STATISTICAL LEARNING REPORT

LM – Data Science

Salvatore Damiano Calabrese
100005587

BREAST CANCER WISCONSIN ANALYSIS: HOW WE CAN DISTINGUISH BENIGN AND MALIGNANT BREAST TUMOURS

INDICE PAG.

ABOUT THE DATA

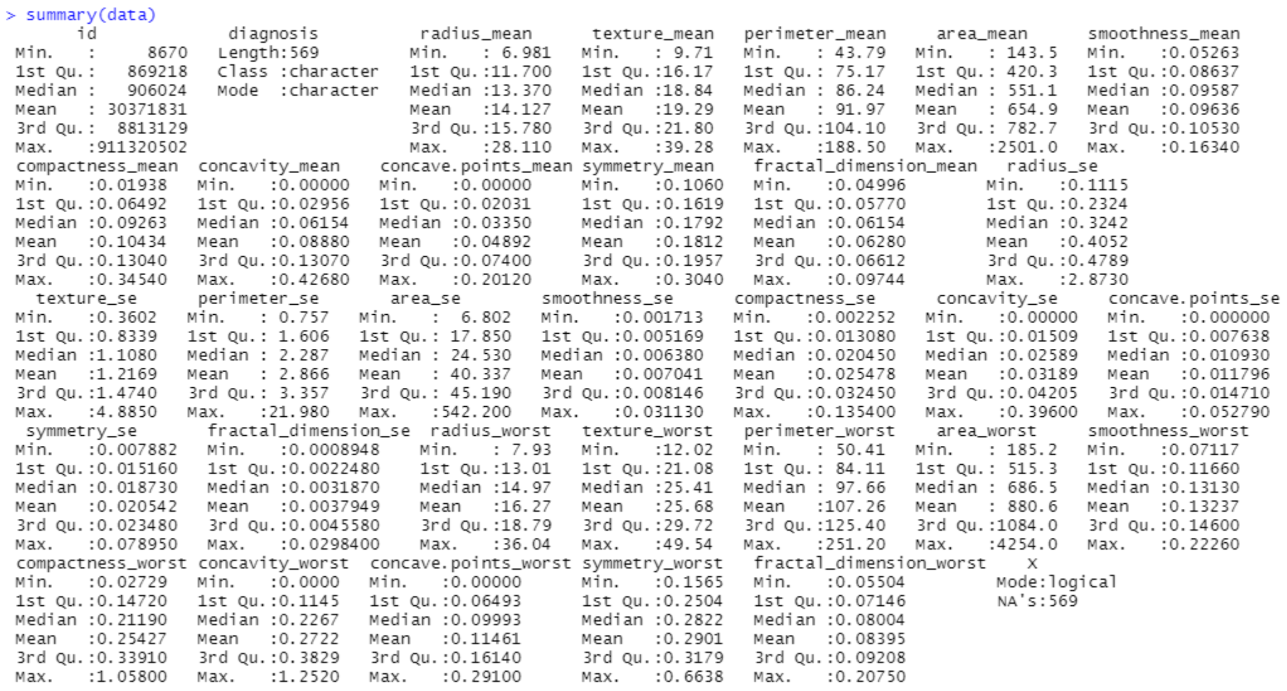
- The "Breast Cancer Wisconsin (Diagnostic) Data Set" is a widely used dataset for classifying breast tumors as either benign or malignant. The dataset contains information about breast tumors that were diagnosed using fine needle aspiration (FNA) of the breast mass.
  The dataset contains a total of 569 observations and 32 variables. The first variable is an ID number, the second variable is the diagnosis (M for malignant or B for benign), and the remaining variables are numeric and represent various characteristics of the cell nuclei present in the FNA samples. These characteristics include radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension.
  Here is a brief description of the variables in the dataset:
  ID: Patient ID number
- Diagnosis: M for malignant or B for benign
- radius_mean: Mean radius of the cell nuclei
- texture_mean: Mean texture (standard deviation of gray-scale values) of the cell nuclei
- perimeter_mean: Mean perimeter of the cell nuclei
- area_mean: Mean area of the cell nuclei
- smoothness_mean: Mean smoothness (local variation in radius lengths) of the cell nuclei
- compactness_mean: Mean compactness (perimeter^2 / area - 1.0) of the cell nuclei
- concavity_mean: Mean concavity (severity of concave portions of the contour) of the cell nuclei
- concave points_mean: Mean number of concave portions of the contour of the cell nuclei
- symmetry_mean: Mean symmetry of the cell nuclei
- fractal_dimension_mean: Mean "coastline approximation" - 1 of the cell nuclei
- radius_se: Standard error of the radius of the cell nuclei
- texture_se: Standard error of the texture of the cell nuclei
- perimeter_se: Standard error of the perimeter of the cell nuclei
- area_se: Standard error of the area of the cell nuclei
- smoothness_se: Standard error of the smoothness of the cell nuclei
- compactness_se: Standard error of the compactness of the cell nuclei
- concavity_se: Standard error of the concavity of the cell nuclei
- concave points_se: Standard error of the number of concave portions of the contour of the cell nuclei
- symmetry_se: Standard error of the symmetry of the cell nuclei
- fractal_dimension_se: Standard error of the "coastline approximation" - 1 of the cell nuclei
- radius_worst: Worst (largest) radius of the cell nuclei
- texture_worst: Worst (most irregular) texture of the cell nuclei
- perimeter_worst: Worst (largest) perimeter of the cell nuclei
- area_worst: Worst (largest) area of the cell nuclei
- smoothness_worst: Worst (most irregular) smoothness of the cell nuclei
- compactness_worst: Worst (largest) compactness of the cell nuclei
- concavity_worst: Worst (largest) concavity of the cell nuclei
- concave points_worst: Worst (largest) number of concave portions of the contour of the cell nuclei
- symmetry_worst: Worst (most irregular) symmetry of the cell nuclei
- fractal_dimension_worst: Worst (most irregular) "coastline approximation" - 1 of the cell nuclei.

n.b. THE DATA HAVE NO MISSING VALUE.

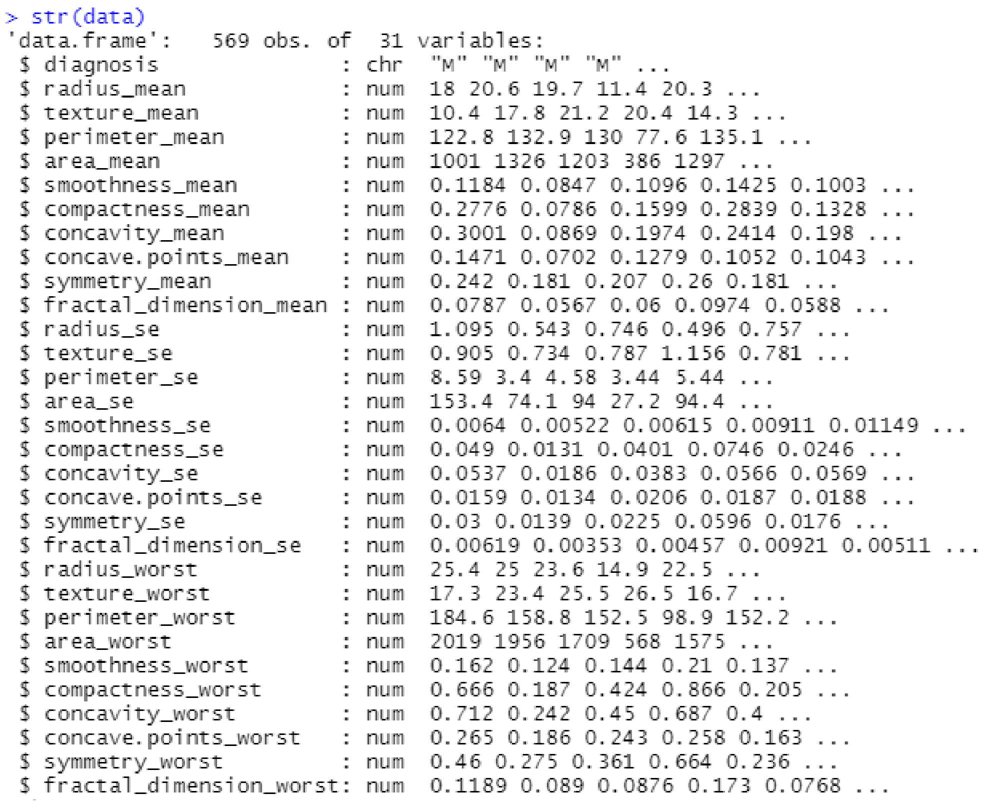
EXPLORATORY ANALYSIS IN THE OTHER FILE

A first look to our dataset

First of all, we use $summary()$ and str() to have a first view of our data.



```
> summary(data)
       id              diagnosis          radius_mean      texture_mean   perimeter_mean     area_mean      smoothness_mean  
 Min.   :     8670   Length:569         Min.   : 6.981   Min.   : 9.71   Min.   : 43.79   Min.   : 143.5   Min.   :0.05263  
 1st Qu.:   869218   Class :character   1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637  
 Median :   906024   Mode  :character   Median :13.370   Median :18.84   Median : 86.24   Median : 551.1   Median :0.09587  
 Mean   : 30371831                      Mean   :14.127   Mean   :19.29   Mean   : 91.97   Mean   : 654.9   Mean   :0.09636  
 3rd Qu.:  8813129                      3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530  
 Max.   :911320502                      Max.   :28.110   Max.   :39.28   Max.   :188.50   Max.   :2501.0   Max.   :0.16340  
 compactness_mean  concavity_mean    concave.points_mean symmetry_mean    fractal_dimension_mean   radius_se     
 Min.   :0.01938   Min.   :0.00000   Min.   :0.00000     Min.   :0.1060   Min.   :0.04996         Min.   :0.1115  
 1st Qu.:0.06492   1st Qu.:0.02956   1st Qu.:0.02031     1st Qu.:0.1619   1st Qu.:0.05770         1st Qu.:0.2324  
 Median :0.09263   Median :0.06154   Median :0.03350     Median :0.1792   Median :0.06154         Median :0.3242  
 Mean   :0.10434   Mean   :0.08880   Mean   :0.04892     Mean   :0.1812   Mean   :0.06280         Mean   :0.4052  
 3rd Qu.:0.13040   3rd Qu.:0.13070   3rd Qu.:0.07400     3rd Qu.:0.1957   3rd Qu.:0.06612         3rd Qu.:0.4789  
 Max.   :0.34540   Max.   :0.42680   Max.   :0.20120     Max.   :0.3040   Max.   :0.09744         Max.   :2.8730  
   texture_se      perimeter_se       area_se        smoothness_se       compactness_se       concavity_se     concave.points_se  
 Min.   :0.3602   Min.   : 0.757   Min.   :  6.802   Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000  
 1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638  
 Median :1.1080   Median : 2.287   Median : 24.530   Median :0.006380   Median :0.020450   Median :0.02589   Median :0.010930  
 Mean   :1.2169   Mean   : 2.866   Mean   : 40.337   Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean   :0.011796  
 3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710  
 Max.   :4.8850   Max.   :21.980   Max.   :542.200   Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.   :0.052790  
  symmetry_se       fractal_dimension_se  radius_worst   texture_worst   perimeter_worst     area_worst     smoothness_worst 
 Min.   :0.007882   Min.   :0.0008948    Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2   Min.   :0.07117  
 1st Qu.:0.015160   1st Qu.:0.0022480    1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660  
 Median :0.018730   Median :0.0031870    Median :14.97   Median :25.41   Median : 97.66   Median : 686.5   Median :0.13130  
 Mean   :0.020542   Mean   :0.0037949    Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6   Mean   :0.13237  
 3rd Qu.:0.023480   3rd Qu.:0.0045580    3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600  
 Max.   :0.078950   Max.   :0.0298400    Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0   Max.   :0.22260  
 compactness_worst concavity_worst   concave.points_worst symmetry_worst    fractal_dimension_worst    X          
 Min.   :0.02729   Min.   :0.0000   Min.   :0.00000      Min.   :0.1565   Min.   :0.05504          Mode:logical  
 1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493      1st Qu.:0.2504   1st Qu.:0.07146          NA's:569      
 Median :0.21190   Median :0.2267   Median :0.09993      Median :0.2822   Median :0.08004                        
 Mean   :0.25427   Mean   :0.2722   Mean   :0.11461      Mean   :0.2901   Mean   :0.08395                        
 3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140      3rd Qu.:0.3179   3rd Qu.:0.09208                        
 Max.   :1.05800   Max.   :1.2520   Max.   :0.29100      Max.   :0.6638   Max.   :0.20750                        
```



```
> str(data)
'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 ...
```

We have all data in numeric form, except diagnosis which is M and B. Let's convert this into numeric only.

data$diagnosis <- factor(data$diagnosis, levels = c("M","B"), labels = c(0,1))

Now converting factors to character and then character to numeric, if we convert this directly to numeric it will give errors.

data$diagnosis <- as.character(data$diagnosis)

data$diagnosis <- as.numeric(data$diagnosis)

Changing the position of dependent variable i.e., diagnosis to the extreme right of the data to avoid confusion . We will use this by using tidyverse function relocate() , .after(), .before(). Here we need to shift diagnosis column after fractal_dimension_worst.

data <- data %>% relocate(diagnosis,.after= fractal_dimension_worst)

Univariate statistical modelling

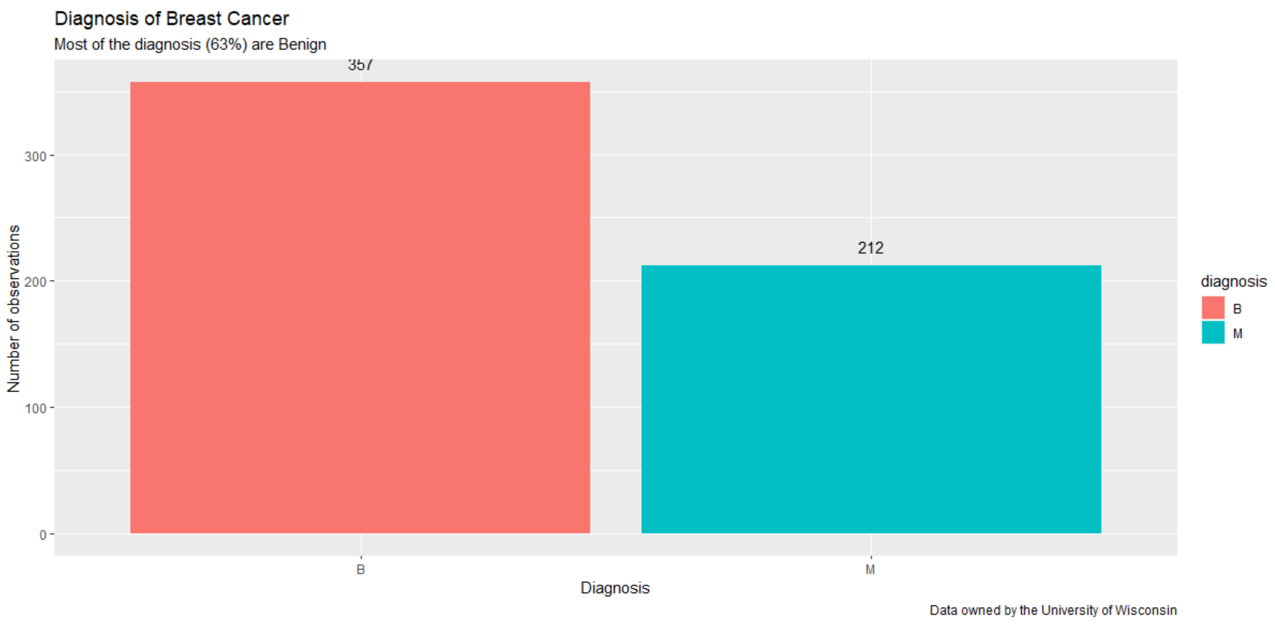
To be more precise, all variables, except diagnosis which is binary, are continuous rv. Here are some examples of univariate analysis on just some of the variables. Unfortunately, analyzing all 32 variables would be particularly time-consuming, so I decided to analyze only a continuous rv, "radius_mean", and "diagnosis", who is a binary variable.

Diagnosis

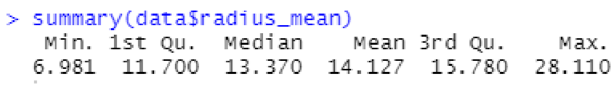
Our knowledge on breast cancer has shown us that there they span between benign and malignant. Gaining insight through the visualization of both will help us understand how both are different in characteristics. We shall analyze the data first by the variables relating to size.

```
> table <- table(data$diagnosis)
> table

  B   M 
357 212 
```

```
ggplot(data = data, aes(x = diagnosis, fill = diagnosis)) +
  geom_bar()+
  geom_text(stat='count', aes(label=..count..), vjust=-1) +
  labs(title = 'Diagnosis of Breast Cancer',
       subtitle = 'Most of the diagnosis (63%) are Benign',
       caption = 'Data owned by the University of Wisconsin',
       x = 'Diagnosis', y = 'Number of observations')
```

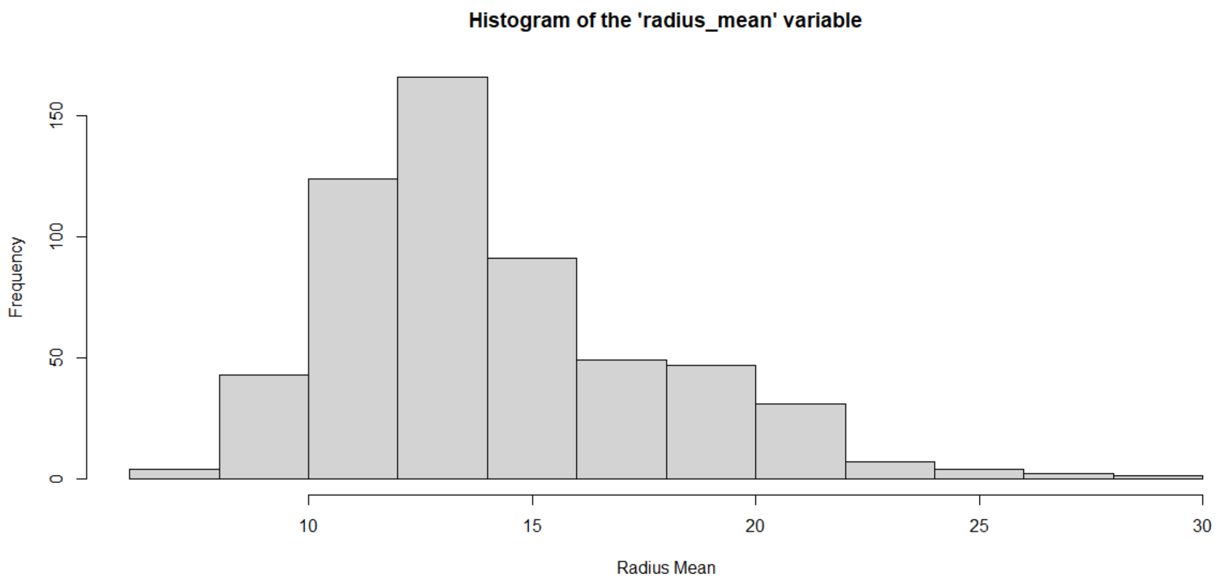


Radius mean

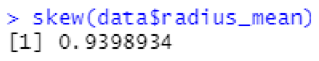


```
> summary(data$radius_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  6.981  11.700  13.370  14.127  15.780  28.110 
```

hist(data$radius_mean, main = "Histogram of the 'radius_mean' variable", xlab = "Radius Mean")



From the graph we can see how the radius_mean character does not seem to adapt to a normal distribution: there is a peak of the frequency distributions around the value of 14 and then a right skewness. We can confirm this observation by looking to the function skew() from labstatR package, who can compute the Fisher asymmetry index, that assume positive values for right skewness and negative values for left skewness:



```
> skew(data$radius_mean)
[1] 0.9398934
```

We observe a positive value, so we can affirm that the distribution is right skewed.

Normality test: > shapiro.test(data$radius_mean)

```
Shapiro-Wilk normality test

data:  data$radius_mean

W = 0.94107, p-value = 3.106e-14
```

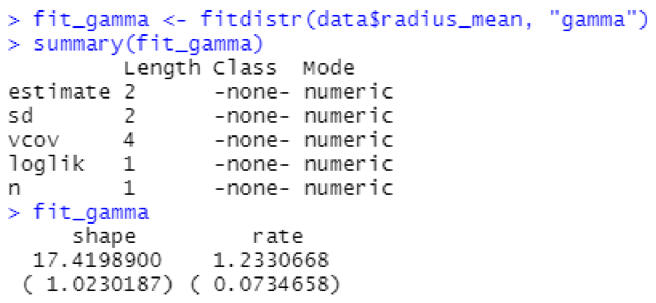
The Shapiro-Wilk test for normality provides a test statistic (W) and a p-value.
In our example, the test statistic is W = 0.94107 and the p-value is p = 3.106e-14. The test statistic measures the discrepancy between the observed data and what would be expected under the assumption of normality. The p-value indicates the probability of observing a test statistic as extreme or more extreme than the observed one, assuming that the null hypothesis of normality is true.

In this case, the p-value is very small (3.106e-14), which indicates strong evidence against the null hypothesis of normality. Therefore, we can conclude that the "radius_mean" variable is not normally distributed.

Gamma distribution

The output fit_gamma is a summary of the maximum likelihood estimates of the shape and rate parameters of the gamma distribution that best fit the data.

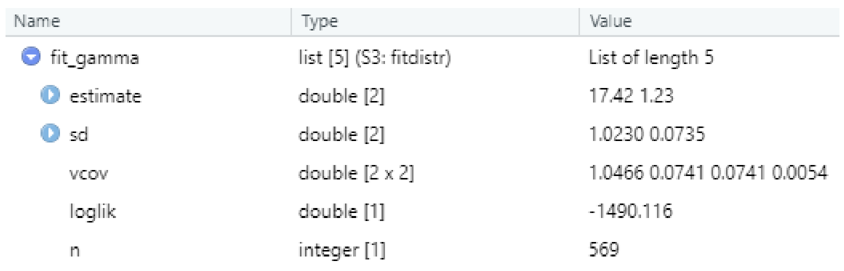
In our case, the estimated shape parameter is 17.4199 and the estimated rate parameter is 1.2331. The standard errors of these estimates are given in parentheses below the estimate values.
The shape parameter is an index of the shape of the gamma distribution. A higher shape parameter indicates a more peaked and longer-tailed distribution. The rate parameter, on the other hand, is an index of the scale of the distribution. A higher rate parameter indicates a distribution with a higher mean value.



```
> fit_gamma <- fitdistr(data$radius_mean, "gamma")
> summary(fit_gamma)
         Length Class  Mode   
estimate 2      -none- numeric
sd       2      -none- numeric
vcov     4      -none- numeric
loglik   1      -none- numeric
n        1      -none- numeric
> fit_gamma
     shape         rate    
  17.4198900    1.2330668 
 ( 1.0230187) ( 0.0734658)
```

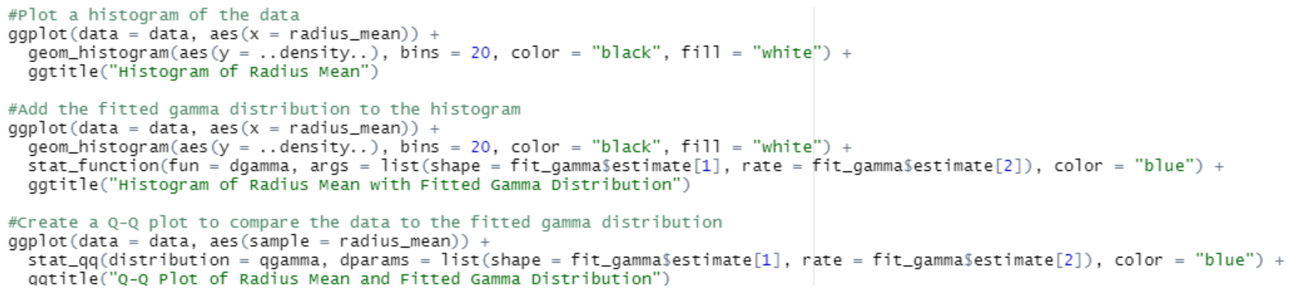
The fitdistr() function from the MASS package estimates the parameters of the gamma distribution from the data using the maximum likelihood method. The output of the fitdistr() function is an object containing information about the parameter estimates, standard errors, covariance matrix, log-likelihood value, and number of observations used in the estimation.
The output of the summary() function applied to this object provides a summary of the parameter estimates and associated statistics. In this case, the output of summary(fit_gamma) contains the following components:

- estimate: A vector of length 2 containing the estimated values of the shape and rate parameters of the gamma distribution, respectively.
- sd: A vector of length 2 containing the estimated standard errors of the shape and rate parameters, respectively.
- vcov: A 2x2 matrix containing the estimated covariance matrix of the shape and rate parameters.
- loglik: The value of the log-likelihood function at the estimated parameter values.
- n: The number of observations used in the estimation.

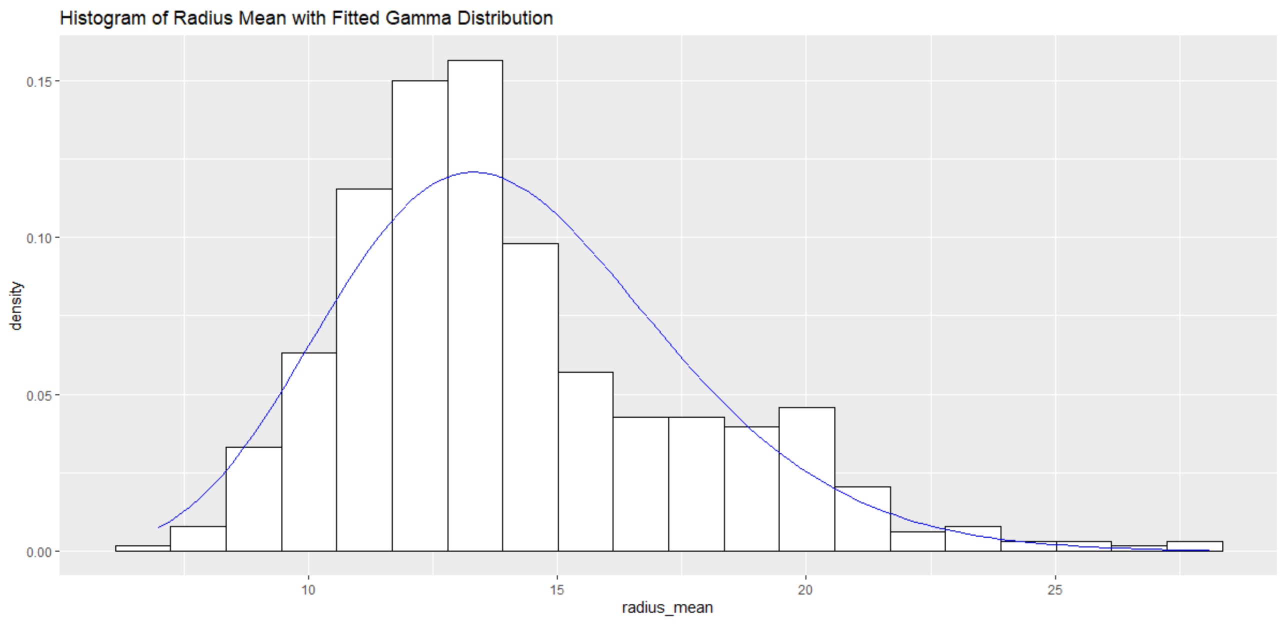


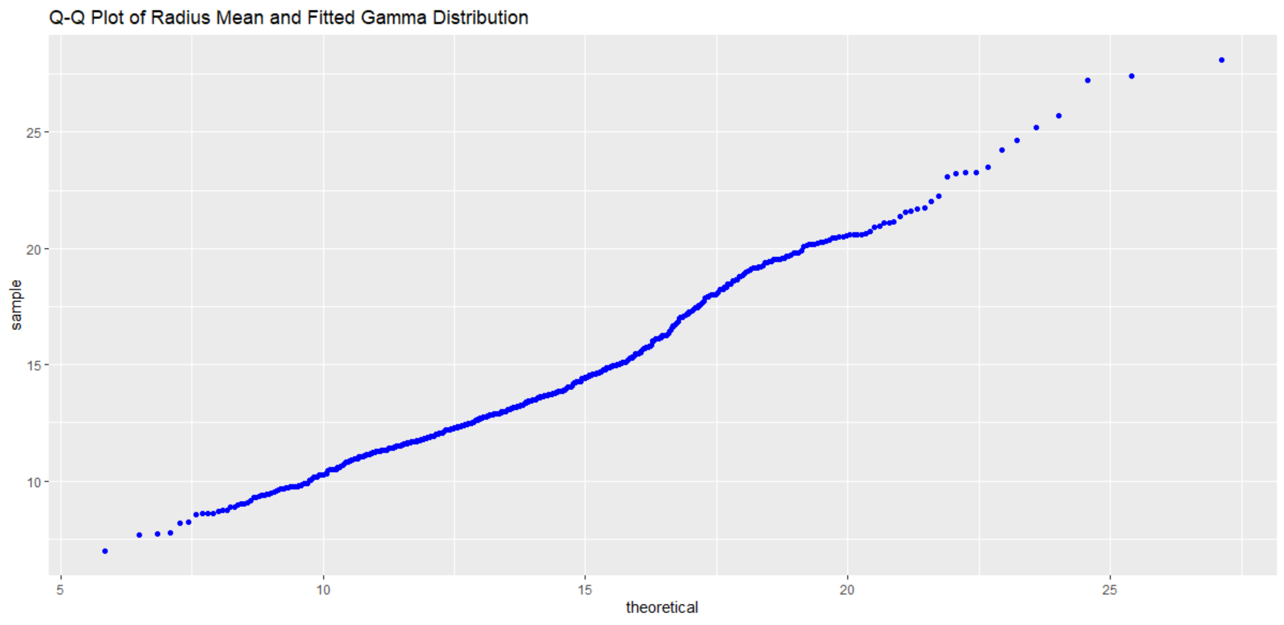
Name	Type	Value
fit_gamma	list [5] (S3: fitdistr)	List of length 5
estimate	double [2]	17.42 1.23
sd	double [2]	1.0230 0.0735
vcov	double [2 x 2]	1.0466 0.0741 0.0741 0.0054
loglik	double [1]	-1490.116
n	integer [1]	569

To visually assess the goodness of fit of the gamma distribution to our data we have to build plots:



```
#Plot a histogram of the data
ggplot(data = data, aes(x = radius_mean)) +
  geom_histogram(aes(y = ..density..), bins = 20, color = "black", fill = "white") +
  ggtitle("Histogram of Radius Mean")

#Add the fitted gamma distribution to the histogram
ggplot(data = data, aes(x = radius_mean)) +
  geom_histogram(aes(y = ..density..), bins = 20, color = "black", fill = "white") +
  stat_function(fun = dgamma, args = list(shape = fit_gamma$estimate[1], rate = fit_gamma$estimate[2]), color = "blue") +
  ggtitle("Histogram of Radius Mean with Fitted Gamma Distribution")

#Create a Q-Q plot to compare the data to the fitted gamma distribution
ggplot(data = data, aes(sample = radius_mean)) +
  stat_qq(distribution = qgamma, dparams = list(shape = fit_gamma$estimate[1], rate = fit_gamma$estimate[2]), color = "blue") +
  ggtitle("Q-Q Plot of Radius Mean and Fitted Gamma Distribution")
```



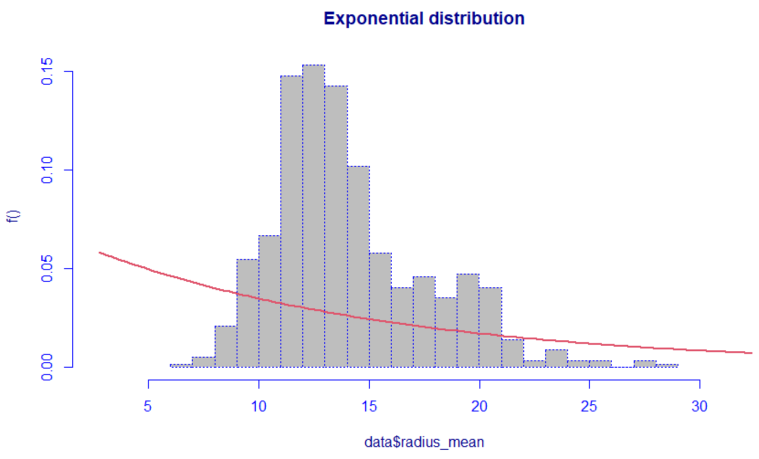


By visually inspecting the histogram and the Q-Q plot, we can get an idea of how well the gamma distribution fits the data. If the histogram closely matches the fitted gamma distribution and the points on the Q-Q plot fall on a straight line, like in our case, it suggests that the gamma distribution is a good fit for the data.

For a complete analysis we should compute AIC and BIC:

```
> AIC(fit_gamma) # AIC (to be minimized)
[1] 2984.232
> fit_gamma$sbc  # BIC (to be minimized)
[1] 2992.92
```

Exponential distribution

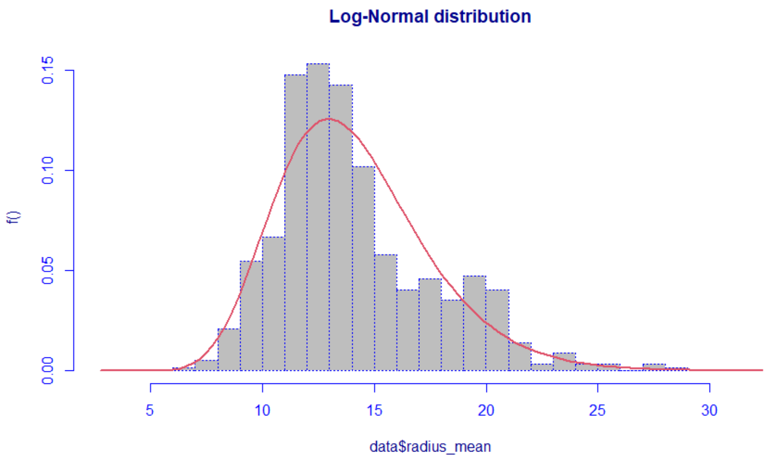


fit.EXP <- histDist(data$radius_mean, family=EXP, nbins = 30, main="Exponential distribution")

We can easily see that exponential distribution fit very bad, as is confirmed by AIC and BIC:

```
> AIC(fit.EXP) # AIC (to be minimized)
[1] 4153.547
> fit.EXP$sbc  # BIC (to be minimized)
[1] 4157.891
```

Log-normal distribution

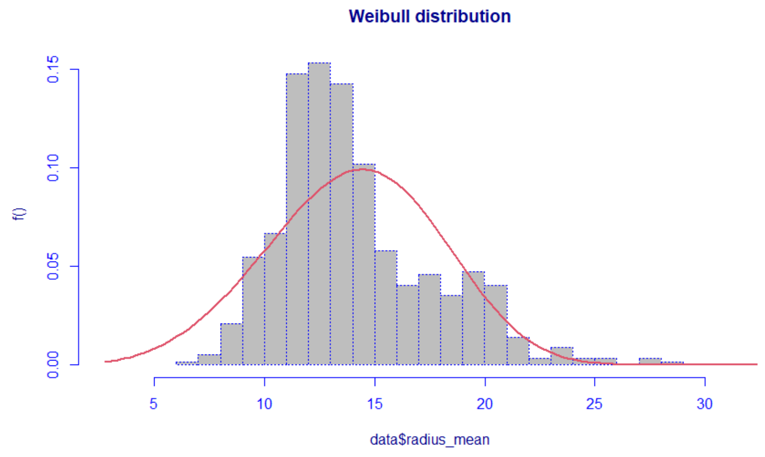


fit.LOGNO <- histDist(data$radius_mean, family=LOGNO, nbins = 30, main="Log-Normal distribution")

Computing AIC and BIC we can see that this distribution fit better than the Gamma distribution and is, right now, the better model to describe our data:

```
> AIC(fit.LOGNO) # AIC (to be minimized)
[1] 2965.642
> fit.LOGNO$sbc  # BIC (to be minimized)
[1] 2974.33
```

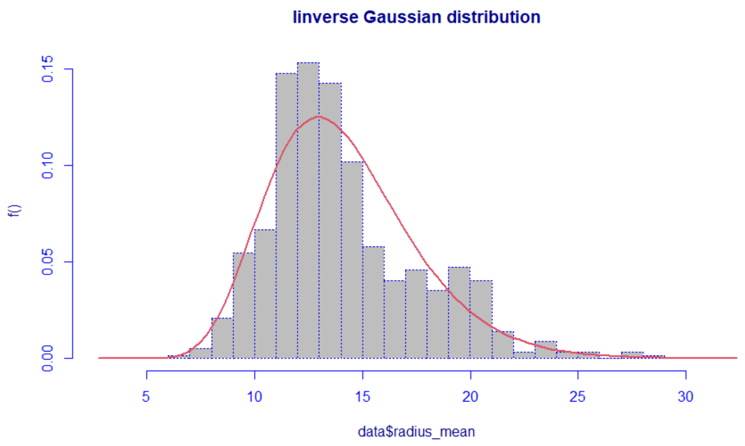
Weibull distribution

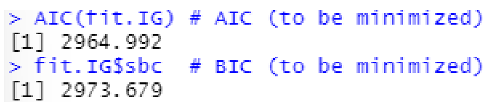


```
> AIC(fit.WEI) # AIC (to be minimized)
[1] 3096.203
> fit.WEI$sbc  # BIC (to be minimized)
[1] 3104.891
```

This distribution provides us worst results than Log-normal and Gamma.

Inverse gaussian distribution

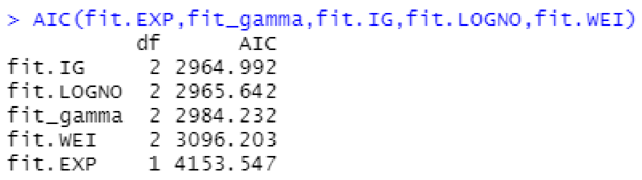




```
> AIC(fit.IG) # AIC (to be minimized)
[1] 2964.992
> fit.IG$sbc  # BIC (to be minimized)
[1] 2973.679
```

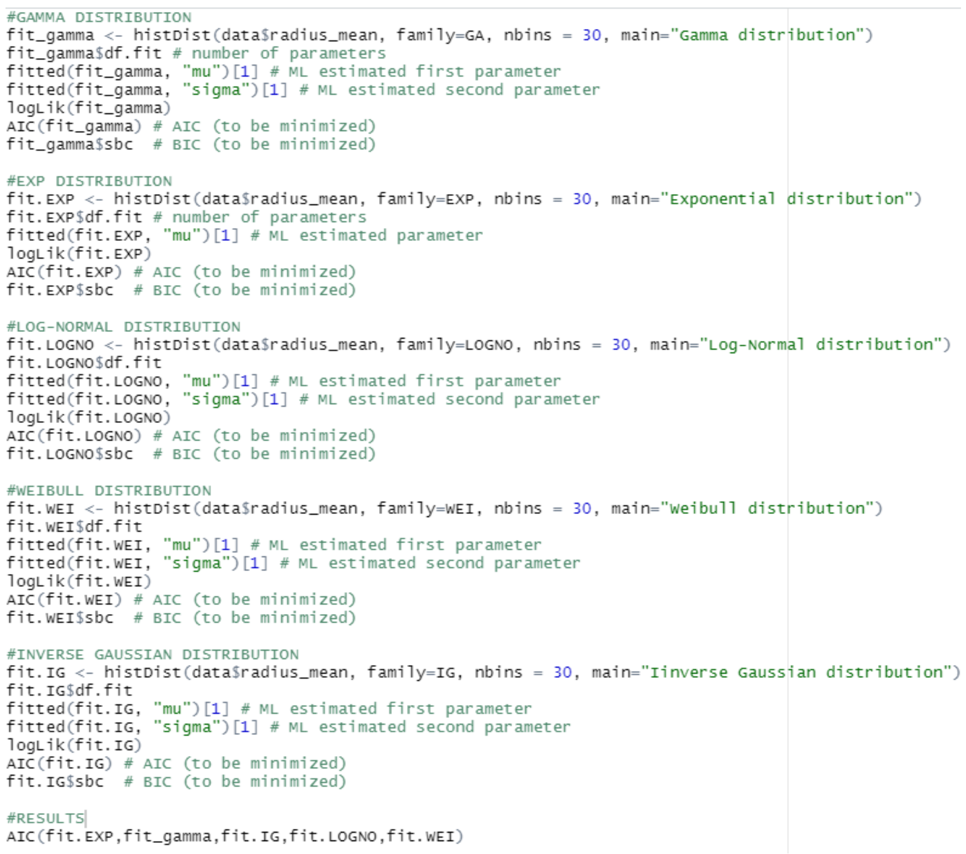
The results are very close to the log-normal but fit a little better.

As we can see from the following table the Inverse Gaussian distribution is the one who fit better.



```
> AIC(fit.EXP,fit_gamma,fit.IG,fit.LOGNO,fit.WEI)
          df      AIC
fit.IG     2 2964.992
fit.LOGNO  2 2965.642
fit_gamma  2 2984.232
fit.WEI    2 3096.203
fit.EXP    1 4153.547
```

Code used for this analysis:



```
#GAMMA DISTRIBUTION
fit_gamma <- histDist(data$radius_mean, family=GA, nbins = 30, main="Gamma distribution")
fit_gamma$df.fit # number of parameters
fitted(fit_gamma, "mu")[1] # ML estimated first parameter
fitted(fit_gamma, "sigma")[1] # ML estimated second parameter
logLik(fit_gamma)
AIC(fit_gamma) # AIC (to be minimized)
fit_gamma$sbc  # BIC (to be minimized)

#EXP DISTRIBUTION
fit.EXP <- histDist(data$radius_mean, family=EXP, nbins = 30, main="Exponential distribution")
fit.EXP$df.fit # number of parameters
fitted(fit.EXP, "mu")[1] # ML estimated parameter
logLik(fit.EXP)
AIC(fit.EXP) # AIC (to be minimized)
fit.EXP$sbc  # BIC (to be minimized)

#LOG-NORMAL DISTRIBUTION
fit.LOGNO <- histDist(data$radius_mean, family=LOGNO, nbins = 30, main="Log-Normal distribution")
fit.LOGNO$df.fit
fitted(fit.LOGNO, "mu")[1] # ML estimated first parameter
fitted(fit.LOGNO, "sigma")[1] # ML estimated second parameter
logLik(fit.LOGNO)
AIC(fit.LOGNO) # AIC (to be minimized)
fit.LOGNO$sbc  # BIC (to be minimized)

#WEIBULL DISTRIBUTION
fit.WEI <- histDist(data$radius_mean, family=WEI, nbins = 30, main="Weibull distribution")
fit.WEI$df.fit
fitted(fit.WEI, "mu")[1] # ML estimated first parameter
fitted(fit.WEI, "sigma")[1] # ML estimated second parameter
logLik(fit.WEI)
AIC(fit.WEI) # AIC (to be minimized)
fit.WEI$sbc  # BIC (to be minimized)

#INVERSE GAUSSIAN DISTRIBUTION
fit.IG <- histDist(data$radius_mean, family=IG, nbins = 30, main="Iinverse Gaussian distribution")
fit.IG$df.fit
fitted(fit.IG, "mu")[1] # ML estimated first parameter
fitted(fit.IG, "sigma")[1] # ML estimated second parameter
logLik(fit.IG)
AIC(fit.IG) # AIC (to be minimized)
fit.IG$sbc  # BIC (to be minimized)

#RESULTS
AIC(fit.EXP,fit_gamma,fit.IG,fit.LOGNO,fit.WEI)
```

Correlation between variables

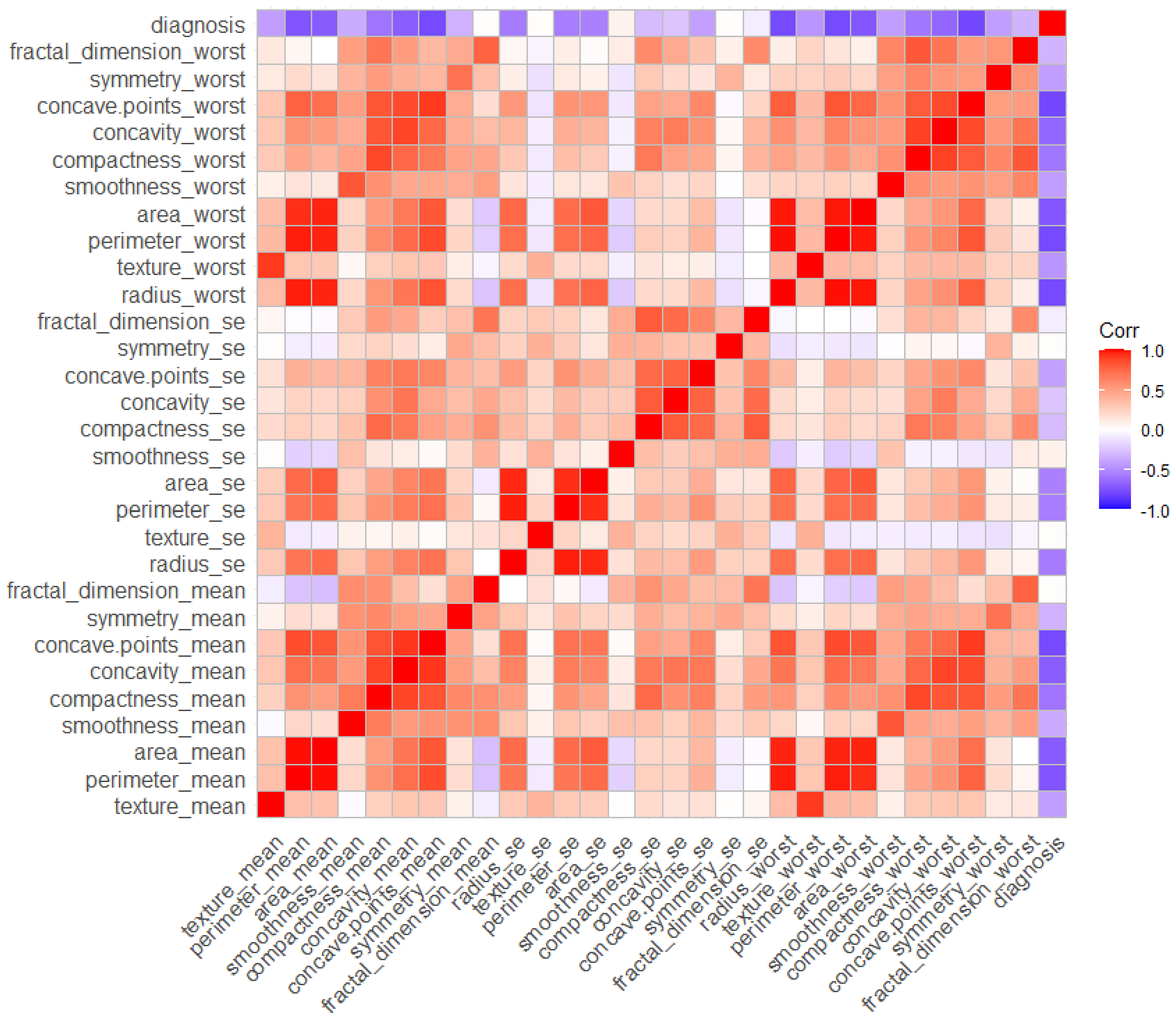
Taking a first look at the correlation between variables serves to identify any potential relationships or patterns between the variables. This can help to guide further analysis and modelling, as well as to identify potential areas of interest for further exploration. Examining the correlation between variables is an important step in any data analysis process.

```
correlation <- cor(data[,3:32])
round(correlation,2)
```

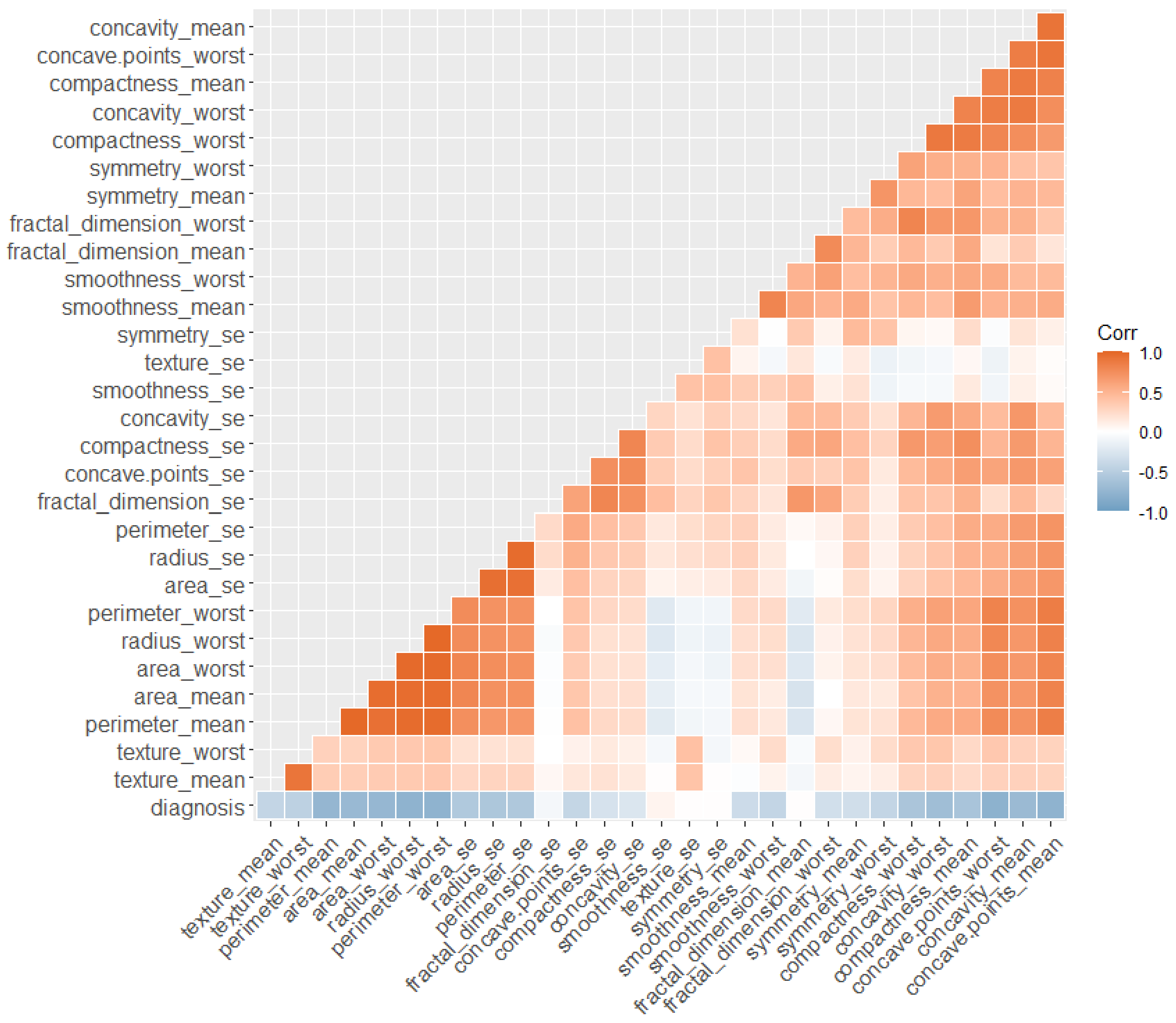
We can describe the matrix using the following plot:

```
ggcorrplot(correlation)
```

Given the size of the matrix, in order to facilitate the reading of the report, only the plot is shown below.



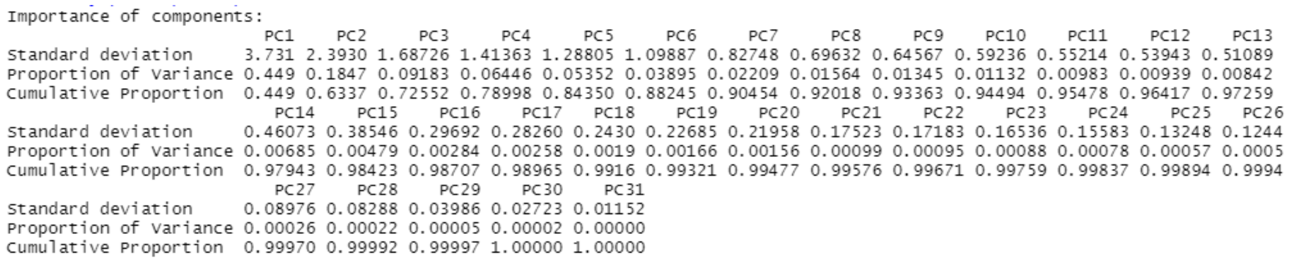
The ggcorrplot package also includes a function for computing a matrix of correlation p-value:



Is important to remember that the p-value is a statistical measure that helps us to determine the likelihood of obtaining a certain result if the null hypothesis is true. It provides a measure of the strength of evidence against the null hypothesis. A p-value greater than 0.05 indicates weak evidence against the null hypothesis and suggests that the observed result could have occurred by chance.

Principal component Analysis

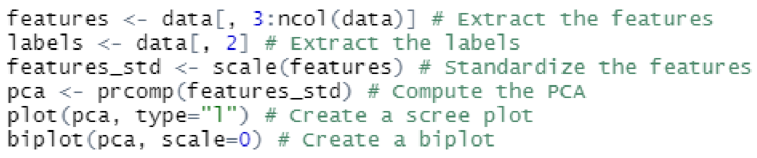
Too many variables can cause such problems like increased computer throughput, complex visualization problems, decrease efficiency by including variables that have no effect on the analysis, make interpretation difficult, etc.
We can use the R bult-in function prcop() to calculate the principal component of the dataset, we have to be sure to specify “scale = TRUE” so that each variables in the dataset are scaled to have a mean of 0 and a standard deviation of 1 before calculating the principal components, or we can scale it manually like I’m doing in the next steps.
n.b. in the results of PCA if cumulative proportion of PCn is 88.7, it means that the sum of proportion of PC1-PCn is 88.7.



```
Importance of components:
                         PC1    PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10    PC11    PC12    PC13
Standard deviation     3.731 2.3930 1.68726 1.41363 1.28805 1.09887 0.82748 0.69632 0.64567 0.59236 0.55214 0.53943 0.51089
Proportion of Variance 0.449 0.1847 0.09183 0.06446 0.05352 0.03895 0.02209 0.01564 0.01345 0.01132 0.00983 0.00939 0.00842
Cumulative Proportion  0.449 0.6337 0.72552 0.78998 0.84350 0.88245 0.90454 0.92018 0.93363 0.94494 0.95478 0.96417 0.97259
                          PC14    PC15    PC16    PC17   PC18    PC19    PC20    PC21    PC22    PC23    PC24    PC25   PC26
Standard deviation     0.46073 0.38546 0.29692 0.28260 0.2430 0.22685 0.21958 0.17523 0.17183 0.16536 0.15583 0.13248 0.1244
Proportion of Variance 0.00685 0.00479 0.00284 0.00258 0.0019 0.00166 0.00156 0.00099 0.00095 0.00088 0.00078 0.00057 0.0005
Cumulative Proportion  0.97943 0.98423 0.98707 0.98965 0.9916 0.99321 0.99477 0.99576 0.99671 0.99759 0.99837 0.99894 0.9994
                          PC27    PC28    PC29    PC30    PC31
Standard deviation     0.08976 0.08288 0.03986 0.02723 0.01152
Proportion of Variance 0.00026 0.00022 0.00005 0.00002 0.00000
Cumulative Proportion  0.99970 0.99992 0.99997 1.00000 1.00000
```

The percentage of variability explained by the principal components can be ascertained through screeplot.

To compute a PCA on the "Breast Cancer Wisconsin (Diagnostic) Data Set" in R, we use the prcomp function. Here's the code I used:



```r
features <- data[, 3:ncol(data)] # Extract the features
labels <- data[, 2] # Extract the labels
features_std <- scale(features) # Standardize the features
pca <- prcomp(features_std) # Compute the PCA
plot(pca, type="l") # Create a scree plot
biplot(pca, scale=0) # Create a biplot
```

Let's comment the code:

- We extract the features (columns 3 to the end) and labels (column 2) from the data.
- We standardize the features using the scale function.
- We compute the PCA using the prcomp() function on the standardized features.
- We create a scree plot using the plot() function, specifying type="l" to plot a line chart.
- We create a biplot using the biplot() function, specifying scale=0 to prevent the arrows from being rescaled based on the eigenvalues.

The scree plot shows the proportion of variance explained by each principal component. We can see that the first two principal components explain a large portion of the variance, with the first component explaining 44,3% and the second component explaining 19%.

The biplot shows the loadings (directions) and scores (coordinates) of the data points on the first two principal components. We can see that the features "texture_mean", "perimeter_mean", "area_mean", "concavity_mean", "concave points_mean", "texture_se", "perimeter_se", "area_se", "concavity_se", "concave points_se", "texture_worst", "perimeter_worst", "area_worst", "concavity_worst", and "concave points_worst" have strong positive loadings on the first principal component, while the features "smoothness_mean", "compactness_mean", "symmetry_mean", "fractal_dimension_mean", "smoothness_se", "compactness_se", "symmetry_se", "fractal_dimension_se", "smoothness_worst", "compactness_worst", "symmetry_worst", and "fractal_dimension_worst" have weak or negative loadings. This suggests that the first principal component is mainly capturing variation in the size and shape of the cell nuclei.

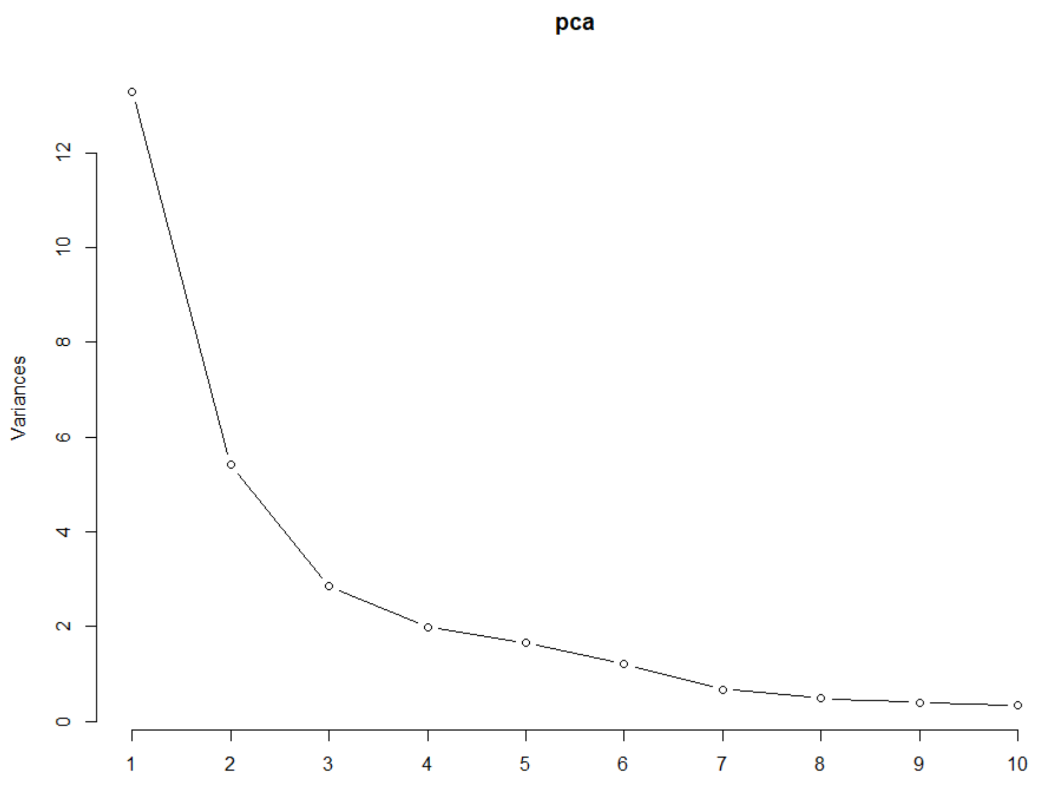
On the other hand, the features "texture_mean", "area_mean", "concavity_mean", "concave points_mean", "texture_se", "area_se", "concavity_se", "concave points_se", "texture_worst", "area_worst", "concavity_worst", and "concave points_worst" have strong positive loadings on the second principal component, while the features "smoothness_mean", "compactness_mean", "symmetry_mean", "fractal_dimension_mean", "smoothness_se", "compactness_se", "symmetry_se", "fractal_dimension_se", "smoothness_worst", "compactness_worst", "symmetry_worst", and "fractal_dimension_worst" have weak or negative loadings. This suggests that the second principal component is mainly capturing variation in the texture and compactness.

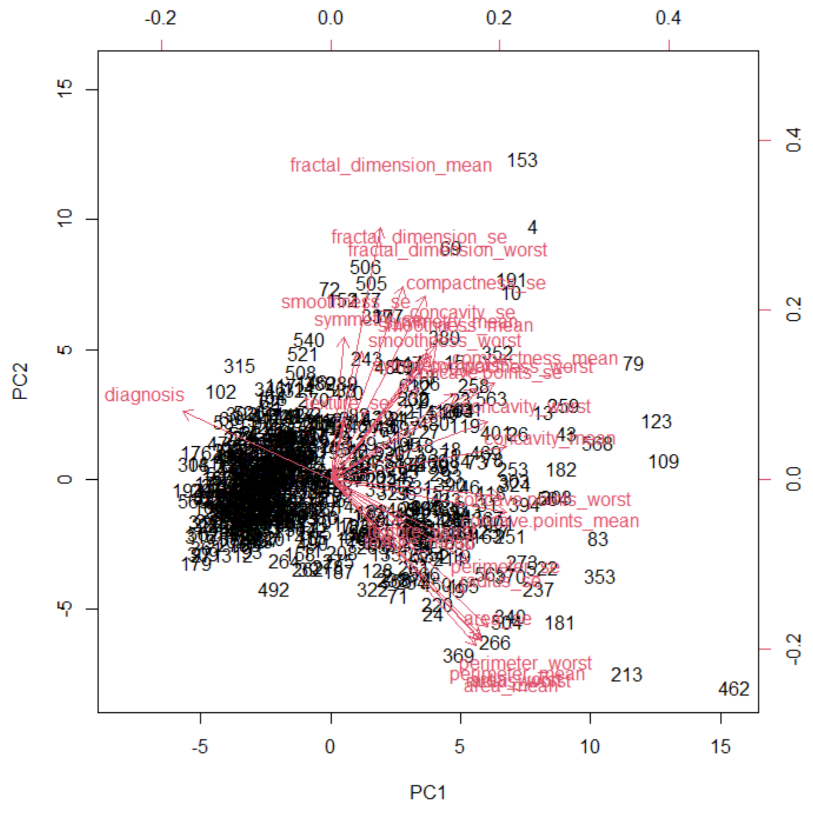
So, describing the results, based on the PCA analysis of the "Breast Cancer Wisconsin (Diagnostic) Data Set", we can see that the first two principal components capture a significant amount of the variation in the data. The first principal component mainly captures variation in the size and shape of the cell nuclei, while the second principal component mainly captures variation in the texture and compactness of the cell nuclei.

These results suggest that the PCA could be useful for reducing the dimensionality of the data and identifying important features for predicting whether a breast tumour is malignant or benign. By focusing on the principal components that explain the most variation in the data, we may be able to build more accurate models using fewer features.

It's also worth noting that these results should be interpreted in conjunction with other analyses of the data, such as predictive modelling or clustering. PCA can provide insights into the structure of the data, but it doesn't necessarily tell us which features are most important for making accurate predictions or identifying subgroups of patients.

Scree plot and biplot





Scree plot

The scree plot is a graphical representation of the proportion of variance explained by each principal component in a PCA analysis. It can be used to determine how many principal components to retain for further analysis.

To read a scree plot, you need to look at the y-axis, which represents the proportion of variance explained, and the x-axis, which represents the number of principal components.

Typically, you want to retain the principal components that explain a large proportion of the variance in the data. The scree plot shows a curve that starts high on the left side and gradually decreases as we move to the right. The point at which the curve starts to level off can be used as a criterion for deciding how many principal components to retain. This is called the "elbow point", and it represents the point at which adding more principal components does not explain much more variance in the data.

In the scree plot for the "Breast Cancer Wisconsin (Diagnostic) Data Set", we can see that the curve starts high on the left and decreases gradually. The elbow point appears to be around the second or third principal component, which suggests that we could retain those components for further analysis. However, the decision of how many components to retain ultimately depends on the specific goals and context of the analysis.

Biplot

A biplot is a graphical representation of the principal components in a PCA analysis, which allows us to visualize the relationships between variables and observations in a lower-dimensional space. To read a biplot, we need to interpret both the direction and length of the arrows representing variables, as well as the position of the observations in the space. Here are some key steps to interpreting a biplot:

- Look at the **direction** of the arrows representing variables. The direction of the arrows indicates the correlation between the variables and the principal components. Variables that point in similar directions are highly correlated with each other and with the principal component.
- Look at the **length** of the arrows representing variables. The length of the arrows indicates the importance of the variable for explaining the variation in the data. Longer arrows indicate variables that are more important for explaining the variation.
- Look at the **position** of the observations in the space. The position of the observations in the biplot indicates their relative scores on the principal components. Observations that are close together in the space are more similar to each other in terms of their variable values.
- Look for **patterns and clusters** in the biplot. Patterns and clusters in the biplot can suggest relationships between variables and observations in the data.

In the biplot for the "Breast Cancer Wisconsin (Diagnostic) Data Set", we can see that the first principal component is mainly associated with variables related to the size and shape of the cell nuclei, such as "radius_mean", "perimeter_mean", "area_mean", "compactness_mean", and "concavity_mean". The second principal component is mainly associated with variables related to the texture and smoothness of the cell nuclei, such as "texture_mean", "smoothness_mean", "concavity_worst", and "fractal_dimension_worst".

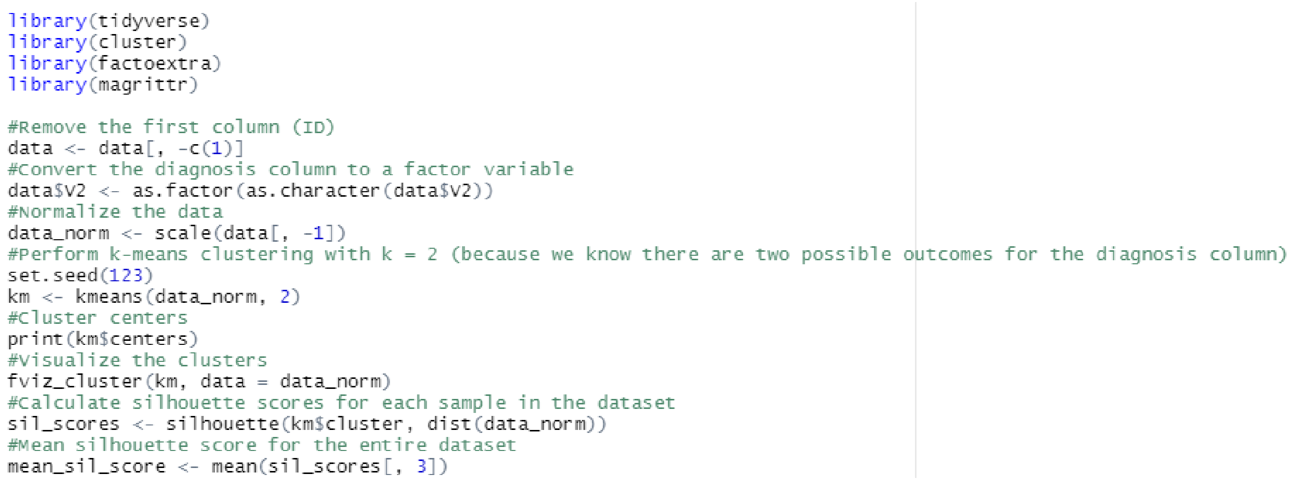
We can also see that there are some clear clusters of observations in the biplot, which could be related to the diagnosis of the breast tumour. The malignant observations are generally located in the upper left quadrant of the plot, while the benign observations are located in the lower right quadrant. This suggests that the first two principal components could be useful for predicting the diagnosis of the breast tumour based on the variables in the dataset.

To sum up

Based on the PCA analysis of the "Breast Cancer Wisconsin (Diagnostic) Data Set" and the interpretation of the results, here are some potential conclusions:

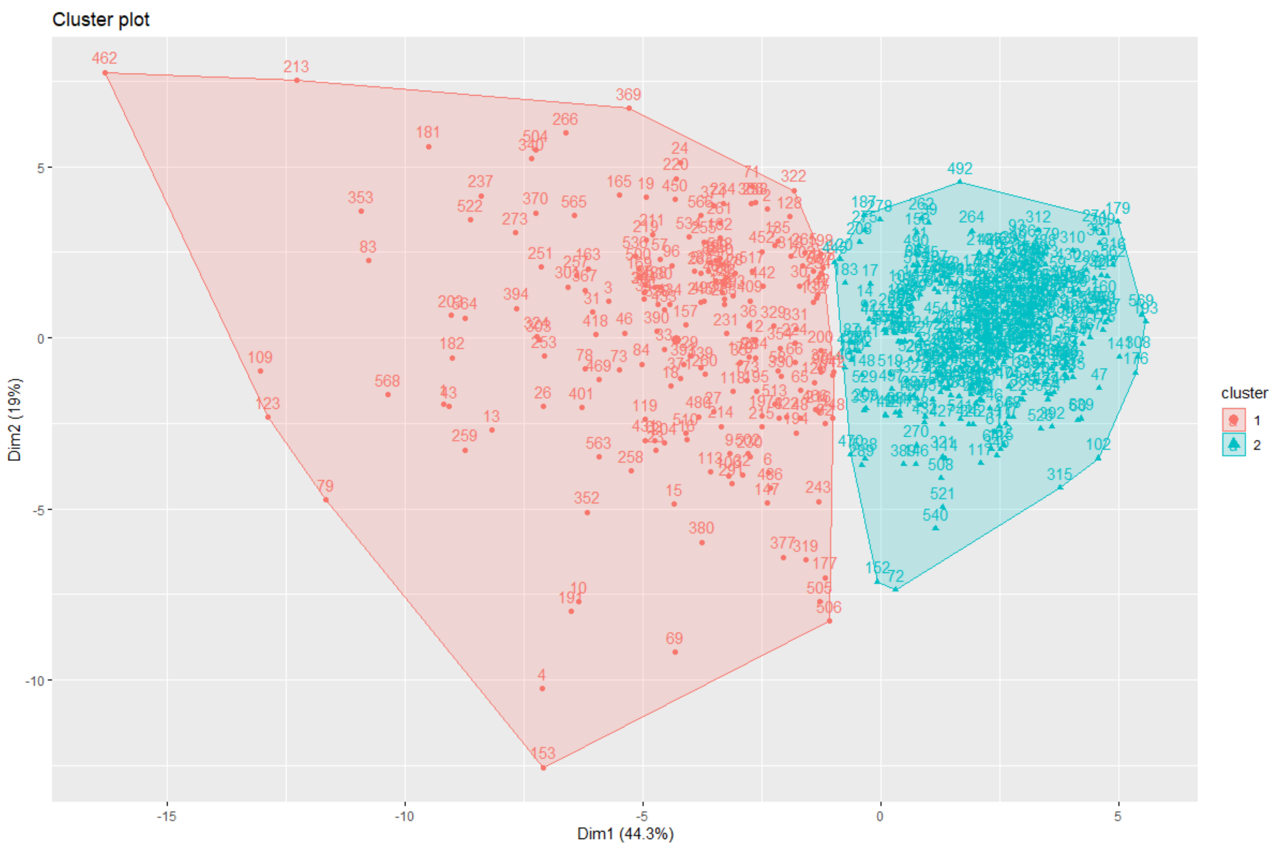
- The first two principal components explain a significant amount of the variation in the data and can be used to reduce the dimensionality of the dataset.
- The first principal component is mainly associated with variables related to the size and shape of the cell nuclei, while the second principal component is mainly associated with variables related to the texture and smoothness of the cell nuclei.
- The biplot shows clear clusters of observations that are potentially related to the diagnosis of the breast tumour, with malignant observations generally located in the upper left quadrant and benign observations located in the lower right quadrant.
- The insights gained from the PCA analysis can be used to inform further analyses, such as predictive modelling or clustering, that may help to better understand the relationships between the variables and observations in the data.
- However, it's important to note that the PCA analysis is only one tool for exploring and understanding the data and should be interpreted in conjunction with other analyses and domain knowledge in order to draw robust and accurate conclusions.

Cluster analysis: Partitioning (or partitional) clustering methods: K-means



```r
library(tidyverse)
library(cluster)
library(factoextra)
library(magrittr)

#Remove the first column (ID)
data <- data[, -c(1)]
#Convert the diagnosis column to a factor variable
data$V2 <- as.factor(as.character(data$V2))
#Normalize the data
data_norm <- scale(data[, -1])
#Perform k-means clustering with k = 2 (because we know there are two possible outcomes for the diagnosis column)
set.seed(123)
km <- kmeans(data_norm, 2)
#Cluster centers
print(km$centers)
#Visualize the clusters
fviz_cluster(km, data = data_norm)
#Calculate silhouette scores for each sample in the dataset
sil_scores <- silhouette(km$cluster, dist(data_norm))
#Mean silhouette score for the entire dataset
mean_sil_score <- mean(sil_scores[, 3])
```

1. First, we remove the first column (ID) from the dataset, since we don’t need it for the clustering analysis.
2. We convert the new first column (V2, diagnosis) to a factor variable, so R can recognize it as categorical variable.
3. Normalize the data using scale() function. Now all variables have the same scale and are equally important in the cluster analysis, they all have mean 0 and sd 1.
4. Perform the k-means algorithm with k = 2, since we know that there are two possible outcomes for the diagnosis column (benign or malignant). To ensure reproducibility of the results we set the seed.
5. The purpose of setting the seed is to ensure that the random number generator used by the k-means algorithm produces the same results each time the code is run.
6. Setting a seed is useful for reproducibility, which means that you can obtain the same results each time you run the code. This can be particularly important when working with data that has a high degree of randomness, such as clustering. By setting the seed, you can ensure that your results are consistent and can be reproduced.
7. In this particular case, setting the seed to 123 is an arbitrary choice. You could choose any other value for the seed and still obtain the same results each time the code is run, as long as you use the same seed each time.
8. Print the cluster centers, which are the means of each variable for each cluster.
9. Visualize the clusters using the fviz_cluster() function from the factoextra library. This function creates a scatter plot of the data points, colored by cluster membership.
10. We can now calculate the silhouette coefficient to evaluate the quality of the clustering. We calculate the silhouette scores for each sample in the dataset using silhouette() function from cluster package. The two arguments of the function are the cluster assignments (km$cluster) and the distance matrix (dist(data_norm).
11. We calculate the mean silhouette score for the entire dataset by taking the average of the third column of the sil_scores matrix, which contains the actual silhouette scores. We can now read, round and interpret the score.



Silhouette validation

Silhouette analysis measures the quality of the clustering by computing a silhouette coefficient for each sample, which measures how well the sample fits in its assigned cluster.

The mean silhouette score can range from -1 to 1, with a higher score indicating better defined cluster, a score closes to 0 to indicates overlapping or poorly separated cluster, while a negative score indicates that the samples may have been assigned to the wrong cluster.

In our case the Silhouette score is 0,34, indicates that the data points within the clusters are somewhat similar to each other but also somewhat similar to data points in other cluster. This suggests that the clustering algorithm has some degree of separation between cluster as we can see in the plot above.

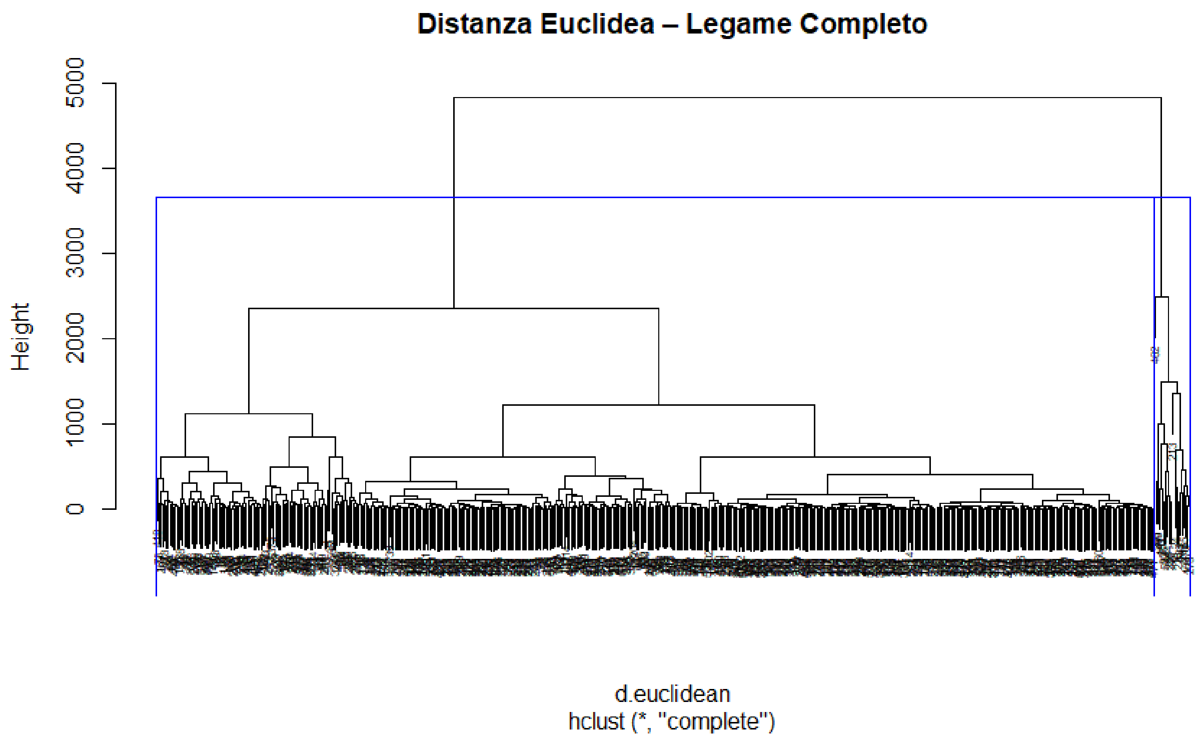
Cluster analysis: agglomerative hierarchical clustering

First of all we compute different distances:

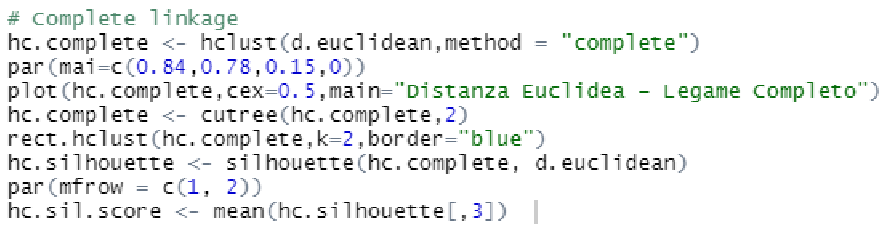
```
d.euclidean <- dist(data, method = "euclidean")
d.manhattan <- dist(data, method = "manhattan")
d.maximum   <- dist(data, method = "maximum")
d.canberra  <- dist(data, method = "canberra")
d.minkowski <- dist(data, method = "minkowski",p=2)  # p=2: distanza euclidea
```

We can perform the analysis using those distances and one of the different linkage methods, in this case I will use only Euclidean distance and I'll compute the analysis 5 times using different linkage methods: complete linkage, single linkage, average linkage, centroid and WARD. I will validate the results using "Silhouette score".

Complete linkage - Euclidean



Now I can compute the Silhouette scores to validate the analysis: the score is 0.69, very close to 1, so we can accept the results. This is the code I used to compute this model:



```
# Complete linkage
hc.complete <- hclust(d.euclidean,method = "complete")
par(mai=c(0.84,0.78,0.15,0))
plot(hc.complete,cex=0.5,main="Distanza Euclidea - Legame Completo")
hc.complete <- cutree(hc.complete,2)
rect.hclust(hc.complete,k=2,border="blue")
hc.silhouette <- silhouette(hc.complete, d.euclidean)
par(mfrow = c(1, 2))
hc.sil.score <- mean(hc.silhouette[,3])
```

hc.complete <- hclust(d.euclidean, method = "complete"): This code performs hierarchical clustering on a distance matrix d.euclidean using the complete linkage method. The resulting dendrogram is stored in the hc.complete variable.

par(mai=c(0.84,0.78,0.15,0)): This code sets the margin parameters of the plotting region in the current device. In this case, it adjusts the margin sizes to ensure that the dendrogram is plotted with an appropriate size and aspect ratio.

plot(hc.complete, cex = 0.5, main = "Distanza Euclidea – Legame Completo"): This code plots the dendrogram stored in hc.complete. The cex argument controls the size of the labels on the dendrogram, and the main argument provides a title for the plot.

hc.complete <- cutree(hc.complete, 2): This code cuts the dendrogram at a height that produces two clusters and assigns each observation to one of the two clusters. The resulting cluster assignments are stored in hc.complete.

rect.hclust(hc.complete, k = 2, border = "blue"): This code adds a rectangular border to the dendrogram to visually highlight the two clusters found in the previous step.

hc.silhouette <- silhouette(hc.complete, d.euclidean): This code calculates the silhouette width for each observation based on the clustering stored in hc.complete. The resulting silhouette widths are stored in hc.silhouette.

par(mfrow = c(1, 2)): This code sets the layout of the plotting region to have one row and two columns, so that two plots can be shown side by side.

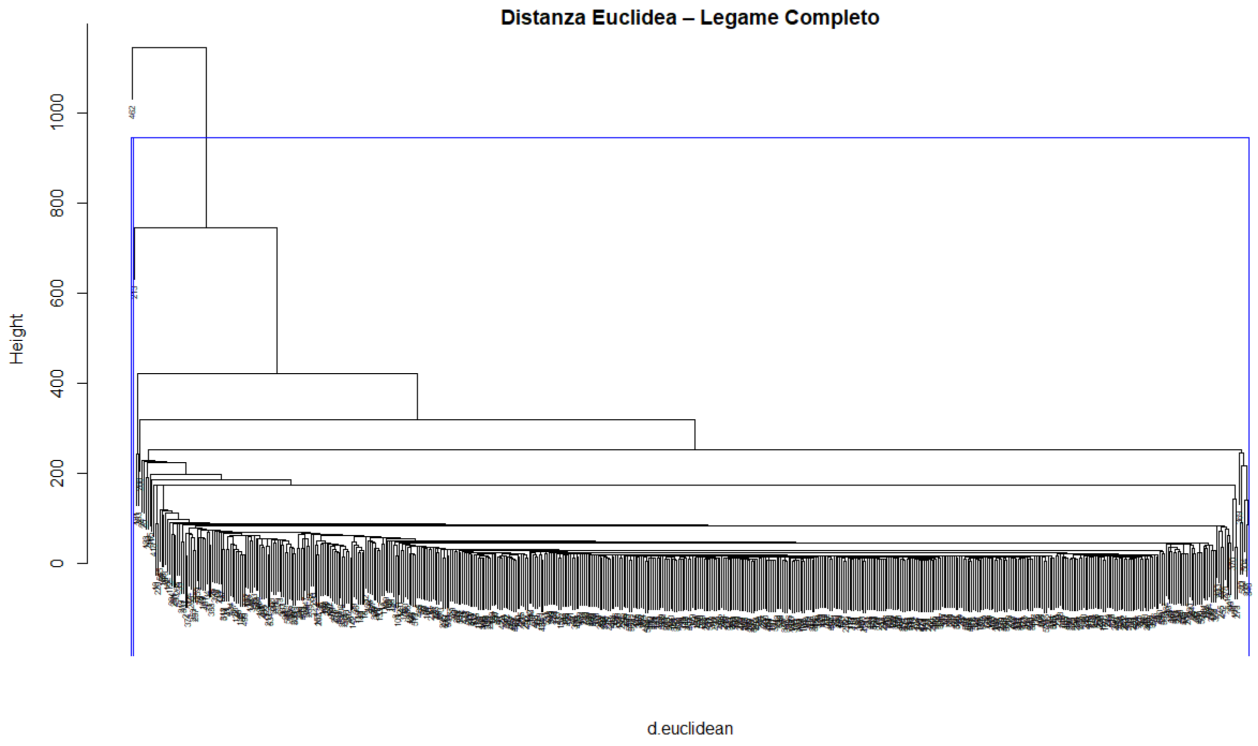
hc.sil.score <- mean(hc.silhouette[, 3]): This code calculates the mean silhouette width across all observations in the clustering stored in hc.complete. The resulting value is stored in hc.sil.score.

Overall, this code performs hierarchical clustering on a set of observations using the complete linkage method, cuts the resulting dendrogram to form two clusters, visualizes the clusters using a dendrogram plot with a rectangular border, calculates the silhouette width for each observation in the clustering, and calculates the mean silhouette width across all observations.

Single linkage

The same code used before can be modified to compute the analysis using other linkage type like single linkage:

```
#single linkage
hc.single <- hclust(d.euclidean,method = "single")
par(mai=c(0.84,0.78,0.15,0))
plot(hc.single,cex=0.5,main="Distanza Euclidea - Legame Completo")
rect.hclust(hc.single,k=2,border="blue")
hc.single <- cutree(hc.single,2)
hc.silhouette <- silhouette(hc.single, d.euclidean)
par(mfrow = c(1, 2))
hc.sil.score <- mean(hc.silhouette[,3])
```



The main difference between complete linkage and single linkage lies in the way they measure the distance between two clusters. Complete linkage clustering measures the distance between the two clusters as the maximum distance between any pair of points from the two clusters. In other words, it measures the similarity of the two clusters based on the similarity of their most dissimilar members. On the other hand, single linkage clustering measures the distance between the two clusters as the minimum distance between any pair of points from the two clusters. In other words, it measures the similarity of the two clusters based on the similarity of their closest members.

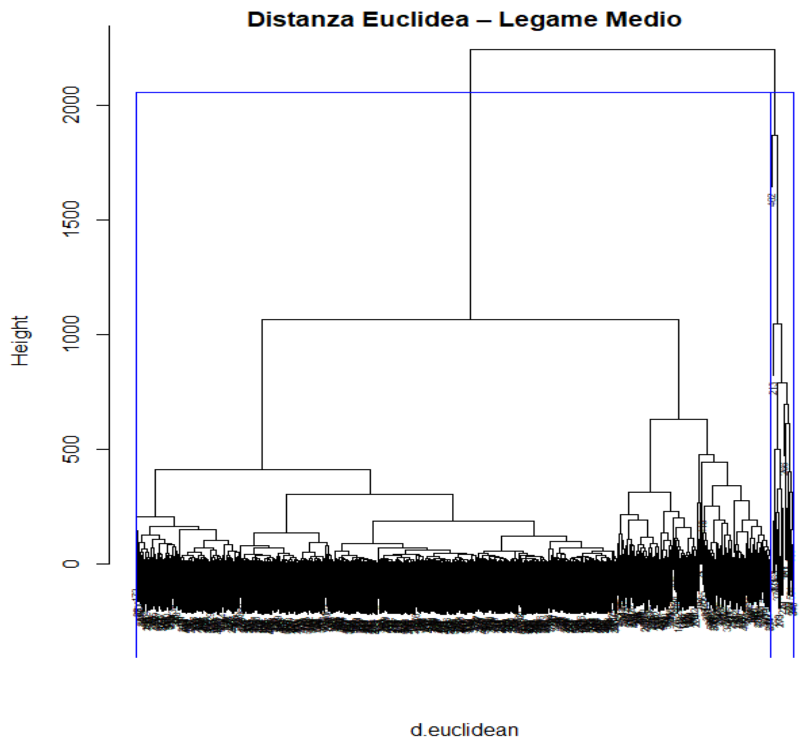
Here are some key differences between complete linkage and single linkage clustering:

- Sensitivity to Outliers: Complete linkage clustering is sensitive to outliers, as it measures the similarity between two clusters based on their most dissimilar members. Single linkage clustering is less sensitive to outliers, as it measures the similarity between two clusters based on their closest members.
- Cluster Shape: Complete linkage clustering tends to produce more compact, spherical clusters, whereas single linkage clustering can produce elongated or irregularly shaped clusters.
- Computational Complexity: Complete linkage clustering is computationally more expensive than single linkage clustering, as it involves calculating the maximum distance between any pair of points from the two clusters.
- Interpretability: Single linkage clustering is more interpretable than complete linkage clustering, as it produces clusters that are based on the closest members, which are easier to understand and interpret.

In this case the Silhouette score is equal to 0.7990, so we can validate the results.

Average linkage

```
#average linkage
hc.average <- hclust(d.euclidean,method = "average")
par(mai=c(0.84,0.78,0.15,0))
plot(hc.average,cex=0.5,main="Distanza Euclidea - Legame Medio")
cutree(hc.average,2)
rect.hclust(hc.average,k=2,border="blue")
hc.silhouette <- silhouette(hc.single, d.euclidean)
par(mfrow = c(1, 2))
hc.sil.score <- mean(hc.silhouette[,3])
```



```
> summary(hc.silhouette)
Silhouette of 569 units in 2 clusters from silhouet
te.default(x = hc.average, dist = d.euclidean) :
 Cluster sizes and average silhouette widths:
      549        20
0.6925051 0.6478387
Individual silhouette widths:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.4832  0.7209  0.8094  0.6909  0.8320  0.8386
```

Silhouette score equal to 0.6909, result validated.

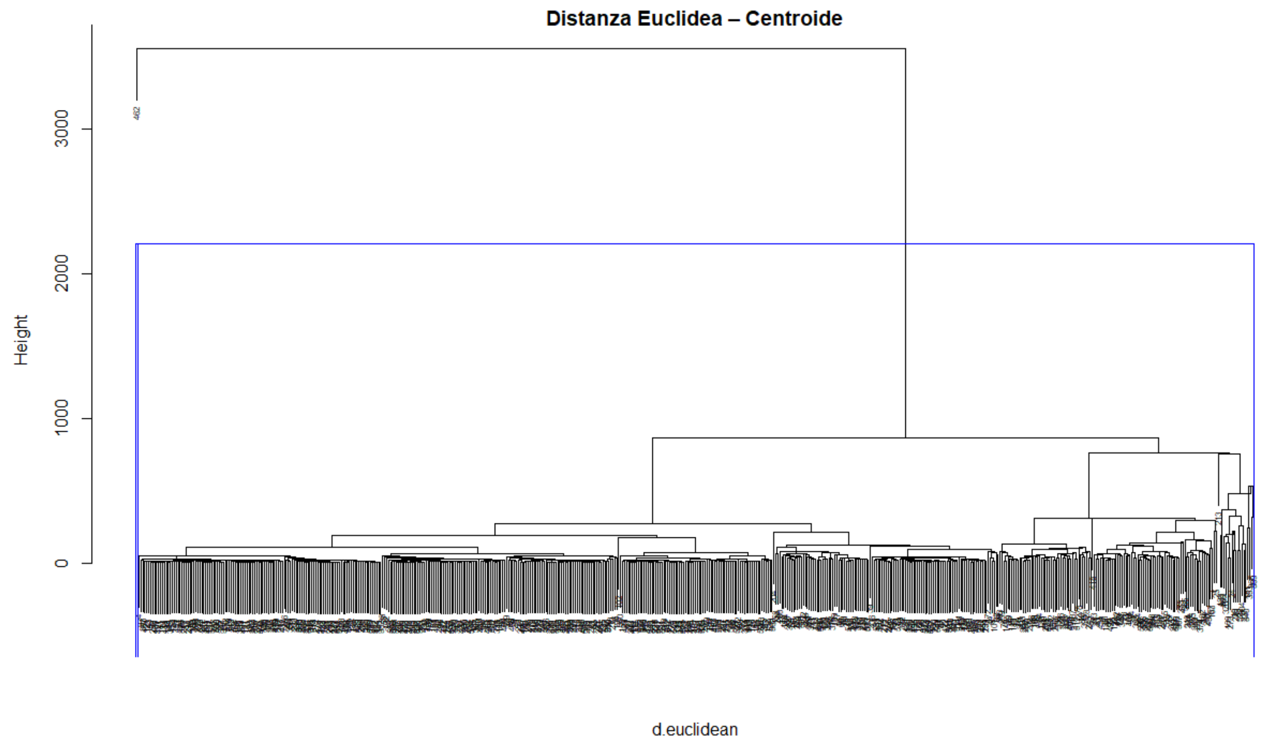
Average linkage is another commonly used method in hierarchical clustering, which calculates the distance between two clusters as the average distance between all pairs of points in the two clusters.

Here are some key differences between average linkage and complete linkage and single linkage clustering:

- Sensitivity to Outliers: Average linkage clustering is less sensitive to outliers than complete linkage clustering but more sensitive to outliers than single linkage clustering.
- Cluster Shape: Average linkage clustering can produce more elongated or irregularly shaped clusters compared to complete linkage clustering but less elongated or irregularly shaped clusters compared to single linkage clustering.
- Computational Complexity: Average linkage clustering is computationally less expensive than complete linkage clustering but more expensive than single linkage clustering.
- Interpretability: Average linkage clustering is more interpretable than complete linkage clustering but less interpretable than single linkage clustering, as it produces clusters based on the average distance between all pairs of points, which can be harder to understand and interpret compared to the closest or most dissimilar members.

Centroid

```
#centroid
hc.centroid <- hclust(d.euclidean,method = "centroid")
par(mai=c(0.84,0.78,0.15,0))
plot(hc.centroid,cex=0.5,main="Distanza Euclidea - Centroide")
hc.centroid <- cutree(hc.centroid,2)
rect.hclust(hc.centroid,k=2,border="blue")
hc.silhouette <- silhouette(hc.centroid, d.euclidean)
```



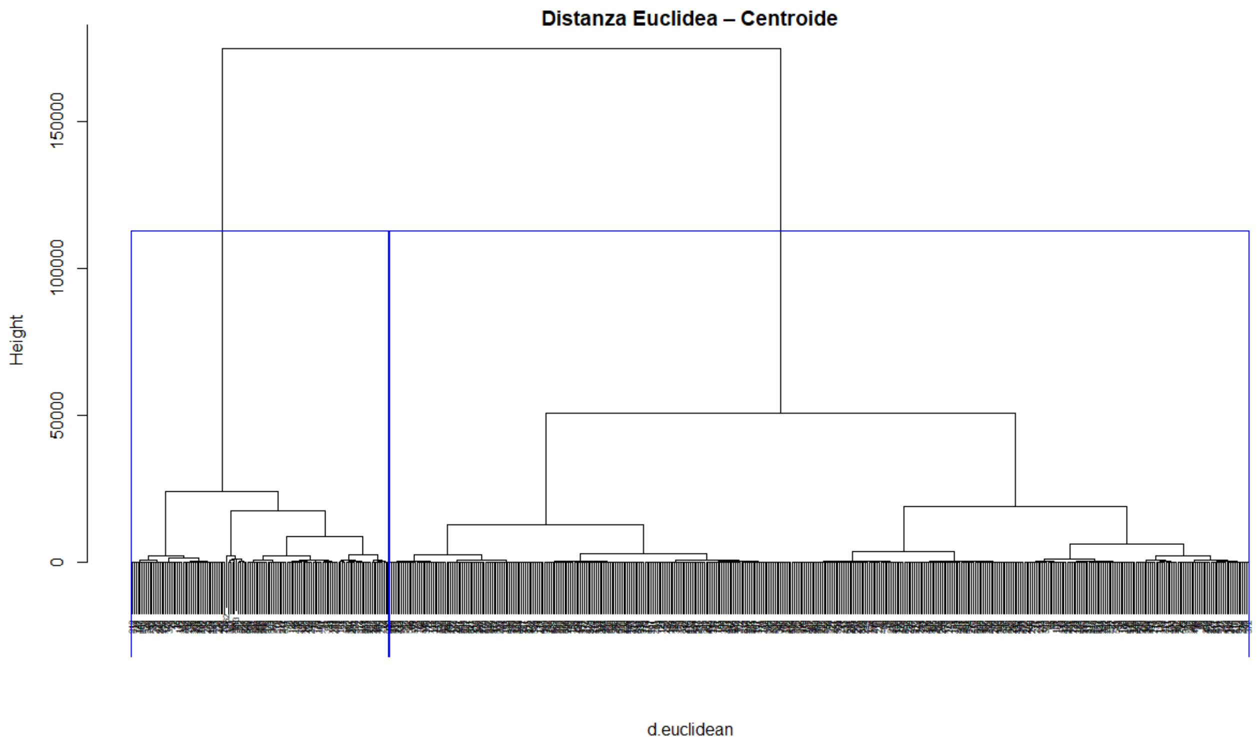
```
> summary(hc.silhouette)
Silhouette of 569 units in 2 clusters from silhouet
te.default(x = hc.centroid, dist = d.euclidean) :
 Cluster sizes and average silhouette widths:
      568         1
0.8003935 0.0000000
Individual silhouette widths:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.5979  0.8292  0.8738  0.7990  0.8852  0.8891
```

Silhouette score equals to 0.7990, we can validate the results.

The centroid method is a commonly used algorithm in hierarchical clustering. In this method, the distance between two clusters is calculated as the distance between their centroids, which are the means of their individual data points. One advantage of the centroid method is that it is relatively fast and can be used with a large number of data points. However, it can be sensitive to outliers and can sometimes result in non-intuitive cluster assignments. Additionally, the use of the mean as a measure of central tendency assumes that the data points are normally distributed, which may not always be the case.

Ward

```
hc.ward <- hclust(d.euclidean,method = "ward.D")
par(mai=c(0.84,0.78,0.15,0))
plot(hc.ward,cex=0.5,main="Distanza Euclidea - Centroide")
hc.ward <- cutree(hc.ward,2)
rect.hclust(hc.ward,k=2,border="blue")
hc.silhouette <- silhouette(hc.ward, d.euclidean)
```



```
> summary(hc.silhouette)
Silhouette of 569 units in 2 clusters from silhouet
te.default(x = hc.ward, dist = d.euclidean) :
 Cluster sizes and average silhouette widths:
      131       438
0.4788485 0.7625900
Individual silhouette widths:
    Min.  1st Qu.   Median     Mean  3rd Qu.
-0.03129  0.62405  0.78694  0.69726  0.83643
    Max.
 0.84949
```

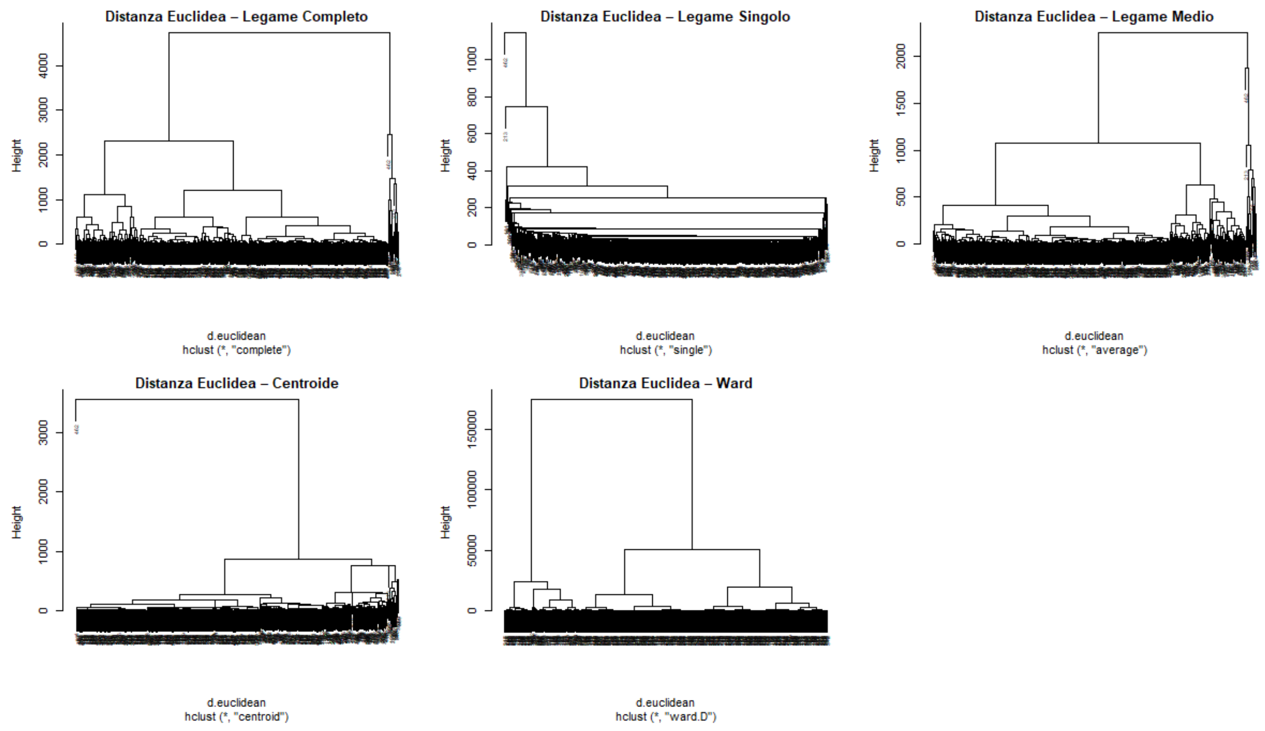
Silhouette score 0.697, we can validate the results.

Ward's method, also known as Ward.D or Ward's linkage, is a hierarchical clustering algorithm that aims to minimize the variance within each cluster. In Ward's method, the distance between two clusters is defined as the increase in the sum of squared deviations from the mean that results from merging the clusters. The algorithm starts with each data point as a separate cluster and then recursively merges the two closest clusters until there is only one cluster left.

Ward's method is known for its ability to produce tight, compact clusters that are relatively homogeneous. It is particularly useful when the goal is to find clusters that are similar in terms of their variance, since it optimizes this criterion directly. Additionally, because it is agglomerative, it is a bottom-up method that can be more efficient than divisive methods when dealing with large datasets.

One disadvantage of Ward's method is that it can be sensitive to outliers, as the algorithm tries to minimize the variance within each cluster. It can also be affected by the scaling of the data, since it is based on the sum of squared deviations. Additionally, the resulting clusters may not be easily interpretable in some cases, depending on the nature of the data and the research question.

Matrix of graphical represented results



Cluster analysis: Fuzzy Clustering

First of all I would like to introduce fuzzy set and the theory behind fuzzy clustering.
Fuzzy set theory is a mathematical framework that provides a way to handle uncertainty and imprecision in data analysis and decision making. Unlike traditional sets that classify elements as either belonging or not belonging to a set, fuzzy sets allow for degrees of membership based on how well an element matches a set's characteristics or criteria. Fuzzy sets are particularly useful in situations where there is ambiguity or vagueness in the data, such as in natural language processing, image recognition, and decision support systems. They are also used in control systems and artificial intelligence applications to model complex and uncertain relationships between variables.

Fuzzy clustering is a technique used in data science to group similar data points into clusters, allowing for uncertainty in the assignment of data points to clusters. Unlike traditional clustering methods where a data point is assigned to a single cluster, fuzzy clustering allows a data point to belong to multiple clusters to a certain degree.
In fuzzy clustering, each data point is assigned a membership value that represents the degree to which it belongs to each cluster. These membership values can be interpreted as degrees of confidence. The goal of fuzzy clustering is to find the optimal number and shape of clusters that best represent the data, given the uncertainty and variability in the data. Fuzzy clustering has applications in a wide range of fields, including image segmentation, pattern recognition, market segmentation, and customer profiling. It is particularly useful in situations where data points have multiple characteristics or dimensions, and the boundaries between clusters are not well-defined. Overall, fuzzy clustering is a powerful tool in data science for uncovering hidden patterns and relationships in complex data sets, and for making more informed and accurate decisions based on uncertain or imprecise data.

Fuzzy k-means FkM

The aim of the FkM algorithm is to look for the best fuzzy partition of n units into k clusters by solving the minimization problem:

$$\begin{aligned}
&\min_{\mathbf{U},\mathbf{H}} J_{\mathrm{F}k\mathrm{M}} = \sum_{i=1}^{n} \sum_{g=1}^{k} u_{ig}^{m} d^{2}\left(\mathbf{x}_i, \mathbf{h}_g\right), \\
&\text{s.t.} \quad u_{ig} \in [0,1], \quad i = 1, \ldots, n, \quad g = 1, \ldots, k, \\
&\qquad \sum_{g=1}^{k} u_{ig} = 1, \quad i = 1, \ldots, n,
\end{aligned}$$

Where the term u_{ig} represents the membership degree of unit i to cluster g taking values [0,1]. The row-wise sums of U are equal to 1. The parameter m is used to tune the fuzziness of the obtained partition.

The optimal solution can be found by means of the following iterative algorithm:

1. Choose a feasible membership degree matrix U.
2. Given U, update the centroid matrix H:

$$\mathbf{h}_g = \frac{\sum_{i=1}^{n} u_{ig}^{m} \mathbf{x}_i}{\sum_{i=1}^{n} u_{ig}^{m}}, \quad g = 1, \ldots, k.$$

3. Given H, update the membership degree U:

$$u_{ig} = \frac{1}{\sum_{g'=1}^{k} \left(\frac{d^2(\mathbf{x}_i, \mathbf{h}_g)}{d^2(\mathbf{x}_i, \mathbf{h}_{g'})} \right)^{\frac{1}{m-1}}}, \quad i = 1, \ldots, n, \quad g = 1, \ldots, k.$$

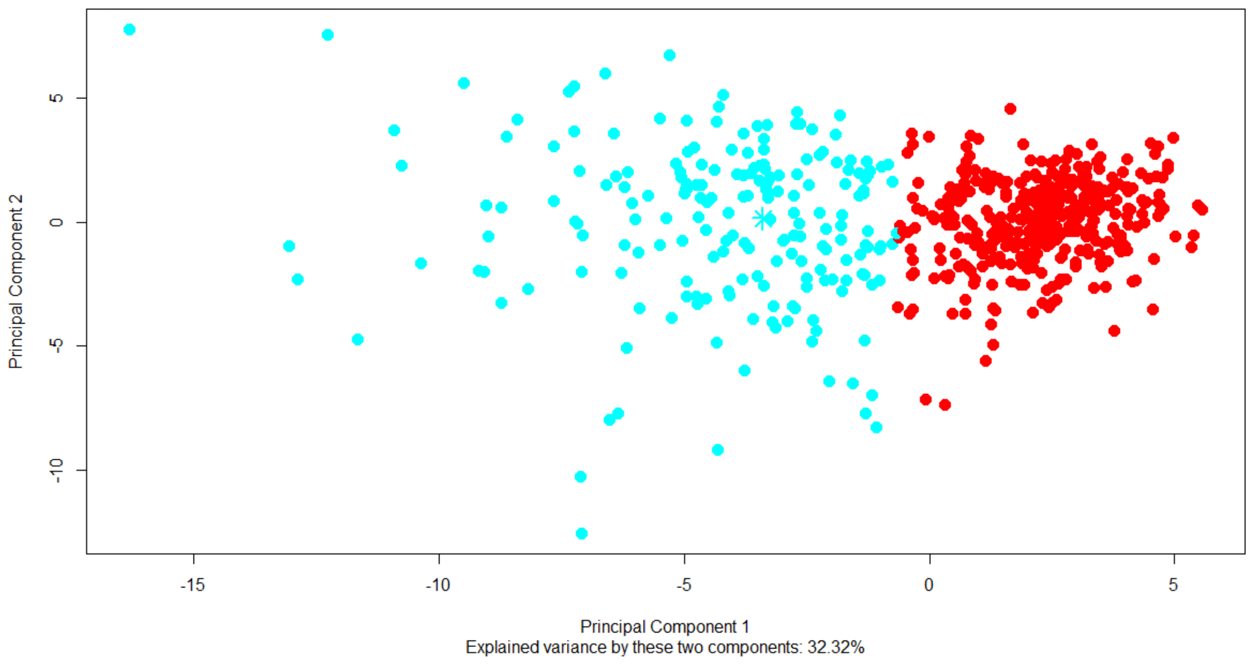
4. Repeat 2 and 3 until convergence is reached.

As in the previous analysis k is supposed to be 2 because we want to divide M and B cancer.

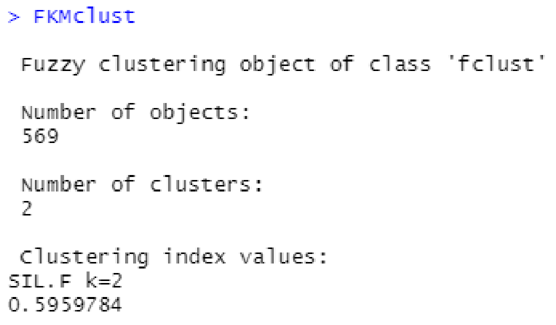
In fuzzy k-means clustering, where a data point can belong to multiple clusters with varying degrees of membership, the calculation of silhouette becomes more complex. One approach to calculating fuzzy silhouette is to use a weighted average of the silhouette values for each cluster that the data point belongs to, weighted by its membership values. Fuzzy silhouette can provide insights into the quality and coherence of the clusters generated by fuzzy k-means and can be used to compare the performance of different clustering algorithms or parameter settings. However, it is important to note that fuzzy silhouette is just one of several measures that can be used to evaluate the quality of clustering results.

FkM: The analysis

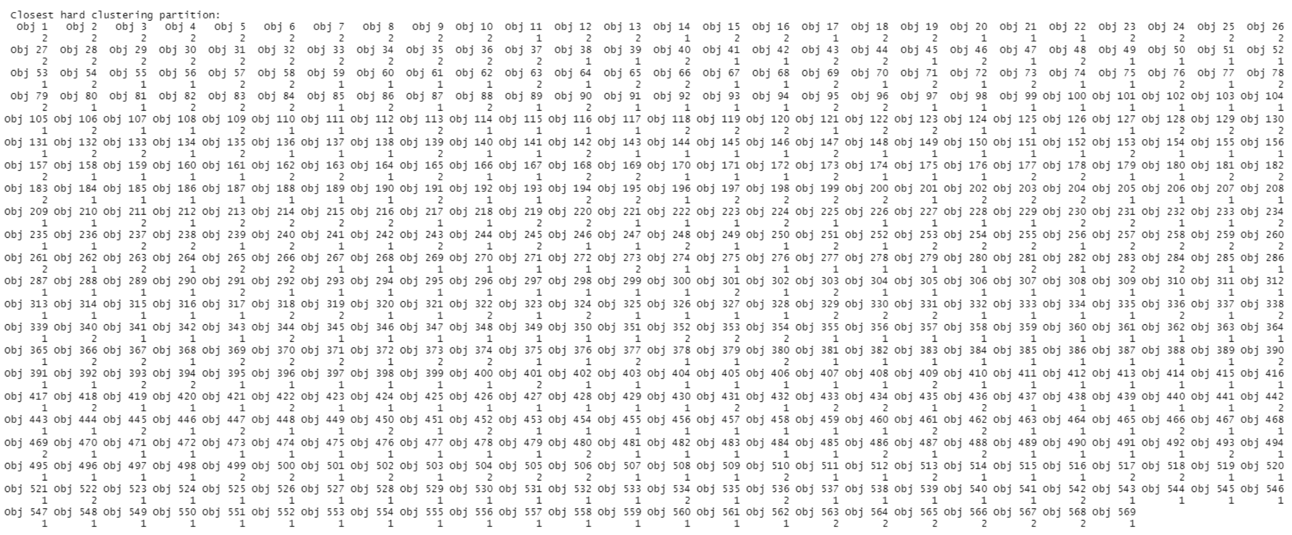
```
#Fuzzy k-means
library(fclust)
data <- scale(data)
FKMclust <- FKM(X = data, k = 2)
fuzzySilhouetteFS <- FKMclust$criterion
cluster.size <- cl.size(FKMclust$U)
Hraw(FKMclust$X, FKMclust$H)
plot.fclust(x = FKMclust, pca = TRUE)
```



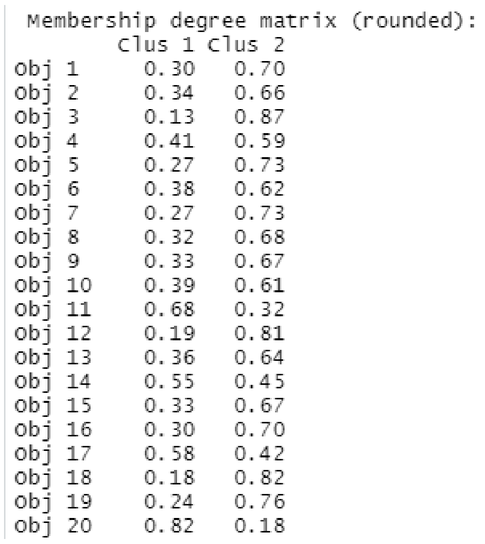
>FKMclust



```
> FKMclust

 Fuzzy clustering object of class 'fclust'

 Number of objects:
 569

 Number of clusters:
 2

 Clustering index values:
SIL.F k=2
0.5959784
```

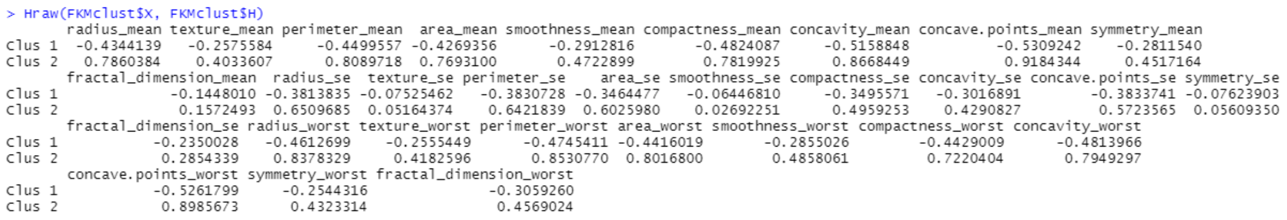


```
 Closest hard clustering partition:
   Obj 1   Obj 2   Obj 3   Obj 4   Obj 5   Obj 6   Obj 7   Obj 8   Obj 9  Obj 10  Obj 11  Obj 12  Obj 13  Obj 14  Obj 15  Obj 16  Obj 17  Obj 18  Obj 19  Obj 20  Obj 21  Obj 22  Obj 23  Obj 24  Obj 25  Obj 26
       2       2       2       2       2       2       2       2       2       2       1       2       2       1       2       2       1       2       2       1       1       1       2       2       2       2
  Obj 27  Obj 28  Obj 29  Obj 30  Obj 31  Obj 32  Obj 33  Obj 34  Obj 35  Obj 36  Obj 37  Obj 38  Obj 39  Obj 40  Obj 41  Obj 42  Obj 43  Obj 44  Obj 45  Obj 46  Obj 47  Obj 48  Obj 49  Obj 50  Obj 51  Obj 52
       2       2       2       2       2       2       2       2       2       2       2       1       1       2       1       1       2       2       1       2       1       2       1       1       1       1
  Obj 53  Obj 54  Obj 55  Obj 56  Obj 57  Obj 58  Obj 59  Obj 60  Obj 61  Obj 62  Obj 63  Obj 64  Obj 65  Obj 66  Obj 67  Obj 68  Obj 69  Obj 70  Obj 71  Obj 72  Obj 73  Obj 74  Obj 75  Obj 76  Obj 77  Obj 78
       1       2       1       1       2       2       1       1       1       1       2       1       2       2       1       1       2       1       2       1       2       1       1       2       1       2
  Obj 79  Obj 80  Obj 81  Obj 82  Obj 83  Obj 84  Obj 85  Obj 86  Obj 87  Obj 88  Obj 89  Obj 90  Obj 91  Obj 92  Obj 93  Obj 94  Obj 95  Obj 96  Obj 97  Obj 98  Obj 99 Obj 100 Obj 101 Obj 102 Obj 103 Obj 104
       2       1       1       2       2       2       1       2       1       2       1       2       1       1       1       1       2       2       1       1       1       1       1       1       1       1
 Obj 105 Obj 106 Obj 107 Obj 108 Obj 109 Obj 110 Obj 111 Obj 112 Obj 113 Obj 114 Obj 115 Obj 116 Obj 117 Obj 118 Obj 119 Obj 120 Obj 121 Obj 122 Obj 123 Obj 124 Obj 125 Obj 126 Obj 127 Obj 128 Obj 129 Obj 130
       1       2       1       1       2       1       1       1       2       1       1       1       1       2       2       2       1       2       2       1       1       1       1       2       2       2
 Obj 131 Obj 132 Obj 133 Obj 134 Obj 135 Obj 136 Obj 137 Obj 138 Obj 139 Obj 140 Obj 141 Obj 142 Obj 143 Obj 144 Obj 145 Obj 146 Obj 147 Obj 148 Obj 149 Obj 150 Obj 151 Obj 152 Obj 153 Obj 154 Obj 155 Obj 156
       1       2       2       1       2       1       1       1       2       1       1       2       1       1       1       1       2       1       1       1       1       1       2       1       1       1
 Obj 157 Obj 158 Obj 159 Obj 160 Obj 161 Obj 162 Obj 163 Obj 164 Obj 165 Obj 166 Obj 167 Obj 168 Obj 169 Obj 170 Obj 171 Obj 172 Obj 173 Obj 174 Obj 175 Obj 176 Obj 177 Obj 178 Obj 179 Obj 180 Obj 181 Obj 182
       2       1       1       1       1       2       2       1       2       1       1       2       2       1       1       1       2       1       1       1       2       2       1       1       2       2
 Obj 183 Obj 184 Obj 185 Obj 186 Obj 187 Obj 188 Obj 189 Obj 190 Obj 191 Obj 192 Obj 193 Obj 194 Obj 195 Obj 196 Obj 197 Obj 198 Obj 199 Obj 200 Obj 201 Obj 202 Obj 203 Obj 204 Obj 205 Obj 206 Obj 207 Obj 208
       2       1       1       1       1       1       1       1       2       1       1       2       2       1       2       2       2       2       1       2       2       2       1       1       1       1
 Obj 209 Obj 210 Obj 211 Obj 212 Obj 213 Obj 214 Obj 215 Obj 216 Obj 217 Obj 218 Obj 219 Obj 220 Obj 221 Obj 222 Obj 223 Obj 224 Obj 225 Obj 226 Obj 227 Obj 228 Obj 229 Obj 230 Obj 231 Obj 232 Obj 233 Obj 234
       1       1       2       1       2       2       2       2       1       1       2       2       1       1       1       2       1       1       1       1       1       2       2       1       1       2
 Obj 235 Obj 236 Obj 237 Obj 238 Obj 239 Obj 240 Obj 241 Obj 242 Obj 243 Obj 244 Obj 245 Obj 246 Obj 247 Obj 248 Obj 249 Obj 250 Obj 251 Obj 252 Obj 253 Obj 254 Obj 255 Obj 256 Obj 257 Obj 258 Obj 259 Obj 260
       1       1       2       2       1       2       1       1       2       1       2       1       1       2       1       1       2       1       2       2       2       1       2       2       2       2
 Obj 261 Obj 262 Obj 263 Obj 264 Obj 265 Obj 266 Obj 267 Obj 268 Obj 269 Obj 270 Obj 271 Obj 272 Obj 273 Obj 274 Obj 275 Obj 276 Obj 277 Obj 278 Obj 279 Obj 280 Obj 281 Obj 282 Obj 283 Obj 284 Obj 285 Obj 286
       2       1       2       1       2       2       1       1       1       1       1       1       2       1       1       1       1       1       1       1       2       1       2       2       1       1
 Obj 287 Obj 288 Obj 289 Obj 290 Obj 291 Obj 292 Obj 293 Obj 294 Obj 295 Obj 296 Obj 297 Obj 298 Obj 299 Obj 300 Obj 301 Obj 302 Obj 303 Obj 304 Obj 305 Obj 306 Obj 307 Obj 308 Obj 309 Obj 310 Obj 311 Obj 312
       1       1       1       1       2       1       1       1       1       1       1       1       1       1       2       1       2       1       1       1       1       1       1       1       1       1
 Obj 313 Obj 314 Obj 315 Obj 316 Obj 317 Obj 318 Obj 319 Obj 320 Obj 321 Obj 322 Obj 323 Obj 324 Obj 325 Obj 326 Obj 327 Obj 328 Obj 329 Obj 330 Obj 331 Obj 332 Obj 333 Obj 334 Obj 335 Obj 336 Obj 337 Obj 338
       1       1       1       1       1       2       2       1       1       2       1       2       1       1       1       1       2       2       2       1       1       1       1       2       1       2
 Obj 339 Obj 340 Obj 341 Obj 342 Obj 343 Obj 344 Obj 345 Obj 346 Obj 347 Obj 348 Obj 349 Obj 350 Obj 351 Obj 352 Obj 353 Obj 354 Obj 355 Obj 356 Obj 357 Obj 358 Obj 359 Obj 360 Obj 361 Obj 362 Obj 363 Obj 364
       1       2       1       1       1       2       1       1       1       1       1       1       1       2       2       2       1       1       1       1       1       1       1       1       1       1
 Obj 365 Obj 366 Obj 367 Obj 368 Obj 369 Obj 370 Obj 371 Obj 372 Obj 373 Obj 374 Obj 375 Obj 376 Obj 377 Obj 378 Obj 379 Obj 380 Obj 381 Obj 382 Obj 383 Obj 384 Obj 385 Obj 386 Obj 387 Obj 388 Obj 389 Obj 390
       1       2       2       1       2       2       2       1       2       2       1       1       2       1       1       2       1       1       1       1       1       1       1       1       1       2
 Obj 391 Obj 392 Obj 393 Obj 394 Obj 395 Obj 396 Obj 397 Obj 398 Obj 399 Obj 400 Obj 401 Obj 402 Obj 403 Obj 404 Obj 405 Obj 406 Obj 407 Obj 408 Obj 409 Obj 410 Obj 411 Obj 412 Obj 413 Obj 414 Obj 415 Obj 416
       1       1       2       2       1       1       1       1       1       1       2       1       1       1       1       1       1       1       2       1       1       1       1       1       1       1
 Obj 417 Obj 418 Obj 419 Obj 420 Obj 421 Obj 422 Obj 423 Obj 424 Obj 425 Obj 426 Obj 427 Obj 428 Obj 429 Obj 430 Obj 431 Obj 432 Obj 433 Obj 434 Obj 435 Obj 436 Obj 437 Obj 438 Obj 439 Obj 440 Obj 441 Obj 442
       1       2       1       1       1       2       1       1       1       1       1       1       1       1       2       1       2       2       1       2       1       1       1       1       1       2
 Obj 443 Obj 444 Obj 445 Obj 446 Obj 447 Obj 448 Obj 449 Obj 450 Obj 451 Obj 452 Obj 453 Obj 454 Obj 455 Obj 456 Obj 457 Obj 458 Obj 459 Obj 460 Obj 461 Obj 462 Obj 463 Obj 464 Obj 465 Obj 466 Obj 467 Obj 468
       1       1       2       1       2       1       1       2       1       2       1       1       1       1       1       1       1       1       2       2       1       1       1       2       1       1
 Obj 469 Obj 470 Obj 471 Obj 472 Obj 473 Obj 474 Obj 475 Obj 476 Obj 477 Obj 478 Obj 479 Obj 480 Obj 481 Obj 482 Obj 483 Obj 484 Obj 485 Obj 486 Obj 487 Obj 488 Obj 489 Obj 490 Obj 491 Obj 492 Obj 493 Obj 494
       2       1       1       1       1       1       1       1       1       1       1       2       1       1       1       1       1       2       1       2       1       1       1       1       2       1
 Obj 495 Obj 496 Obj 497 Obj 498 Obj 499 Obj 500 Obj 501 Obj 502 Obj 503 Obj 504 Obj 505 Obj 506 Obj 507 Obj 508 Obj 509 Obj 510 Obj 511 Obj 512 Obj 513 Obj 514 Obj 515 Obj 516 Obj 517 Obj 518 Obj 519 Obj 520
       1       1       1       1       2       2       1       2       1       2       2       2       1       1       1       2       1       1       2       1       1       1       2       2       1       1
 Obj 521 Obj 522 Obj 523 Obj 524 Obj 525 Obj 526 Obj 527 Obj 528 Obj 529 Obj 530 Obj 531 Obj 532 Obj 533 Obj 534 Obj 535 Obj 536 Obj 537 Obj 538 Obj 539 Obj 540 Obj 541 Obj 542 Obj 543 Obj 544 Obj 545 Obj 546
       1       2       1       1       1       1       1       1       1       1       1       1       1       2       1       2       1       1       1       1       1       2       1       1       1       1
 Obj 547 Obj 548 Obj 549 Obj 550 Obj 551 Obj 552 Obj 553 Obj 554 Obj 555 Obj 556 Obj 557 Obj 558 Obj 559 Obj 560 Obj 561 Obj 562 Obj 563 Obj 564 Obj 565 Obj 566 Obj 567 Obj 568 Obj 569
       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       2       2       2       2       2       2       1
```



```
 Membership degree matrix (rounded):
       Clus 1 Clus 2
Obj 1    0.30   0.70
Obj 2    0.34   0.66
Obj 3    0.13   0.87
Obj 4    0.41   0.59
Obj 5    0.27   0.73
Obj 6    0.38   0.62
Obj 7    0.27   0.73
Obj 8    0.32   0.68
Obj 9    0.33   0.67
Obj 10   0.39   0.61
Obj 11   0.68   0.32
Obj 12   0.19   0.81
Obj 13   0.36   0.64
Obj 14   0.55   0.45
Obj 15   0.33   0.67
Obj 16   0.30   0.70
Obj 17   0.58   0.42
Obj 18   0.18   0.82
Obj 19   0.24   0.76
Obj 20   0.82   0.18
```

Omitted 549 rows.



```
> Hraw(FKMclust$X, FKMclust$H)
       radius_mean texture_mean perimeter_mean  area_mean smoothness_mean compactness_mean concavity_mean concave.points_mean symmetry_mean
Clus 1  -0.4344139   -0.2575584     -0.4499557 -0.4269356      -0.2912816       -0.4824087     -0.5158848          -0.5309242    -0.2811540
Clus 2   0.7860384    0.4033607      0.8089718  0.7693100       0.4722899        0.7819925      0.8668449           0.9184344     0.4517164
       fractal_dimension_mean  radius_se  texture_se perimeter_se    area_se smoothness_se compactness_se concavity_se concave.points_se symmetry_se
Clus 1             -0.1448010 -0.3813835 -0.07525462   -0.3830728 -0.3464477   -0.06446810     -0.3495571   -0.3016891        -0.3833741 -0.07623903
Clus 2              0.1572493  0.6509685  0.05164374    0.6421839  0.6025980    0.02692251      0.4959253    0.4290827         0.5723565  0.05609350
       fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst
Clus 1           -0.2350028   -0.4612699    -0.2555449      -0.4745411 -0.4416019       -0.2855026        -0.4429009      -0.4813966
Clus 2            0.2854339    0.8378329     0.4182596       0.8530770  0.8016800        0.4858061         0.7220404       0.7949297
       concave.points_worst symmetry_worst fractal_dimension_worst
Clus 1           -0.5261799     -0.2544316              -0.3059260
Clus 2            0.8985673      0.4323314               0.4569024
```

In order to interpret and characterize the obtained clusters, the centroid features are reported. For example, cluster 2 have bigger radius_mean and perimeter_mean etc.

Gustafson-Kessel Fuzzy k-Means

In the FkM algorithm, in in standard k-means, the Euclidean distance is used. This leads to spherical clusters. In the presence of non-spherical clusters, the FkM may fail to properly recognize the clusters. To overcome this limitation, Gustafson and Kessel propose to extend the FkM algorithm by replacing the Euclidean distance by a cluster specific Mahalanobis distance:

$$d_M^2(\mathbf{x}_i, \mathbf{h}_g) = (\mathbf{x}_i - \mathbf{h}_g)' \mathbf{M}_g (\mathbf{x}_i - \mathbf{h}_g)$$

Where M_g is a symmetric and positive-definite matrix. Notice that the distance is equal to the Euclidean when M_g is the identity matrix. The optimization problem of the Gustafson-Kessel-type Fuzzy k-Means can be written as follows:

$$\begin{aligned}
&\min_{\mathbf{U},\mathbf{H},\mathbf{M}_1,\ldots,\mathbf{M}_k} J_{\mathrm{GK-FkM}} = \sum_{i=1}^{n} \sum_{g=1}^{k} u_{ig}^m d_M^2(\mathbf{x}_i, \mathbf{h}_g),\\
&\text{s.t.}\quad u_{ig} \in [0,1], \quad i = 1,\ldots,n, \quad g = 1,\ldots,k,\\
&\qquad \sum_{g=1}^{k} u_{ig} = 1, \quad i = 1,\ldots,n,\\
&\qquad |\mathbf{M}_g| = \rho_g > 0, \quad g = 1,\ldots,k.
\end{aligned}$$

The optimal solution can be found by means of the following iterative algorithm:

1. Rationally or randomly choose a feasible membership degree matrix $\mathbf{U}$.
2. Given $\mathbf{U}$ and $\mathbf{M}_g$, $g = 1\ldots,k$, update the centroid matrix $\mathbf{H}$:

$$\mathbf{h}_g = \frac{\sum_{i=1}^{n} u_{ig}\mathbf{x}_i}{\sum_{i=1}^{n} u_{ig}}, \quad g = 1,\ldots,k.$$

3. Given $\mathbf{U}$ and $\mathbf{H}$, update the matrix $\mathbf{M}_g$, $g = 1,\ldots,k$,

$$\mathbf{M}_g = (|\boldsymbol{\Sigma}_g|)^{\frac{1}{p}} \boldsymbol{\Sigma}_g^{-1},$$

   where

$$\boldsymbol{\Sigma}_g = \frac{\sum_{i=1}^{n} u_{ig}^m (\mathbf{x}_i - \mathbf{h}_g)(\mathbf{x}_i - \mathbf{h}_g)'}{\sum_{i=1}^{n} u_{ig}^m}$$

   is the fuzzy covariance matrix of the g-th cluster.
4. Given $\mathbf{H}$ and $\mathbf{M}_g$, $g = 1\ldots,k$, update the membership degree matrix $\mathbf{U}$:

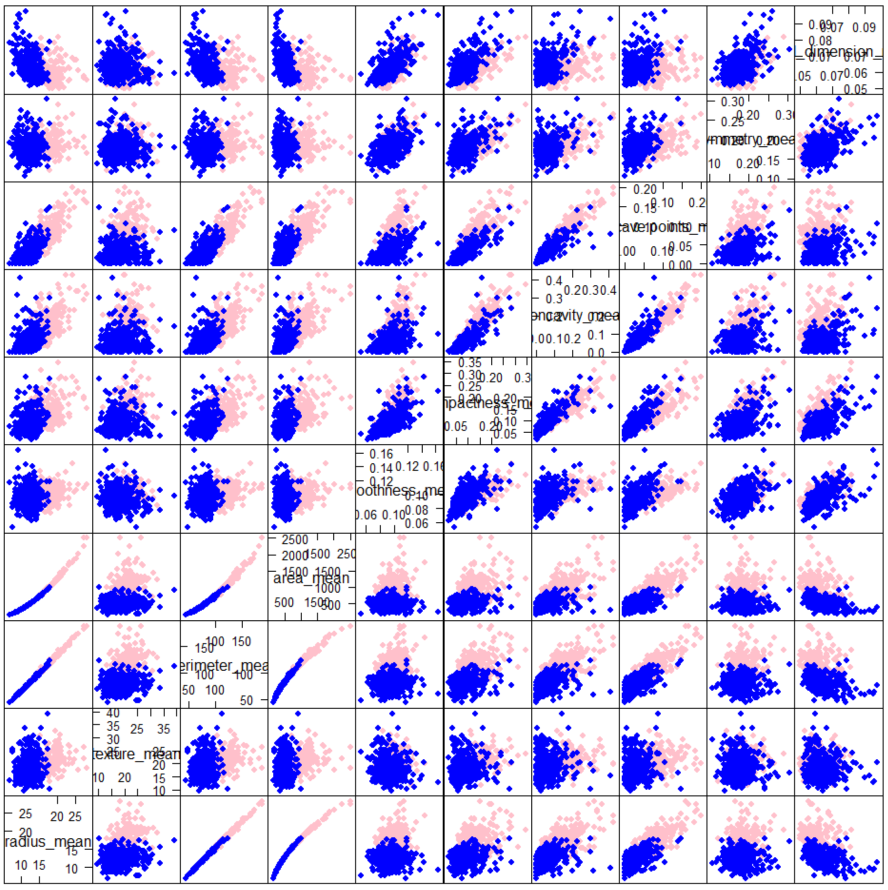
$$u_{ig} = \frac{1}{\sum_{g'=1}^{k} \left(\frac{d_M^2(\mathbf{x}_i,\mathbf{h}_g)}{d_M^2(\mathbf{x}_i,\mathbf{h}_{g'})} \right)^{\frac{1}{m-1}}}, \quad i = 1,\ldots,n, \quad g = 1,\ldots,k.$$

5. Repeat steps 2, 3, and 4 until convergence is reached.

The eigenvalues and eigenvectors of Σ_g describe the shape and orientation of the g^{th} cluster.

Analysis

```
#gustafson-Kessel Fuzzy k-Means
data <- data[,-1]
library(fclust)
GKFKM <- FKM.gk(data, k=2)
library(lattice)
plot <- splom(data, groups = GKFKM$clus[,1],
       pch=19, col = c("pink","blue"))
```



Matrice Scatter Plot

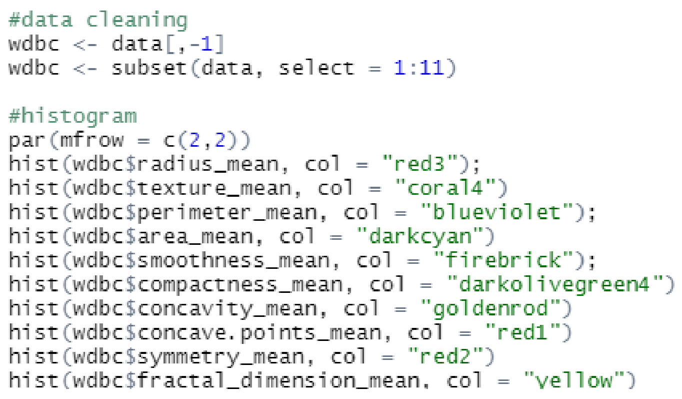
Exploratory Analysis

To better understand the analysis it’s important to have a look to our data by a short exploratory analysis in which I studied and plotted the distribution of diagnosis and “mean” variables.

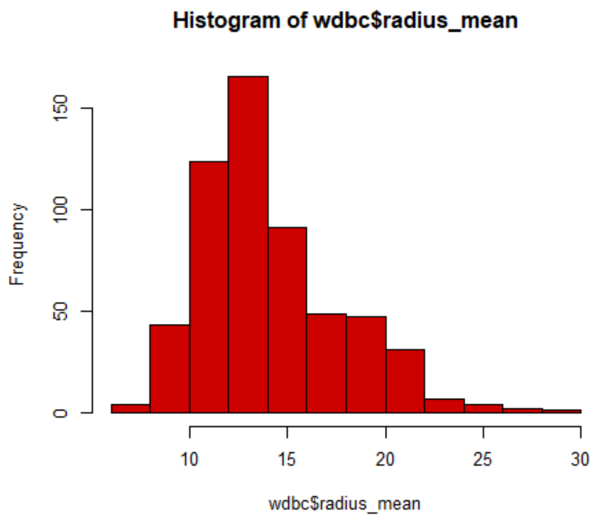
The data contains 569 observations of patients with breast cancer in a Wisconsin hospital. My goal is to use the 'diagnosis' column to see which variables are significant to identify whether a patient's diagnosis is 'benign' or 'malignant'. In this analysis I will go into basic charts to gather any interesting insight.

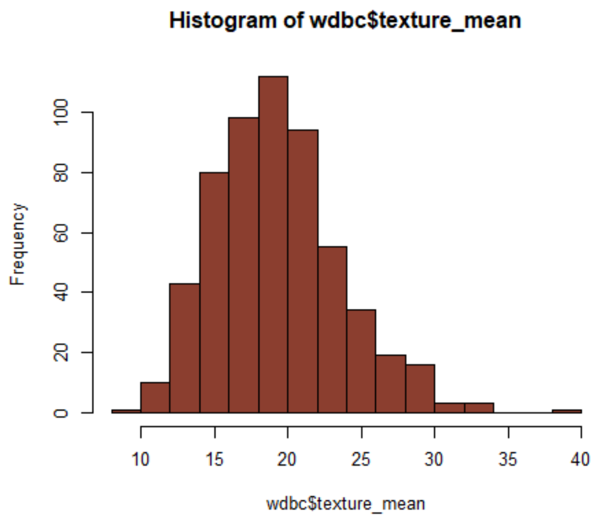
First of all, I remove some columns that I am not interested in, like ID. I only need diagnosis and “mean” variables. All the variables are numeric, except “diagnosis” who is a chr wo can be converted to binary (M and B as 0 and 1).

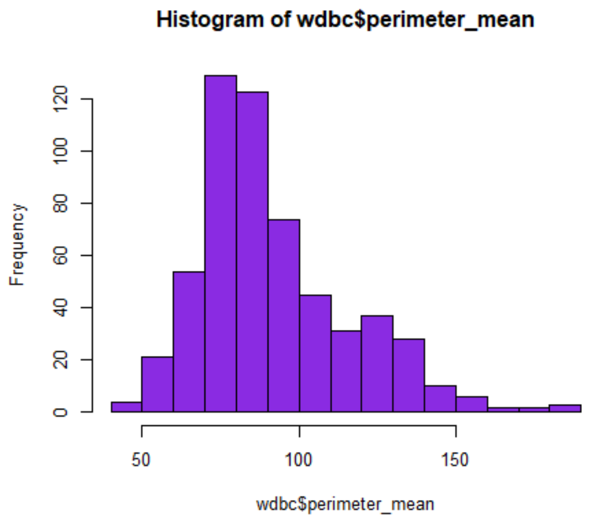
To have a first look of our data we can make some histogram of overall variables:

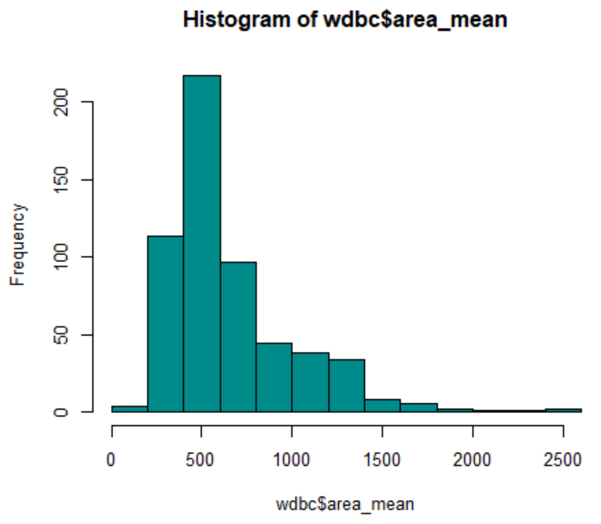


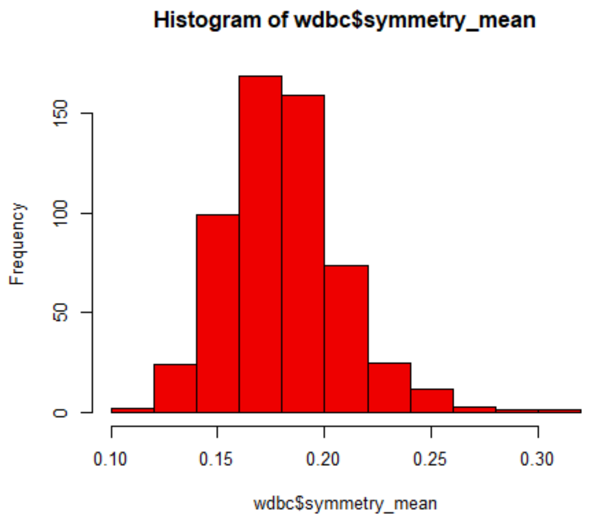
```
#data cleaning
wdbc <- data[,-1]
wdbc <- subset(data, select = 1:11)

#histogram
par(mfrow = c(2,2))
hist(wdbc$radius_mean, col = "red3");
hist(wdbc$texture_mean, col = "coral4")
hist(wdbc$perimeter_mean, col = "blueviolet");
hist(wdbc$area_mean, col = "darkcyan")
hist(wdbc$smoothness_mean, col = "firebrick");
hist(wdbc$compactness_mean, col = "darkolivegreen4")
hist(wdbc$concavity_mean, col = "goldenrod")
hist(wdbc$concave.points_mean, col = "red1")
hist(wdbc$symmetry_mean, col = "red2")
hist(wdbc$fractal_dimension_mean, col = "yellow")
```

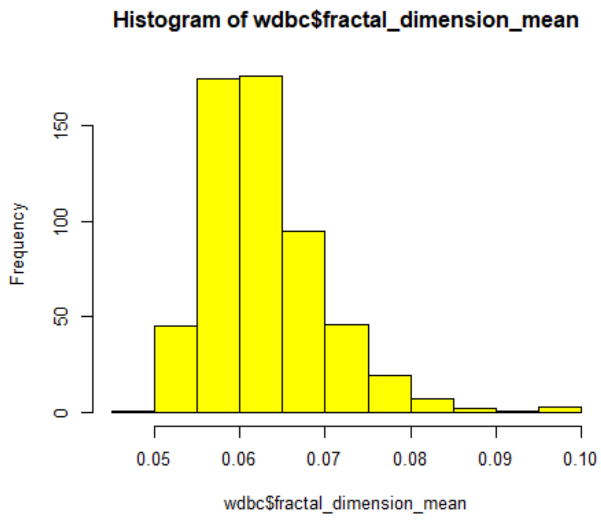

Histogram of wdbc$radius_mean
Frequency
0
50
100
150
10
15
20
25
30
wdbc$radius_mean

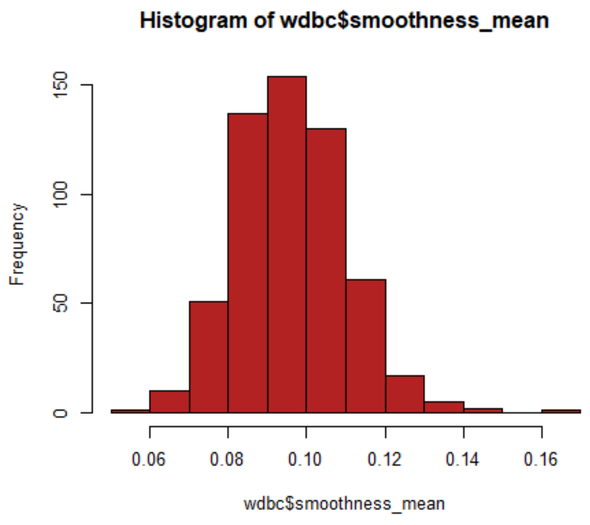

Histogram of wdbc$texture_mean
Frequency
0
20
40
60
80
100
10
15
20
25
30
35
40
wdbc$texture_mean

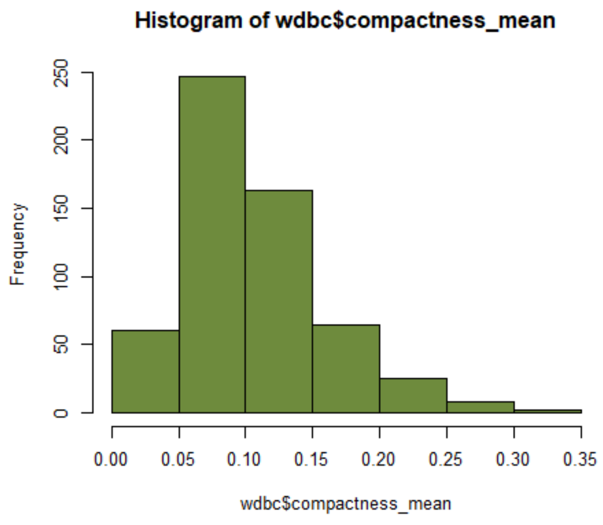

Histogram of wdbc$perimeter_mean
Frequency
0
20
40
60
80
100
120
50
100
150
wdbc$perimeter_mean

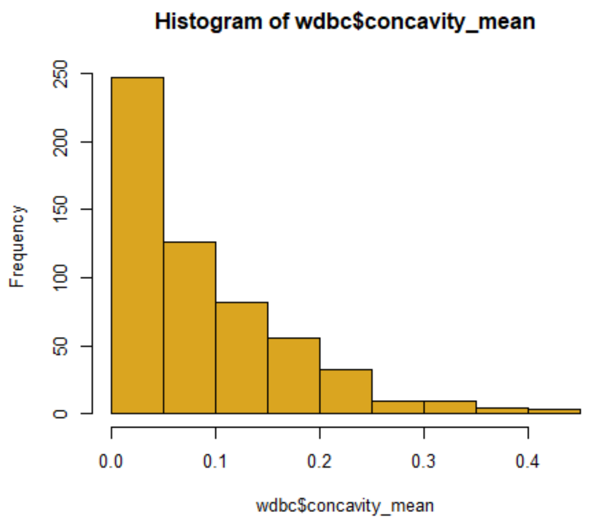

Histogram of wdbc$area_mean
Frequency
0
50
100
150
200
0
500
1000
1500
2000
2500
wdbc$area_mean

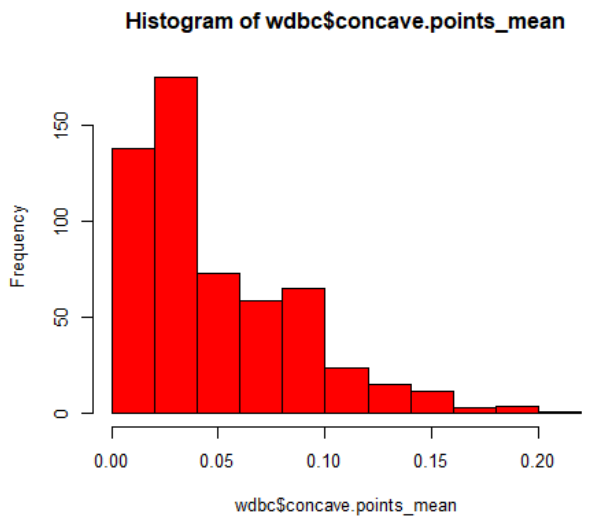

Histogram of wdbc$symmetry_mean
Frequency
0
50
100
150
0.10
0.15
0.20
0.25
0.30
wdbc$symmetry_mean


Histogram of wdbc$fractal_dimension_mean
Frequency
0
50
100
150
0.05
0.06
0.07
0.08
0.09
0.10
wdbc$fractal_dimension_mean



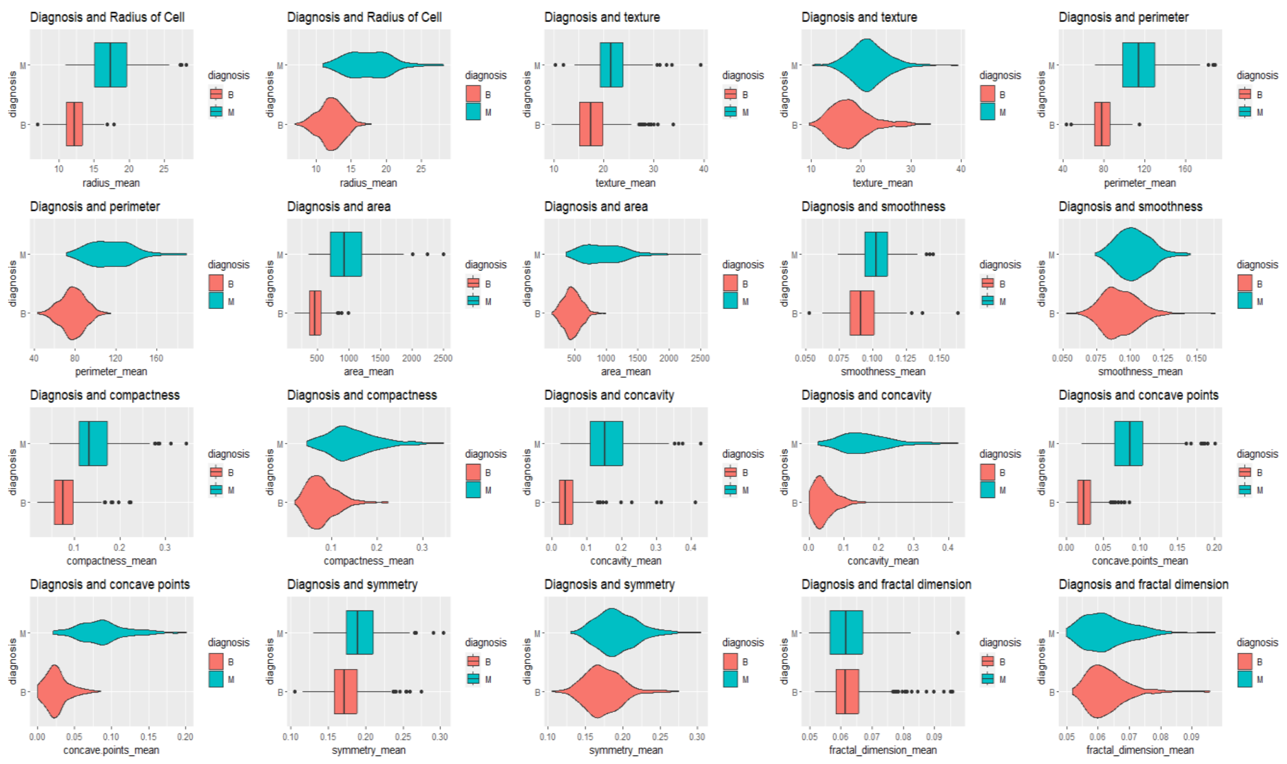






May be also interesting to see the difference in distribution between “benign” and “malignant” diagnosis. A good way to see this is using boxplots and violin charts in “ggplot2”. Boxplots helps us to see basic max, min, quartile ranges, median and also outliers to take note, violin plots show the distribution of each variables:

```
#boxplot and violin plots
library(tidyverse);
library(ggpubr)

p1 <- ggplot(wdbc, mapping = aes(x = diagnosis, y = radius_mean, fill = diagnosis)) +
  geom_boxplot() +
  labs(title = "Diagnosis and Radius of Cell") +
  coord_flip()

p11 <- ggplot(wdbc, mapping = aes(x = diagnosis, y = radius_mean, fill = diagnosis)) +
  geom_violin() +
  labs(title = "Diagnosis and Radius of Cell") +
  coord_flip()

p2 <- ggplot(wdbc, mapping = aes(x = diagnosis, y = texture_mean, fill = diagnosis)) +
  geom_boxplot() +
  labs(title = "Diagnosis and texture") +
  coord_flip()

p22 <- ggplot(wdbc, mapping = aes(x = diagnosis, y = texture_mean, fill = diagnosis)) +
  geom_violin() +
  labs(title = "Diagnosis and texture") +
  coord_flip()

...
...

ggarrange(p1,p11, p2,p22, p3,p33, p4,p44, p5,p55, p6,p66, p7,p77, p8,p88, p9,p99, p10,p1010)
```


Diagnosis and Radius of Cell
Diagnosis and Radius of Cell
Diagnosis and texture
Diagnosis and texture
Diagnosis and perimeter
Diagnosis and perimeter
Diagnosis and area
Diagnosis and area
Diagnosis and smoothness
Diagnosis and smoothness
Diagnosis and compactness
Diagnosis and compactness
Diagnosis and concavity
Diagnosis and concavity
Diagnosis and concave points
Diagnosis and concave points
Diagnosis and symmetry
Diagnosis and symmetry
Diagnosis and fractal dimension
Diagnosis and fractal dimension
diagnosis
B
M
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave.points_mean
symmetry_mean
fractal_dimension_mean