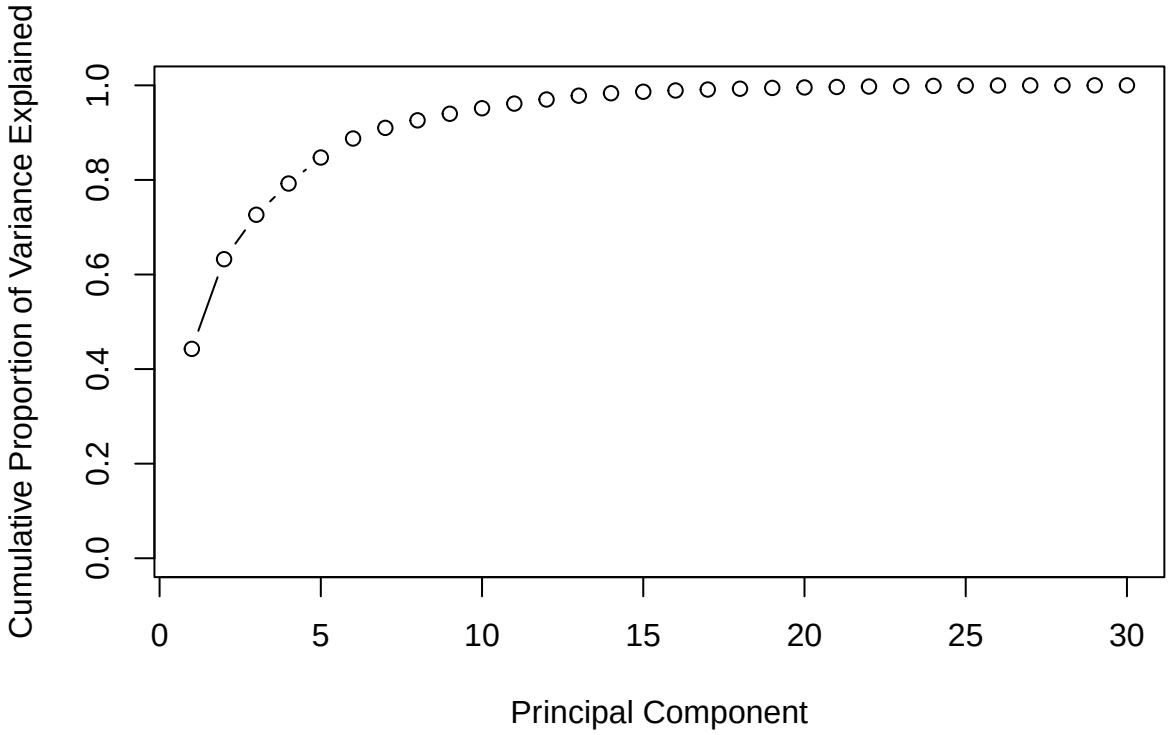


Variance explained

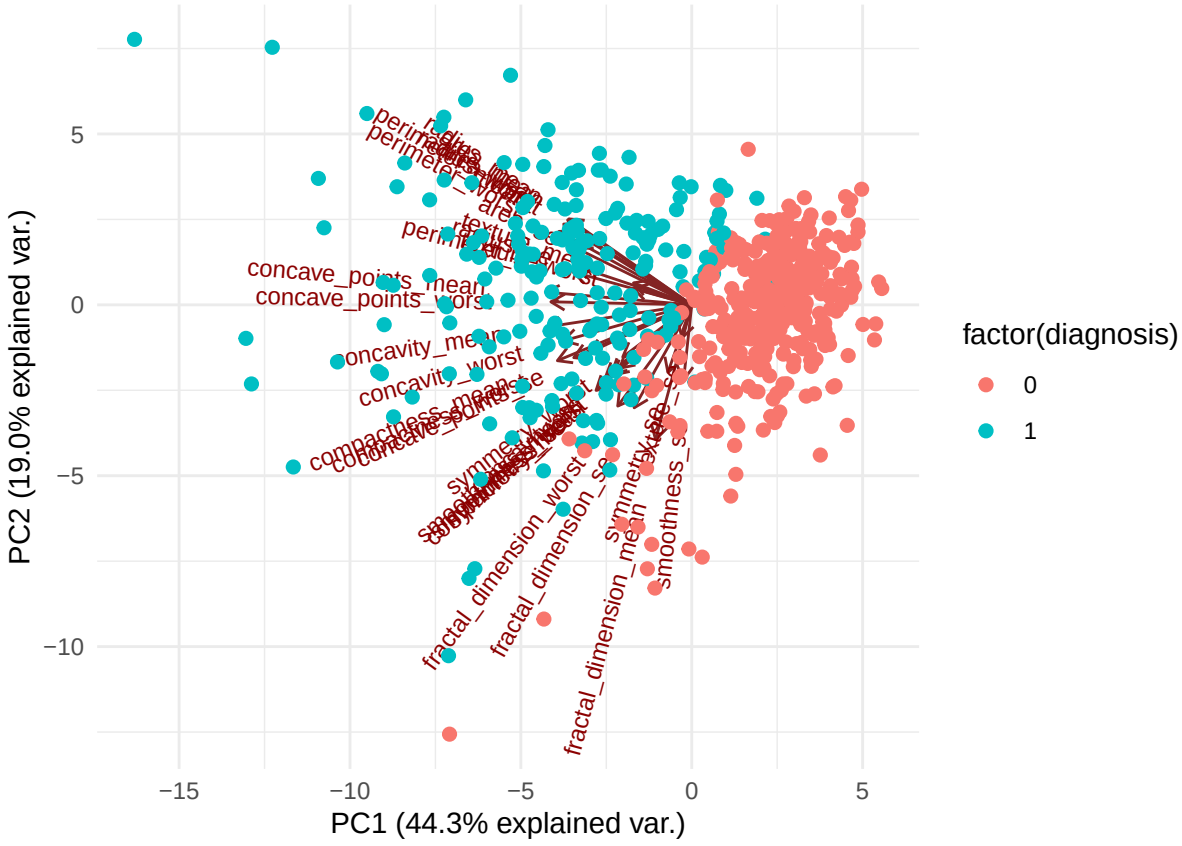
```
# Calculate variability of each component
pr.var <- wisc.pr$sdev^2

# Variance explained by each principal component: pve
pve <- pr.var / sum(pr.var)

# Plot variance explained for each principal component
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0, 1), type = "b")
```




```
# Plot cumulative proportion of variance explained
plot(cumsum(pve), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     ylim = c(0, 1), type = "b")
```




```r
# Extract principal components
PC1 <- wisc.pr$x[, 1]
PC2 <- wisc.pr$x[, 2]

# Create the biplot
biplot <- ggbiplot(wisc.pr,
                   obs.scale = 1,
                   var.scale = 1) +
          theme_minimal()

# Add point layer with colors based on smoking status
biplot + geom_point(aes(x = PC1, y = PC2, color = factor(diagnosis)), size = 2)
```




```
# Convert PCA results to a data frame
pca_df <- as.data.frame(wisc.pr$x)
```

Clustering

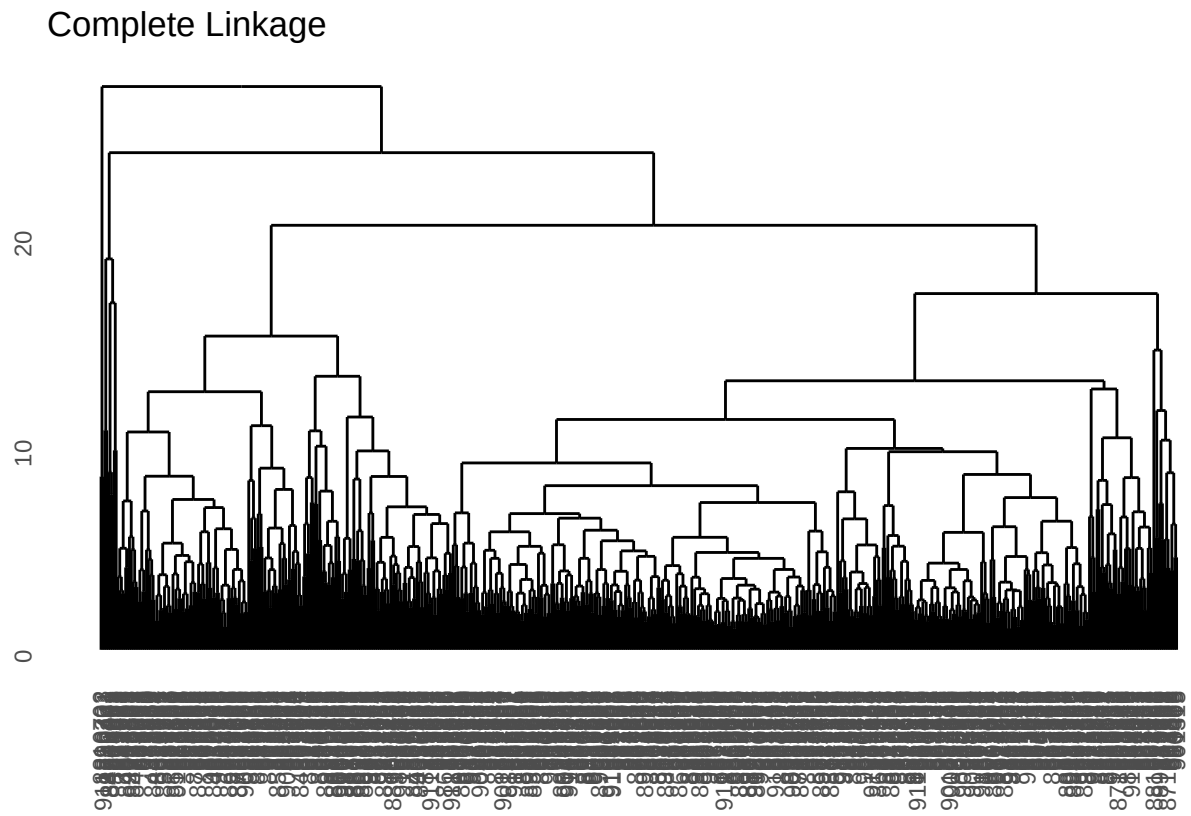
Hierarchical clustering

```
# Scale the wisc.data data: data.scaled
data.scaled <- scale(wisc.data)


# Calculate the (Euclidean) distances: data.dist
data.dist <- dist(data.scaled)

# Create a hierarchical clustering model: wisc.hclust
wisc.hclust = hclust(data.dist, method = "complete")
```

```
library(ggdendro)
```

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

```
ggdendrogram(wisc.hclust, segements=TRUE, labels=TRUE, leaf_labels = TRUE, rotate=FALSE, theme_dendro =
 labs(title='Complete Linkage')
```




```
# Cut tree so that it has 4 clusters: wisc.hclust.clusters
wisc.hclust.clusters <- cutree(wisc.hclust, k = 4)

# Compare cluster membership to actual diagnoses
table(wisc.hclust.clusters, diagnosis)
```

```
##                     diagnosis
## wisc.hclust.clusters   0   1
##                    1  12 165
##                    2   2   5
##                    3 343  40
##                    4   0   2
```

```
# Create a matrix to represent the updated counts from the clustering results
# Rows represent clusters, columns represent actual diagnosis
updated_matrix <- matrix(c(12, 165, 2, 5, 343, 40, 0, 2),
                         nrow = 4, byrow = TRUE,
                         dimnames = list(c("Cluster 1", "Cluster 2", "Cluster 3", "Cluster 4"),
                                         c("Benign", "Malignant")))

# Calculate the total number of correct predictions
```

```r
# Assuming Cluster 1, 2, and 4 are meant to predict malignant, and Cluster 3 predicts benign
correct_predictions <- updated_matrix[1, 2] +  # Malignant correctly predicted in Cluster 1
                       updated_matrix[2, 2] +  # Malignant correctly predicted in Cluster 2
                       updated_matrix[3, 1] +  # Benign correctly predicted in Cluster 3
                       updated_matrix[4, 2]    # Malignant correctly predicted in Cluster 4

# Calculate total predictions
total_predictions <- sum(updated_matrix)

# Calculate accuracy
accuracy <- correct_predictions / total_predictions

# Print the accuracy
print(accuracy)
```

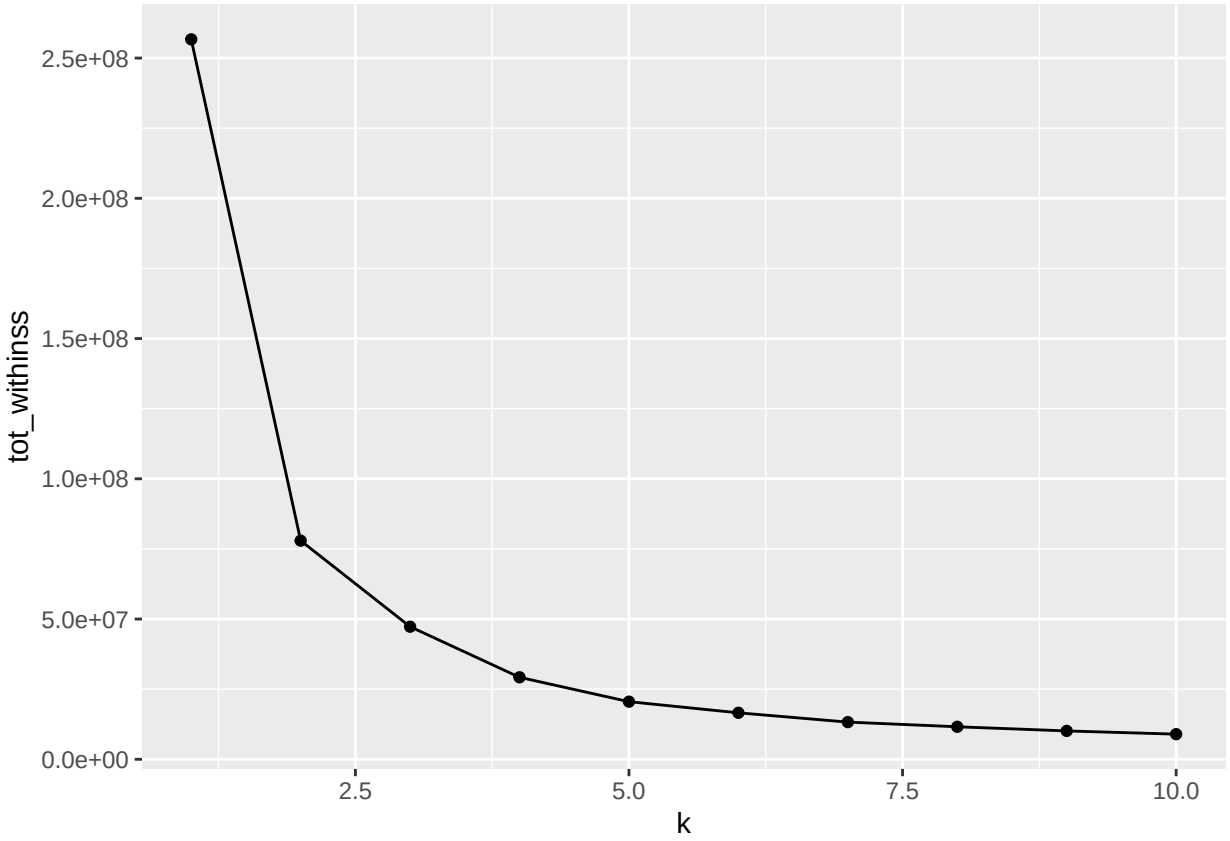
```
## [1] 0.9050967
```

K-Means Clustering

```r
set.seed(143)
km_out_list <- lapply(1:10, function(k) list(
  k=k,
  km_out=kmeans(wisc.data, k, nstart = 20)))

km_results <- data.frame(
  k=sapply(km_out_list, function(k) k$k),
  totss=sapply(km_out_list, function(k) k$km_out$totss),
  tot_withinss=sapply(km_out_list, function(k) k$km_out$tot.withinss)
  )
km_results
```

```
##     k     totss tot_withinss
## 1   1 256677244    256677244
## 2   2 256677244     77943100
## 3   3 256677244     47264842
## 4   4 256677244     29226542
## 5   5 256677244     20535170
## 6   6 256677244     16562262
## 7   7 256677244     13247263
## 8   8 256677244     11582683
## 9   9 256677244     10107786
## 10 10 256677244      8951683
```

```r
ggplot(km_results,aes(x=k,y=tot_withinss))+geom_line()+geom_point()
```

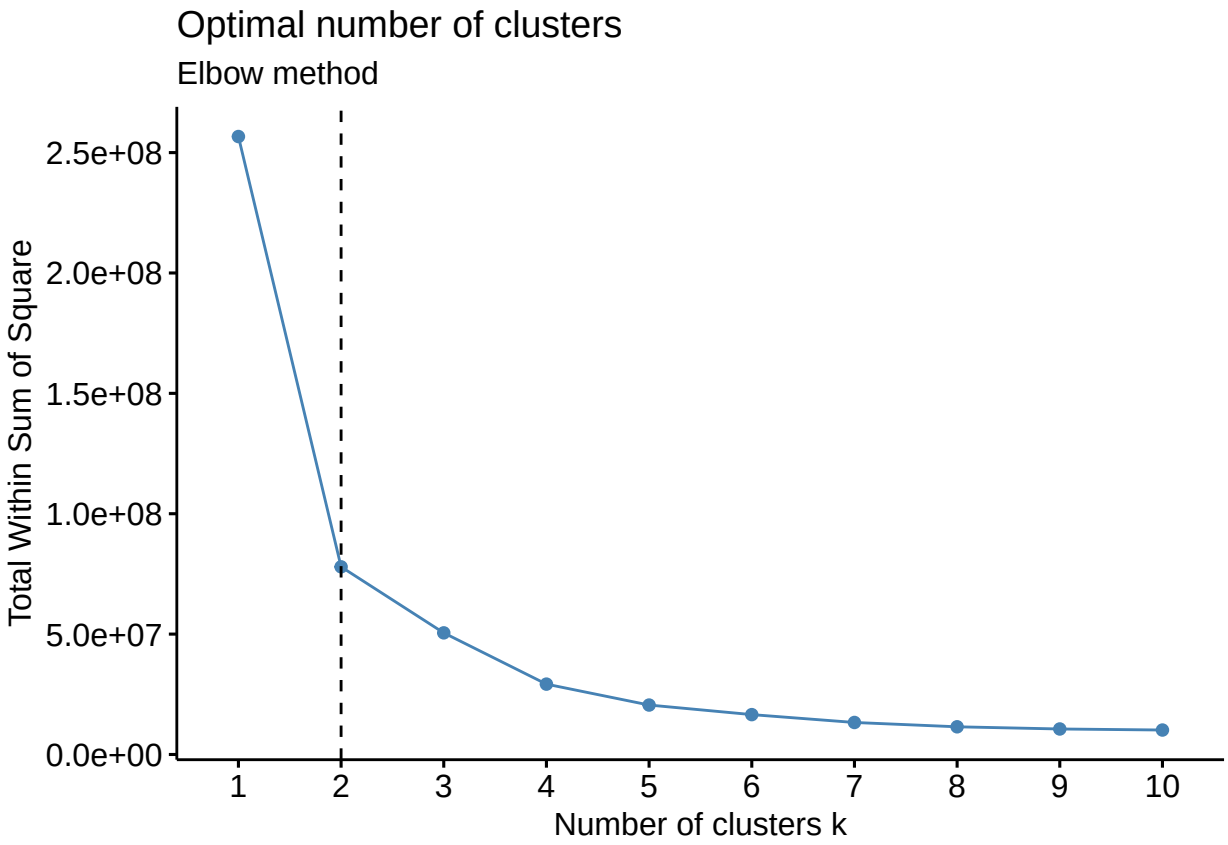



```
set.seed(143)
library(factoextra)
```

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

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

```
fviz_nbclust(wisc.data, kmeans, method = "wss",k.max=10, nstart=20, iter.max=20) +
  geom_vline(xintercept = 2
             , linetype = 2)+
  labs(subtitle = "Elbow method")
```

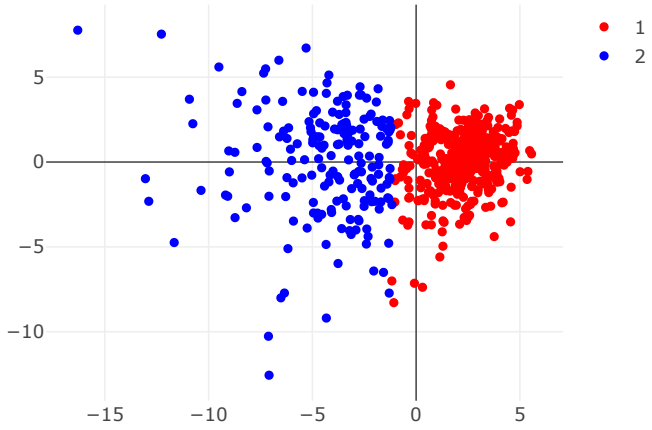



```
set.seed(143)
library(cluster)
library(plotly)
```

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

## The following objects are masked from 'package:plyr':
##
##     arrange, mutate, rename, summarise

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

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

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

```r
# Perform k-Means clustering
km_out <- kmeans(data.scaled, centers = 2, nstart = 25)

plot_ly(x=wisc.pr$x[,1],y=wisc.pr$x[,2], color = as.factor(km_out$cluster), colors=c("red","blue"))
```

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

## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plotly.com/r/reference/#scatter-mode
```




```
# Create a k-means model on wisc.data: wisc.km
wisc.km <- kmeans(scale(wisc.data), centers = 2, nstart = 20)
```

```r
# Compare k-means to actual diagnoses
table(wisc.km$cluster, diagnosis)
```

```
##    diagnosis
##       0   1
##   1 343  37
##   2  14 175
```

```r
# Create a matrix or data table to represent the counts from the clustering results
# Diagnosis (0 for benign, 1 for malignant)
# Rows represent clusters, columns represent actual diagnosis
results_matrix <- matrix(c(14, 175, 343, 37), nrow = 2, byrow = TRUE)

# Assign row names and column names for clarity
rownames(results_matrix) <- c("Cluster 1", "Cluster 2")
colnames(results_matrix) <- c("Benign", "Malignant")

# Calculate the total number of correct predictions
# Correct predictions for Cluster 1 (Malignant) and Cluster 2 (Benign)
correct_predictions <- results_matrix[1, 2] + results_matrix[2, 1]

# Calculate total predictions
total_predictions <- sum(results_matrix)

# Calculate accuracy
accuracy <- correct_predictions / total_predictions

# Print the accuracy
print(accuracy)
```

```
## [1] 0.9103691
```

Clustering on PCA results

```r
# Create a hierarchical clustering model: wisc.pr.hclust
wisc.pr.hclust <- hclust(dist(wisc.pr$x[, 1:7]), method = "complete")
```

```r
# Cut model into 4 clusters: wisc.pr.hclust.clusters
wisc.pr.hclust.clusters <- cutree(wisc.pr.hclust, k = 4)
# Compare to actual diagnoses
table( wisc.pr.hclust.clusters, diagnosis)
```

```
##                         diagnosis
## wisc.pr.hclust.clusters   0   1
##                       1   5 113
##                       2 350  97
##                       3   2   0
##                       4   0   2
```

```r
# Create a matrix to represent the updated counts from the clustering results
# Rows represent clusters, columns represent actual diagnosis
updated_matrix <- matrix(c(5, 113, 350, 97, 2, 0, 0, 2),
                         nrow = 4, byrow = TRUE,
                         dimnames = list(c("Cluster 1", "Cluster 2", "Cluster 3", "Cluster 4"),
                                         c("Benign", "Malignant")))

# Calculate the total number of correct predictions
# Assuming Cluster 1 and 4 are meant to predict malignant, and Cluster 2 and 3 predict benign
correct_predictions <- updated_matrix[1, 2] +  # Malignant correctly predicted in Cluster 1
                       updated_matrix[2, 1] +  # Benign correctly predicted in Cluster 2
                       updated_matrix[3, 1] +  # Benign correctly predicted in Cluster 3
                       updated_matrix[4, 2]    # Malignant correctly predicted in Cluster 4

# Calculate total predictions
total_predictions <- sum(updated_matrix)

# Calculate accuracy
accuracy <- correct_predictions / total_predictions
#accuracy= 0.9321763
# Print the accuracy
print(accuracy)
```

```
## [1] 0.8207381
```