UNIVERSITÀ
degli STUDI
di CATANIA

DATA ANALYSIS REPORT

SEEDS DATASET

Amoroso Fabio ID 1000023899

CHAPTER 1: DESCRIPTION OF THE DATA

The dataset Seeds is available in the library "datasetsICR" and it tells us some information about 210 kernels, belonging from 3 different varieties of wheat. The elements were randomly selected, 70 elements each, from the three varieties: Kama, Rosa and Canadian. The studies were conducted using a soft X-ray technique, that it's non-destructive and allow to record images on 13x18 cm X-ray KODAK plates. The dataset contains 210 observations of 8 variables: area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient, length of kernel groove and variety. We can see the structure of the dataset using the str function, as follows:

```
'data.frame':  210 obs. of  8 variables:
 $ area                  : num  15.3 14.9 14.3 13.8 16.1 ...
 $ perimeter             : num  14.8 14.6 14.1 13.9 15 ...
 $ compactness           : num  0.871 0.881 0.905 0.895 0.903 ...
 $ length of kernel      : num  5.76 5.55 5.29 5.32 5.66 ...
 $ width of kernel       : num  3.31 3.33 3.34 3.38 3.56 ...
 $ asymmetry coefficient : num  2.22 1.02 2.7 2.26 1.35 ...
 $ length of kernel groove: num  5.22 4.96 4.83 4.8 5.17 ...
 $ variety               : Factor w/ 3 levels "Kama","Rosa",..: 1
1 1 1 1 1 1 1 1 1 ...
```

As we can see, all the variables are numerical and continuous, except from variety that is a factor, based on three levels: Kama, Rosa and Canadian. The summary function gives us some information about each variable of the dataset.

```
      area         perimeter      compactness     length of kernel
 Min.   :10.59   Min.   :12.41   Min.   :0.8081   Min.   :4.899
 1st Qu.:12.27   1st Qu.:13.45   1st Qu.:0.8569   1st Qu.:5.262
 Median :14.36   Median :14.32   Median :0.8734   Median :5.524
 Mean   :14.85   Mean   :14.56   Mean   :0.8710   Mean   :5.629
 3rd Qu.:17.30   3rd Qu.:15.71   3rd Qu.:0.8878   3rd Qu.:5.980
 Max.   :21.18   Max.   :17.25   Max.   :0.9183   Max.   :6.675
 width of kernel asymmetry coefficient length of kernel groove     variety
 Min.   :2.630   Min.   :0.7651        Min.   :4.519           Kama    :70
 1st Qu.:2.944   1st Qu.:2.5615        1st Qu.:5.045           Rosa    :70
 Median :3.237   Median :3.5990        Median :5.223           Canadian:70
 Mean   :3.259   Mean   :3.7002        Mean   :5.408
 3rd Qu.:3.562   3rd Qu.:4.7687        3rd Qu.:5.877
 Max.   :4.033   Max.   :8.4560        Max.   :6.550
```

All the variables assume only positive values. Now we can perform the univariate analysis on each variable and find the models that better fits to each of them.

CHAPTER 2: UNIVARIATE ANALYSIS

The univariate analysis is useful to describe each variable, the values that it assumes and to know if there is a statistical model that fits it. A statistical model is a simplified representation of reality, that aim to represent the data generating process. I used the same process on every variable, except for "variety".

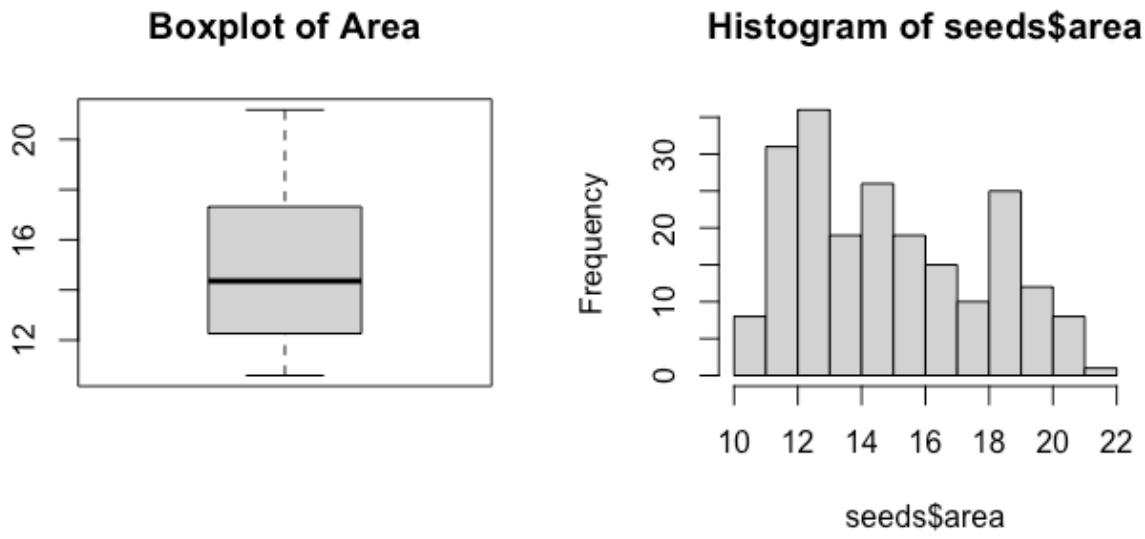
Univariate Analysis: Area

Apart from summary, I used the functions sd and var to have information about the variability of the data.

```
> summary(seeds$area)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.59   12.27   14.36   14.85   17.30   21.18
> sd(seeds$area)
[1] 2.909699
> var(seeds$area)
[1] 8.466351
```

Then, I used the boxplot function to have a plot containing the data and the previous information, and the hist function to visualize the data in a histogram.

```
> boxplot(seeds$area, main = "Boxplot of Area")
> boxplot(seeds$area)$out
numeric(0)
> hist(seeds$area)
```



We can see in the boxplot that there aren't outliers in the variable Area.
Now, we have to find the best model for our data. I used 5 different distribution to find the one that could fit better the data: Exponential, Gamma, Inverse Gaussian, Lognormal and Weibull. I used the function histDist of the gamlss package.

Exponential

```
> Area.EXP <- histDist(seeds$area, family = EXP, nbins = 30, main = "Area
Exponential Distribution")
> Area.EXP$df.fit
[1] 1
> fitted(Area.EXP, "mu")[1]
[1] 14.84752
> logLik(Area.EXP)
'log Lik.' -776.545 (df=1)
> AIC(Area.EXP)
[1] 1555.09
> Area.EXP$sbc
[1] 1558.437
```

Gamma

```
> Area.GA <- histDist(seeds$area, family=GA, nbins = 30, main="Area Gamma
distribution")
> Area.GA$df.fit
[1] 2
> fitted(Area.GA, "mu")[1]
[1] 14.84752
> fitted(Area.GA, "sigma")[1]
[1] 0.1931169
> logLik(Area.GA)
'log Lik.' -516.5516 (df=2)
> AIC(Area.GA)
[1] 1037.103
> Area.GA$sbc
[1] 1043.797
```

Inverse Gaussian

```
> Area.IG <- histDist(seeds$area, family=IG, nbins = 30, main="Area Inverse
Gaussian distribution")
> Area.IG$df.fit
[1] 2
> fitted(Area.IG, "mu")[1]
[1] 14.84752
> fitted(Area.IG, "sigma")[1]
[1] 0.05045369
> logLik(Area.IG)
'log Lik.' -514.6783 (df=2)
> AIC(Area.IG)
[1] 1033.357
> Area.IG$sbc
[1] 1040.051
```

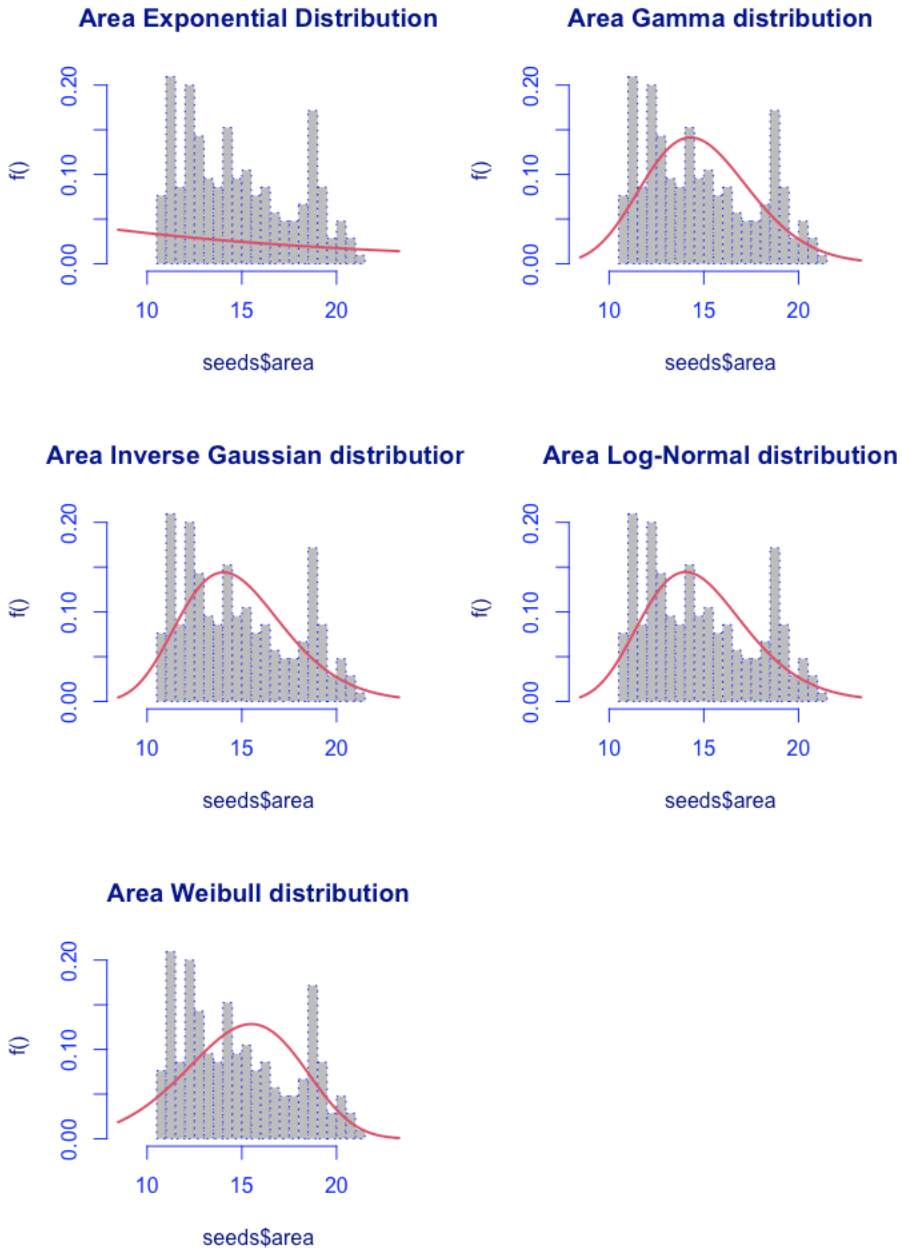
Lognormal

```
> Area.LOGNO <- histDist(seeds$area, family=LOGNO, nbins = 30, main="Area Log-
Normal distribution")
> Area.LOGNO$df.fit
[1] 2
> fitted(Area.LOGNO, "mu")[1]
[1] 2.67907
> fitted(Area.LOGNO, "sigma")[1]
[1] 0.192973
> logLik(Area.LOGNO)
'log Lik.' -515.089 (df=2)
> AIC(Area.LOGNO)
[1] 1034.178
> Area.LOGNO$sbc
[1] 1040.872
```

Weibull

```
> Area.WEI <- histDist(seeds$area, family=WEI, nbins = 30, main="Area Weibull
distribution")
> Area.WEI$df.fit
[1] 2
> fitted(Area.WEI, "mu")[1]
[1] 16.06708
> fitted(Area.WEI, "sigma")[1]
[1] 5.509844
> logLik(Area.WEI)
'log Lik.' -527.2807 (df=2)
> AIC(Area.WEI)
[1] 1058.561
> Area.WEI$sbc
[1] 1065.256
```

In this way, I found some important information about how these models fit to our data. For example, the LL (Log Likelihood), the AIC (Akaike information criterion) and the SBC or BIC (Bayes information criterion) are some important criterions that we can use to select the best model for our dataset. We have to minimize the AIC and the SBC and to maximize the LL to find the best model from the five proposed.



Now, we can compare the AIC, the SBC and the LL of every statistical model fitted to our data, in order to find the one that better describe our data.

```
> Comparing.Area <- data.frame(row.names = c("Exponential", "Gamma", "Inverse
Gaussian", "Log-Normal", "Weibull"), AIC=c(AIC(Area.EXP), AIC(Area.GA),
AIC(Area.IG), AIC(Area.LOGNO), AIC(Area.WEI)), SBC=c(Area.EXP$sbc, Area.GA$sbc,
Area.IG$sbc, Area.LOGNO$sbc, Area.WEI$sbc), LL=c(logLik(Area.EXP),
logLik(Area.GA), logLik(Area.IG), logLik(Area.LOGNO), logLik(Area.WEI)))
> Comparing.Area
                     AIC      SBC        LL
Exponential      1555.090 1558.437 -776.5450
Gamma            1037.103 1043.797 -516.5516
Inverse Gaussian 1033.357 1040.051 -514.6783
Log-Normal       1034.178 1040.872 -515.0890
Weibull          1058.561 1065.256 -527.2807
```

The best solution is the Inverse Gaussian Distribution with values of 1033.357 of AIc, 1040.051 of SBC and -514.6783 of LL. Another solution can be a mixture of two or three Inverse Gaussian, that could fit better to our data. We need the gamlss.mx package to create the mixture.

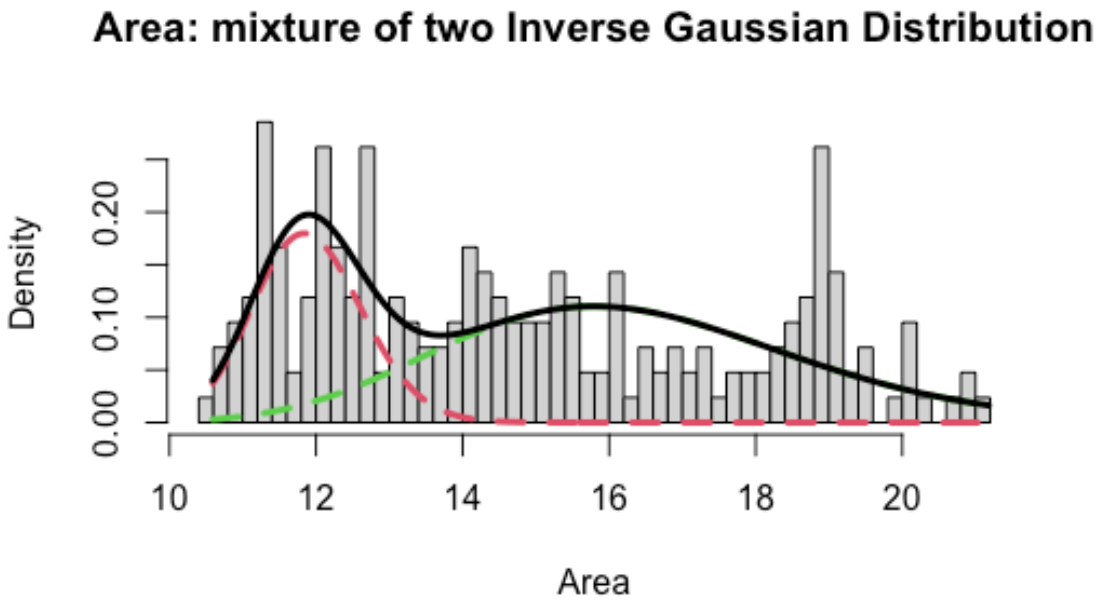
```
> Area.IG.2 <- gamlssMXfits(n = 5, seeds$area~1, family = IG, K = 2, data =
NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> Area.IG.2$aic
[1] 1000.055
> Area.IG.2$sbc
[1] 1016.791
```

The mixture of two Inverse Gaussian provides values of AIC and SBC better than using only an Inverse Gaussian, so we have to estimate the values of mu and sigma parameters to plot the mixture over the histogram of our data.

```
A.mu2.1 <- exp(Area.IG.2[["models"]][[1]][["mu.coefficients"]])
A.sigma2.1 < -exp(Area.IG.2[["models"]][[1]][["sigma.coefficients"]])

A.mu2.2 <- exp(Area.IG.2[["models"]][[2]][["mu.coefficients"]])
A.sigma2.2 <- exp(Area.IG.2[["models"]][[2]][["sigma.coefficients"]])
```

Then, we can create the plot using the hist function.

```
hist(seeds$area, breaks = 50,freq = FALSE, xlab = "Area", main = "Area: mixture
of two Inverse Gaussian Distribution")
lines(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),Area.IG.2[[
"prob"]][1]*dIG(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),
mu = A.mu2.1, sigma = A.sigma2.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),Area.IG.2[[
"prob"]][2]*dIG(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),
mu = A.mu2.2, sigma = A.sigma2.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),
Area.IG.2[["prob"]][1]*dIG(seq(min(seeds$area),max(seeds$area),length=length(see
ds$area)), mu = A.mu2.1, sigma = A.sigma2.1) +
Area.IG.2[["prob"]][2]*dIG(seq(min(seeds$area),max(seeds$area),length=length(see
ds$area)), mu = A.mu2.2, sigma = A.sigma2.2),
          lty = 1, lwd = 3, col = 1)
```

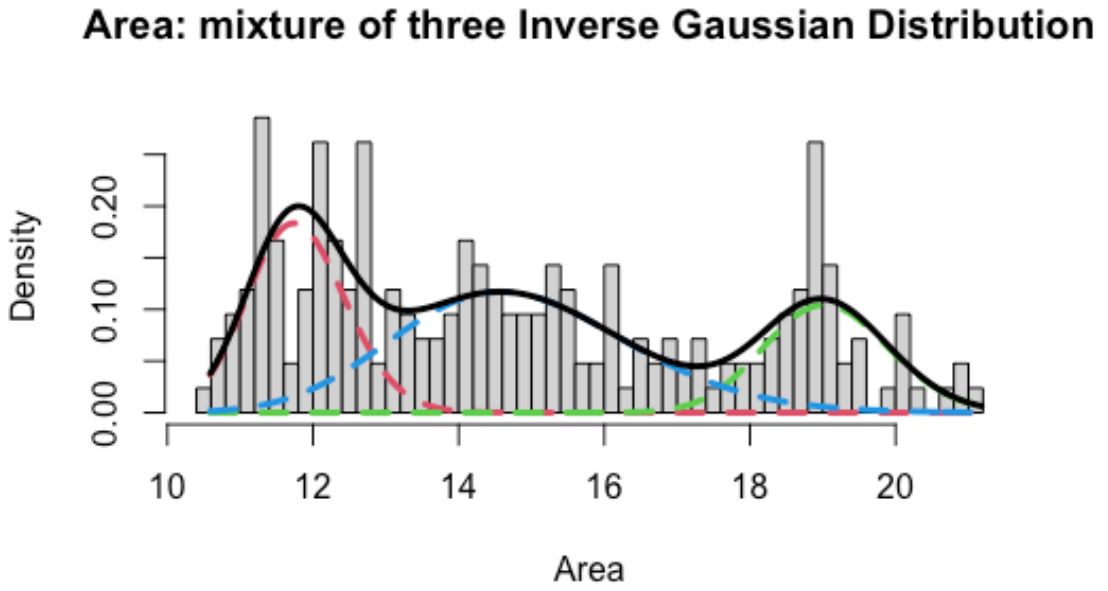



```
> Area.IG.3 <- gamlssMXfits(n = 5, seeds$area~1, family = IG, K = 3, data =
NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> Area.IG.3$aic
[1] 974.7282
> Area.IG.3$sbc
[1] 1001.505
```

The mixture of three Inverse Gaussian provides values of AIC and SBC even better than the mixture of two Inverse Gaussian.

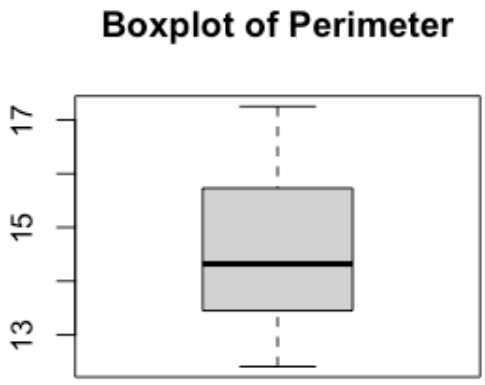
```
A.mu3.1 <- exp(Area.IG.3[["models"]][[1]][["mu.coefficients"]])
A.sigma3.1 <- exp(Area.IG.3[["models"]][[1]][["sigma.coefficients"]])

A.mu3.2 <- exp(Area.IG.3[["models"]][[2]][["mu.coefficients"]])
A.sigma3.2 <- exp(Area.IG.3[["models"]][[2]][["sigma.coefficients"]])

A.mu3.3 <- exp(Area.IG.3[["models"]][[3]][["mu.coefficients"]])
A.sigma3.3 <- exp(Area.IG.3[["models"]][[3]][["sigma.coefficients"]])
```

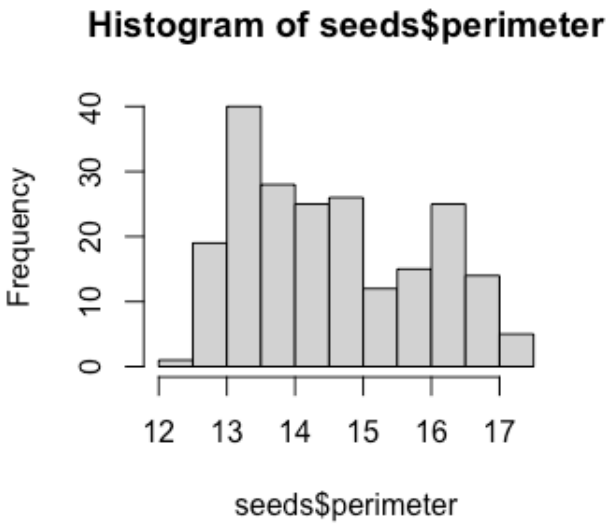
```
hist(seeds$area, breaks = 50,freq = FALSE, xlab = "Area", main = "Area: mixture
of three Inverse Gaussian Distribution")
lines(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),Area.IG.3[[
"prob"]][1]*dIG(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),
mu = A.mu3.1, sigma = A.sigma3.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),Area.IG.3[[
"prob"]][2]*dIG(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),
mu = A.mu3.2, sigma = A.sigma3.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),Area.IG.3[[
"prob"]][3]*dIG(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),
mu = A.mu3.3, sigma = A.sigma3.3),lty=2,lwd=3,col=4)
lines(seq(min(seeds$area),max(seeds$area),length=length(seeds$area)),
Area.IG.3[["prob"]][1]*dIG(seq(min(seeds$area),max(seeds$area),length=length(see
ds$area)), mu = A.mu3.1, sigma = A.sigma3.1) +
Area.IG.3[["prob"]][2]*dIG(seq(min(seeds$area),max(seeds$area),length=length(see
ds$area)), mu = A.mu3.2, sigma = A.sigma3.2) +
Area.IG.3[["prob"]][3]*dIG(seq(min(seeds$area),max(seeds$area),length=length(see
ds$area)), mu = A.mu3.3, sigma = A.sigma3.3) ,
            lty = 1, lwd = 3, col = 1)
```



Univariate Analysis: Perimeter

```
> summary(seeds$perimeter)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.41   13.45   14.32   14.56   15.71   17.25
> sd(seeds$perimeter)
[1] 1.305959
> var(seeds$perimeter)
[1] 1.705528

> boxplot(seeds$perimeter, main = "Boxplot of Perimeter")
> boxplot(seeds$perimeter)$out
numeric(0)
> hist(seeds$perimeter)
```





Exponential

```
> Perimeter.EXP <- histDist(seeds$perimeter, family = EXP, nbins = 30, main =
"Perimeter Exponential Distribution")
> Perimeter.EXP$df.fit
[1] 1
> fitted(Perimeter.EXP, "mu")[1]
[1] 14.55929
> logLik(Perimeter.EXP)
'log Lik.' -772.4281 (df=1)
> AIC(Perimeter.EXP)
[1] 1546.856
> Perimeter.EXP$sbc
[1] 1550.203
```

Gamma

```
> Perimeter.GA <- histDist(seeds$perimeter, family=GA, nbins = 30,
main="Perimeter Gamma distribution")
> Perimeter.GA$df.fit
[1] 2
> fitted(Perimeter.GA, "mu")[1]
[1] 14.55929
> fitted(Perimeter.GA, "sigma")[1]
[1] 0.088723
> logLik(Perimeter.GA)
'log Lik.' -351.1835 (df=2)
> AIC(Perimeter.GA)
[1] 706.367
> Perimeter.GA$sbc
[1] 713.0612
```

Inverse Gaussian

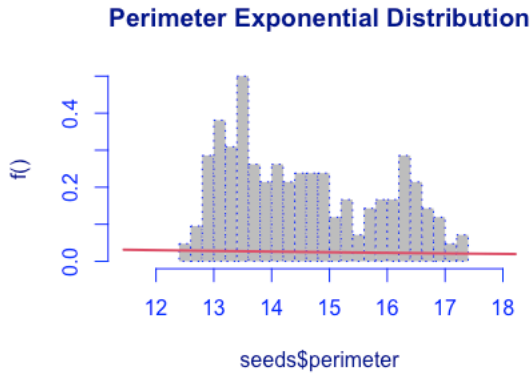
```
> Perimeter.IG <- histDist(seeds$perimeter, family=IG, nbins = 30,
main="Perimeter Inverse Gaussian distribution")
> Perimeter.IG$df.fit
[1] 2
> fitted(Perimeter.IG, "mu")[1]
[1] 14.55929
> fitted(Perimeter.IG, "sigma")[1]
[1] 0.02321589
> logLik(Perimeter.IG)
'log Lik.' -350.165 (df=2)
> AIC(Perimeter.IG)
[1] 704.3299
> Perimeter.IG$sbc
[1] 711.0242
```

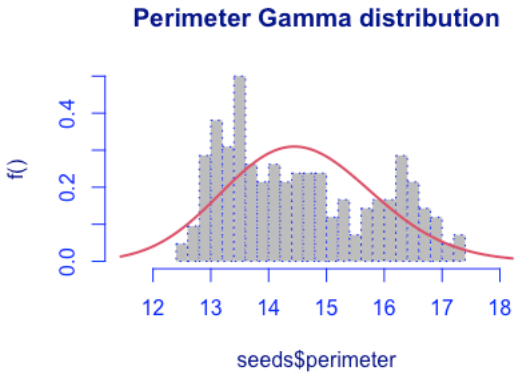
Lognormal

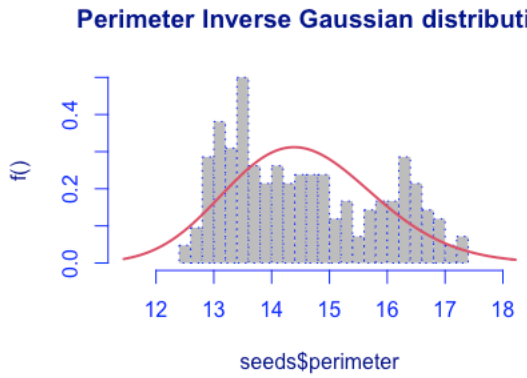
```
> Perimeter.LOGNO <- histDist(seeds$perimeter, family=LOGNO, nbins = 30,
main="Perimeter Log-Normal distribution")
> Perimeter.LOGNO$df.fit
[1] 2
> fitted(Perimeter.LOGNO, "mu")[1]
[1] 2.674289
> fitted(Perimeter.LOGNO, "sigma")[1]
[1] 0.08844561
> logLik(Perimeter.LOGNO)
'log Lik.' -350.2483 (df=2)
> AIC(Perimeter.LOGNO)
[1] 704.4967
> Perimeter.LOGNO$sbc
[1] 711.1909
```

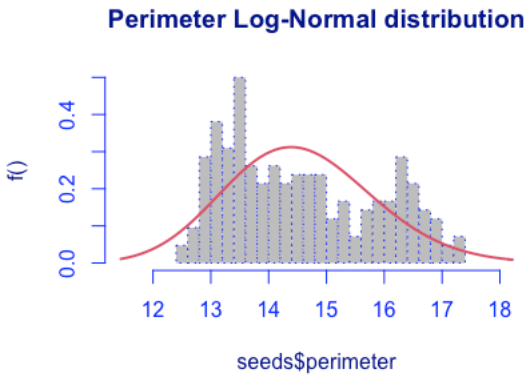
Weibull

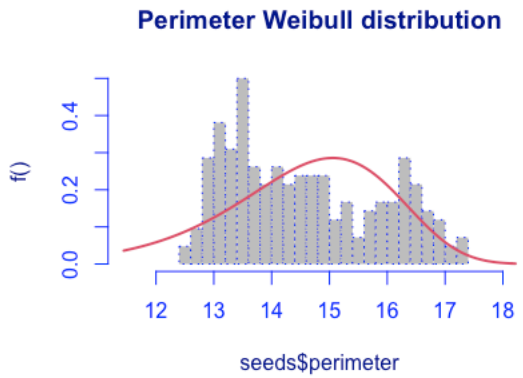
```
> Perimeter.WEI <- histDist(seeds$perimeter, family=WEI, nbins = 30,
main="Perimeter Weibull distribution")
> Perimeter.WEI$df.fit
[1] 2
> fitted(Perimeter.WEI, "mu")[1]
[1] 15.17282
> fitted(Perimeter.WEI, "sigma")[1]
[1] 11.73862
> logLik(Perimeter.WEI)
'log Lik.' -365.8613 (df=2)
> AIC(Perimeter.WEI)
[1] 735.7226
> Perimeter.WEI$sbc
[1] 742.4168
```











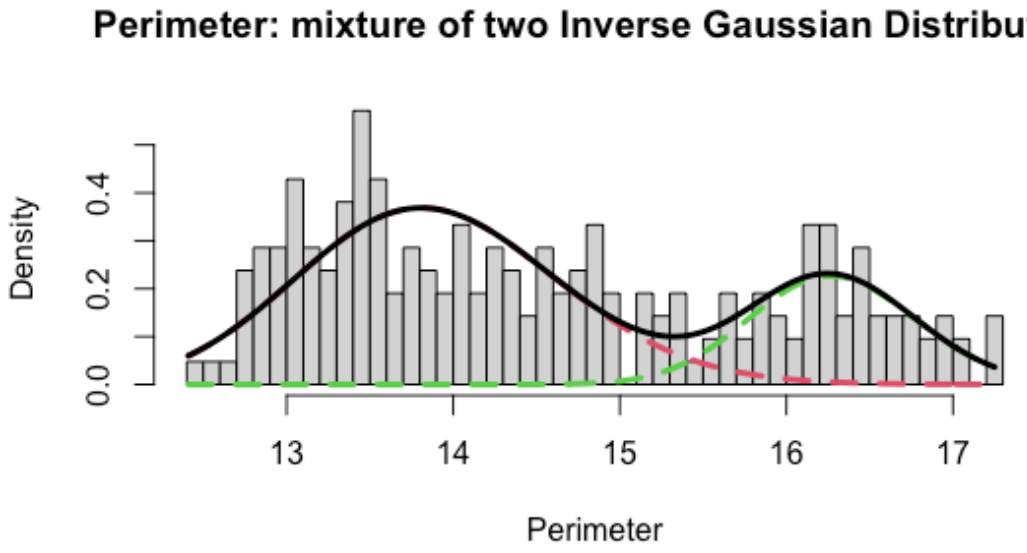

```
> Comparing.Perimeter <- data.frame(row.names = c("Exponential", "Gamma",
"Inverse Gaussian", "Log-Normal", "Weibull"), AIC=c(AIC(Perimeter.EXP),
AIC(Perimeter.GA), AIC(Perimeter.IG), AIC(Perimeter.LOGNO), AIC(Perimeter.WEI)),
SBC=c(Perimeter.EXP$sbc, Perimeter.GA$sbc, Perimeter.IG$sbc,
Perimeter.LOGNO$sbc, Perimeter.WEI$sbc), LL=c(logLik(Perimeter.EXP),
logLik(Perimeter.GA), logLik(Perimeter.IG), logLik(Perimeter.LOGNO),
logLik(Perimeter.WEI)))
> Comparing.Perimeter
                      AIC       SBC        LL
Exponential      1546.8562 1550.2033 -772.4281
Gamma             706.3670  713.0612 -351.1835
Inverse Gaussian  704.3299  711.0242 -350.1650
Log-Normal        704.4967  711.1909 -350.2483
Weibull           735.7226  742.4168 -365.8613
```

We can see that the best values of AIC, SBC and LL are the ones of the Inverse Gaussian Statistical model. So, we try to fit a mixture of two and three Inverse Gaussian to our data.

```
> Perimeter.IG.2 <- gamlssMXfits(n = 5, seeds$perimeter~1, family = IG, K = 2,
data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> Perimeter.IG.2$aic
[1] 660.9432
> Perimeter.IG.2$sbc
[1] 677.6787
```

Both the values are better than the ones found by applying only one Inverse Gaussian to our data.

```
P.mu2.1 <- exp(Perimeter.IG.2[["models"]][[1]][["mu.coefficients"]])
P.sigma2.1 <- exp(Perimeter.IG.2[["models"]][[1]][["sigma.coefficients"]])

P.mu2.2 <- exp(Perimeter.IG.2[["models"]][[2]][["mu.coefficients"]])
P.sigma2.2 <- exp(Perimeter.IG.2[["models"]][[2]][["sigma.coefficients"]])

hist(seeds$perimeter, breaks = 50,freq = FALSE, xlab = "Perimeter", main =
"Perimeter: mixture of two Inverse Gaussian Distribution")
lines(seq(min(seeds$perimeter),max(seeds$perimeter),length=length(seeds$perimete
r)),Perimeter.IG.2[["prob"]][1]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter
),length=length(seeds$perimeter)), mu = P.mu2.1, sigma =
P.sigma2.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$perimeter),max(seeds$perimeter),length=length(seeds$perimete
r)),Perimeter.IG.2[["prob"]][2]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter
),length=length(seeds$perimeter)), mu = P.mu2.2, sigma =
P.sigma2.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$perimeter),max(seeds$perimeter),length=length(seeds$perimete
r)),
Perimeter.IG.2[["prob"]][1]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter),le
ngth=length(seeds$perimeter)), mu = P.mu2.1, sigma = P.sigma2.1) +
Perimeter.IG.2[["prob"]][2]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter),le
ngth=length(seeds$perimeter)), mu = P.mu2.2, sigma = P.sigma2.2),
      lty = 1, lwd = 3, col = 1)
```

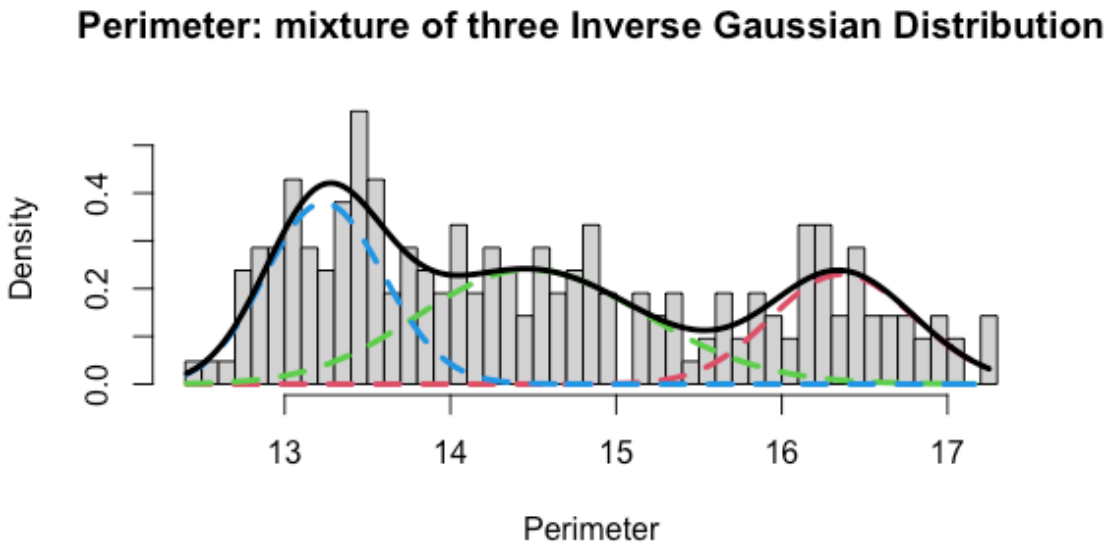


Now, we try to fit a mixture of three Inverse Gaussian.

```
> Perimeter.IG.3 <- gamlssMXfits(n = 5, seeds$perimeter~1, family = IG, K = 3,
data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> Perimeter.IG.3$aic
[1] 647.111
> Perimeter.IG.3$sbc
[1] 673.8879
```

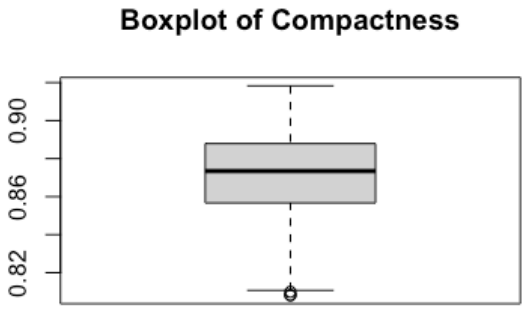
Even in this case, the values of AIC and SBC are better than the values found previously.

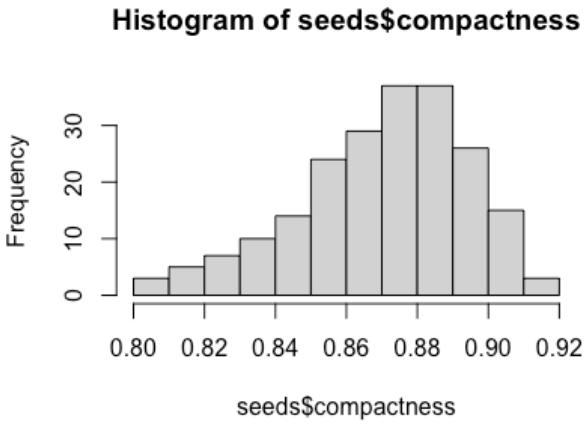
```
P.mu3.1 <- exp(Perimeter.IG.3[["models"]][[1]][["mu.coefficients"]])
P.sigma3.1 <- exp(Perimeter.IG.3[["models"]][[1]][["sigma.coefficients"]])

P.mu3.2 <- exp(Perimeter.IG.3[["models"]][[2]][["mu.coefficients"]])
P.sigma3.2 <- exp(Perimeter.IG.3[["models"]][[2]][["sigma.coefficients"]])

P.mu3.3 <- exp(Perimeter.IG.3[["models"]][[3]][["mu.coefficients"]])
P.sigma3.3 <- exp(Perimeter.IG.3[["models"]][[3]][["sigma.coefficients"]])

hist(seeds$perimeter, breaks = 50,freq = FALSE, xlab = "Perimeter", main =
"Perimeter: mixture of three Inverse Gaussian Distribution")
lines(seq(min(seeds$perimeter),max(seeds$perimeter),length=length(seeds$perimete
r)),Perimeter.IG.3[["prob"]][1]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter
),length=length(seeds$perimeter)), mu = P.mu3.1, sigma =
P.sigma3.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$perimeter),max(seeds$perimeter),length=length(seeds$perimete
r)),Perimeter.IG.3[["prob"]][2]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter
),length=length(seeds$perimeter)), mu = P.mu3.2, sigma =
P.sigma3.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$perimeter),max(seeds$perimeter),length=length(seeds$perimete
r)),Perimeter.IG.3[["prob"]][3]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter
),length=length(seeds$perimeter)), mu = P.mu3.3, sigma =
P.sigma3.3),lty=2,lwd=3,col=4)
lines(seq(min(seeds$perimeter),max(seeds$perimeter),length=length(seeds$perimete
r)),
Perimeter.IG.3[["prob"]][1]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter),le
ngth=length(seeds$perimeter)), mu = P.mu3.1, sigma = P.sigma3.1) +
Perimeter.IG.3[["prob"]][2]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter),le
ngth=length(seeds$perimeter)), mu = P.mu3.2, sigma = P.sigma3.2) +
Perimeter.IG.3[["prob"]][3]*dIG(seq(min(seeds$perimeter),max(seeds$perimeter),le
ngth=length(seeds$perimeter)), mu = P.mu3.3, sigma = P.sigma3.3) ,
            lty = 1, lwd = 3, col = 1)
```



Univariate Analysis: Compactness

```
> summary(seeds$compactness)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8081  0.8569  0.8734  0.8710  0.8878  0.9183
> sd(seeds$compactness)
[1] 0.02362942
> var(seeds$compactness)
[1] 0.0005583493

> boxplot(seeds$compactness, main = "Boxplot of Compactness")
> boxplot(seeds$compactness)$out
[1] 0.8081 0.8082 0.8099
> hist(seeds$compactness)
```





Exponential

```
> Compactness.EXP <- histDist(seeds$compactness, family = EXP, nbins = 30, main
= "Compactness Exponential Distribution")
> Compactness.EXP$df.fit
[1] 1
> fitted(Compactness.EXP, "mu")[1]
[1] 0.8709986
> logLik(Compactness.EXP)
'log Lik.' -180.9959 (df=1)
> AIC(Compactness.EXP)
[1] 363.9917
> Compactness.EXP$sbc
[1] 367.3388
```

Gamma

```
> Compactness.GA <- histDist(seeds$compactness, family=GA, nbins = 30,
main="Compactness Gamma distribution")
> Compactness.GA$df.fit
[1] 2
> fitted(Compactness.GA, "mu")[1]
[1] 0.8709986
> fitted(Compactness.GA, "sigma")[1]
[1] 0.02720725
> logLik(Compactness.GA)
'log Lik.' 487.9759 (df=2)
> AIC(Compactness.GA)
[1] -971.9518
> Compactness.GA$sbc
[1] -965.2576
```

Inverse Gaussian

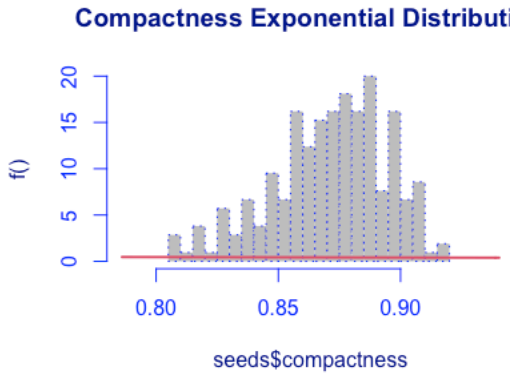
```
> Compactness.IG <- histDist(seeds$compactness, family=IG, nbins = 30,
main="Compactness Inverse Gaussian distribution")
> Compactness.IG$df.fit
[1] 2
> fitted(Compactness.IG, "mu")[1]
[1] 0.8709986
> fitted(Compactness.IG, "sigma")[1]
[1] 0.02923934
> logLik(Compactness.IG)
'log Lik.' 487.4161 (df=2)
> AIC(Compactness.IG)
[1] -970.8323
> Compactness.IG$sbc
[1] -964.138
```

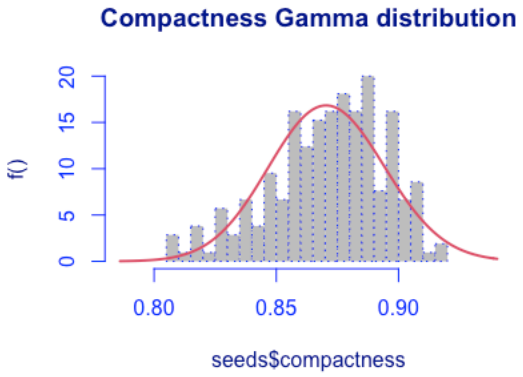
Lognormal

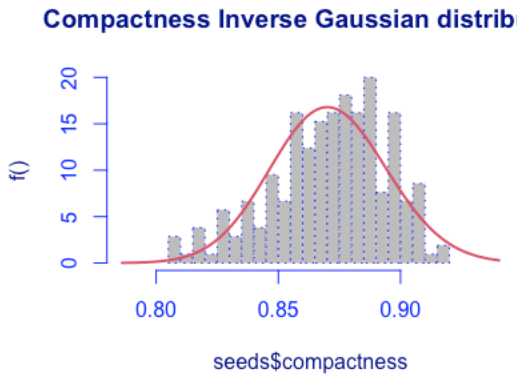
```
> Compactness.LOGNO <- histDist(seeds$compactness, family=LOGNO, nbins = 30,
main="Compactness Log-Normal distribution")
> Compactness.LOGNO$df.fit
[1] 2
> fitted(Compactness.LOGNO, "mu")[1]
[1] -0.1384851
> fitted(Compactness.LOGNO, "sigma")[1]
[1] 0.02728334
> logLik(Compactness.LOGNO)
'log Lik.' 487.4159 (df=2)
> AIC(Compactness.LOGNO)
[1] -970.8317
> Compactness.LOGNO$sbc
[1] -964.1375
```

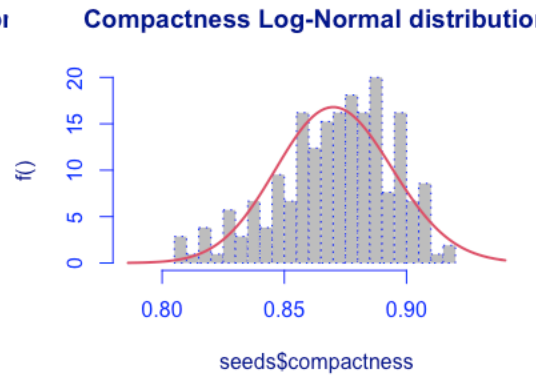
Weibull

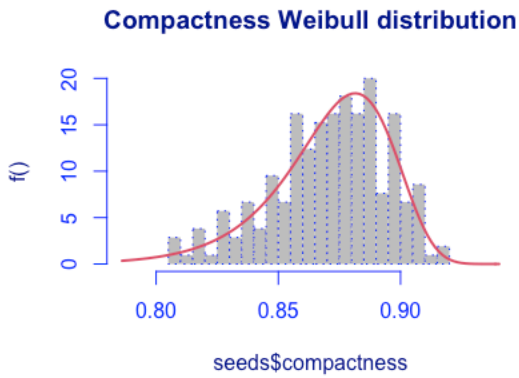
```
> Compactness.WEI <- histDist(seeds$compactness, family=WEI, nbins = 30,
main="Compactness Weibull distribution")
> Compactness.WEI$df.fit
[1] 2
> fitted(Compactness.WEI, "mu")[1]
[1] 0.8819238
> fitted(Compactness.WEI, "sigma")[1]
[1] 44.08887
> logLik(Compactness.WEI)
'log Lik.' 495.3465 (df=2)
> AIC(Compactness.WEI)
[1] -986.693
> Compactness.WEI$sbc
[1] -979.9988
```












```
> Comparing.Compactness <- data.frame(row.names = c("Exponential", "Gamma",
"Inverse Gaussian", "Log-Normal", "Weibull"), AIC=c(AIC(Compactness.EXP),
AIC(Compactness.GA), AIC(Compactness.IG), AIC(Compactness.LOGNO),
AIC(Compactness.WEI)), SBC=c(Compactness.EXP$sbc, Compactness.GA$sbc,
Compactness.IG$sbc, Compactness.LOGNO$sbc, Compactness.WEI$sbc),
LL=c(logLik(Compactness.EXP), logLik(Compactness.GA), logLik(Compactness.IG),
logLik(Compactness.LOGNO), logLik(Compactness.WEI)))
> Comparing.Compactness
                      AIC       SBC        LL
Exponential       363.9917  367.3388 -180.9959
Gamma            -971.9518 -965.2576  487.9759
Inverse Gaussian -970.8323 -964.1380  487.4161
Log-Normal       -970.8317 -964.1375  487.4159
Weibull          -986.6930 -979.9988  495.3465
```

The best values of AIC, SBC and LL are the Weibull values. So, we try to fit the mixtures of two and three Weibull distributions to our data.

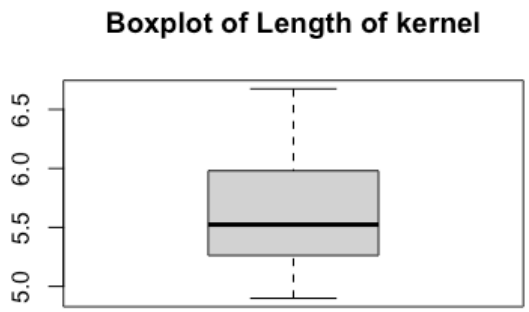
```
> Compactness.WEI.2 <- gamlssMXfits(n = 5, seeds$compactness~1, family = WEI, K
= 2, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> Compactness.WEI.2$aic
[1] -983.2419
> Compactness.WEI.2$sbc
[1] -966.5063
```

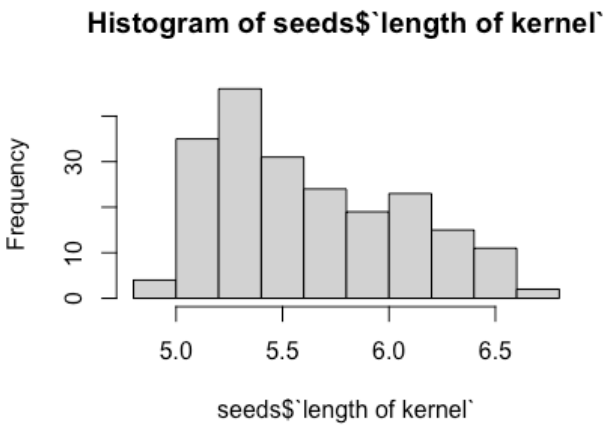
In this case, the values of the mixture of two Inverse Gaussian are worse than using only an Invers Gaussian. Trying with a mixture of three Inverse Gaussian the result is even worse.

```
> Compactness.WEI.3 <- gamlssMXfits(n = 5, seeds$compactness~1, family = WEI, K
= 3, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> Compactness.WEI.3$aic
[1] -977.2469
> Compactness.WEI.3$sbc
[1] -950.47
```

Univariate Analysis: Length of kernel

```
> summary(seeds$`length of kernel`)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.899   5.262   5.524   5.629   5.980   6.675
> sd(seeds$`length of kernel`)
[1] 0.4430635
> var(seeds$`length of kernel`)
[1] 0.1963052

> boxplot(seeds$`length of kernel`, main = "Boxplot of Length of kernel")
> boxplot(seeds$`length of kernel`)$out
numeric(0)
> hist(seeds$`length of kernel`)
```





Exponential

```
> LengthKernel.EXP <- histDist(seeds$`length of kernel`, family = EXP, nbins =
30, main = "Length of kernel Exponential Distribution")
> LengthKernel.EXP$df.fit
[1] 1
> fitted(LengthKernel.EXP, "mu")[1]
[1] 5.628533
> logLik(LengthKernel.EXP)
'log Lik.' -572.8483 (df=1)
> AIC(LengthKernel.EXP)
[1] 1147.697
> LengthKernel.EXP$sbc
[1] 1151.044
```

Gamma

```
> LengthKernel.GA <- histDist(seeds$`length of kernel`, family=GA, nbins = 30,
main="Length of kernel Gamma distribution")
> LengthKernel.GA$df.fit
[1] 2
> fitted(LengthKernel.GA, "mu")[1]
[1] 5.628534
> fitted(LengthKernel.GA, "sigma")[1]
[1] 0.07766445
> logLik(LengthKernel.GA)
'log Lik.' -123.7774 (df=2)
> AIC(LengthKernel.GA)
[1] 251.5547
> LengthKernel.GA$sbc
[1] 258.249
```

Inverse Gaussian

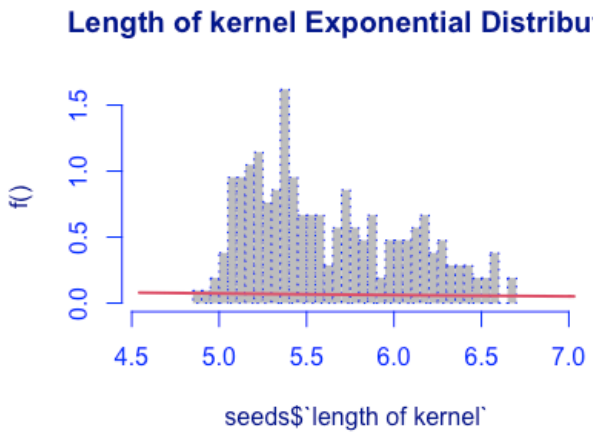
```
> LengthKernel.IG <- histDist(seeds$`length of kernel`, family=IG, nbins = 30,
main="`Length of kernel Inverse Gaussian distribution")
> LengthKernel.IG$df.fit
[1] 2
> fitted(LengthKernel.IG, "mu")[1]
[1] 5.628533
> fitted(LengthKernel.IG, "sigma")[1]
[1] 0.03262756
> logLik(LengthKernel.IG)
'log Lik.' -122.553 (df=2)
> AIC(LengthKernel.IG)
[1] 249.106
> LengthKernel.IG$sbc
[1] 255.8002
```

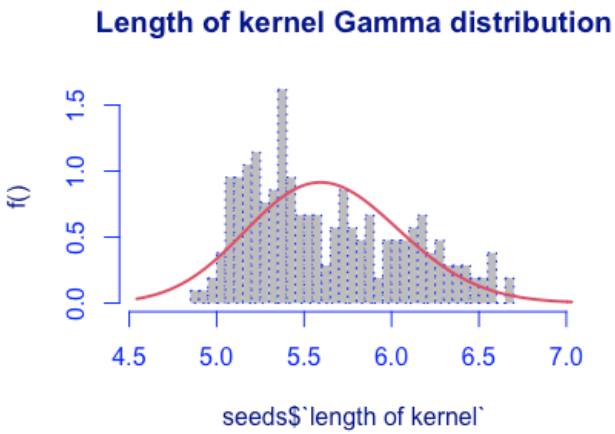
Lognormal

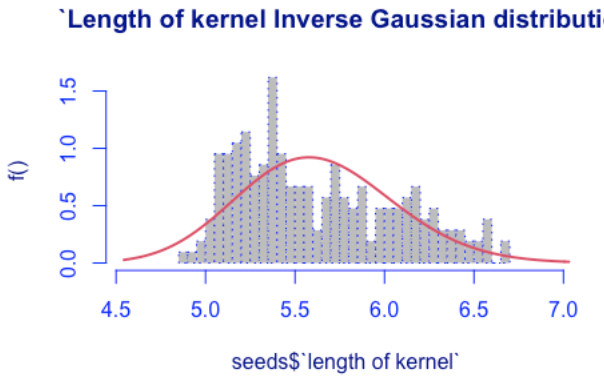
```
> LengthKernel.LOGNO <- histDist(seeds$`length of kernel`, family=LOGNO, nbins =
30, main="Length of kernel Log-Normal distribution")
> LengthKernel.LOGNO$df.fit
[1] 2
> fitted(LengthKernel.LOGNO, "mu")[1]
[1] 1.72483
> fitted(LengthKernel.LOGNO, "sigma")[1]
[1] 0.07731013
> logLik(LengthKernel.LOGNO)
'log Lik.' -122.6039 (df=2)
> AIC(LengthKernel.LOGNO)
[1] 249.2078
> LengthKernel.LOGNO$sbc
[1] 255.902
```

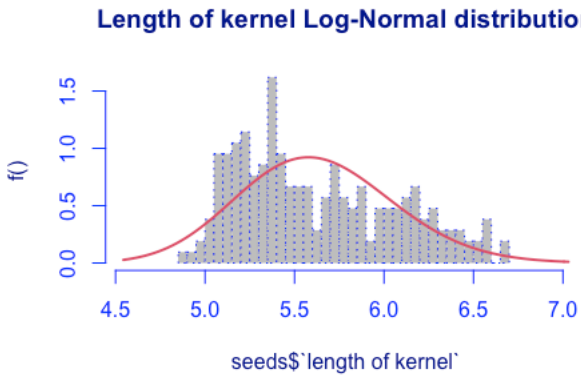
Weibull

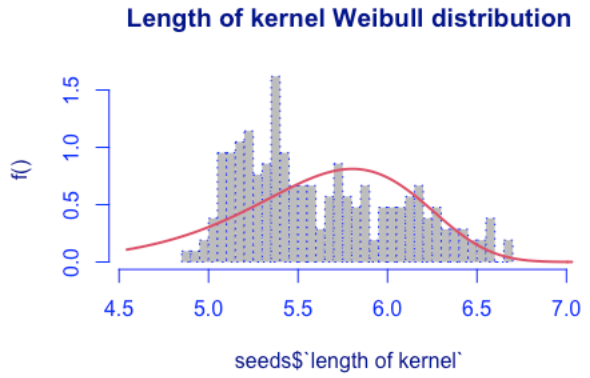
```
> LengthKernel.WEI <- histDist(seeds$`length of kernel`, family=WEI, nbins = 30,
main="Length of kernel Weibull distribution")
> fitted(LengthKernel.WEI, "mu")[1]
[1] 5.840559
> fitted(LengthKernel.WEI, "sigma")[1]
[1] 12.85227
> logLik(LengthKernel.WEI)
'log Lik.' -143.9249 (df=2)
> AIC(LengthKernel.WEI)
[1] 291.8498
> LengthKernel.WEI$sbc
[1] 298.544
```











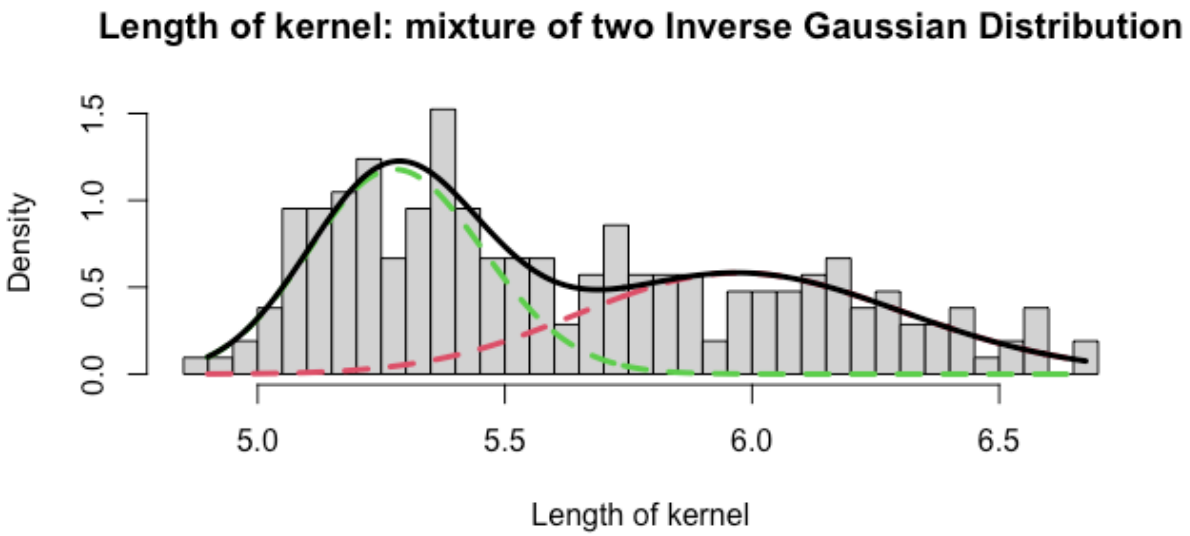

```
> Comparing.LengthKernel <- data.frame(row.names = c("Exponential", "Gamma",
"Inverse Gaussian", "Log-Normal", "Weibull"), AIC=c(AIC(LengthKernel.EXP),
AIC(LengthKernel.GA), AIC(LengthKernel.IG), AIC(LengthKernel.LOGNO),
AIC(LengthKernel.WEI)), SBC=c(LengthKernel.EXP$sbc, LengthKernel.GA$sbc,
LengthKernel.IG$sbc, LengthKernel.LOGNO$sbc, LengthKernel.WEI$sbc),
LL=c(logLik(LengthKernel.EXP), logLik(LengthKernel.GA), logLik(LengthKernel.IG),
logLik(LengthKernel.LOGNO), logLik(LengthKernel.WEI)))
> Comparing.LengthKernel
                      AIC       SBC        LL
Exponential      1147.6965 1151.0436 -572.8483
Gamma             251.5547  258.2490 -123.7774
Inverse Gaussian  249.1060  255.8002 -122.5530
Log-Normal        249.2078  255.9020 -122.6039
Weibull           291.8498  298.5440 -143.9249
```

In this case, we can observe the best values of the AIC, SBC and LL on the Inverse Gaussian Distribution. So, I tried to fit the mixture of two and three Inverse Gaussian Distributions to our data.

```
> LengthKernel.IG.2 <- gamlssMXfits(n = 5, seeds$`length of kernel`~1, family =
IG, K = 2, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> LengthKernel.IG.2$aic
[1] 212.2259
> LengthKernel.IG.2$sbc
[1] 228.9615
```

In this case, the values observed on the mixture of two Inverse Gaussian are better than the ones observed using only one Inverse Gaussian.

```
LK.mu2.1 <- exp(LengthKernel.IG.2[["models"]][[1]][["mu.coefficients"]])
LK.sigma2.1 <- exp(LengthKernel.IG.2[["models"]][[1]][["sigma.coefficients"]])

LK.mu2.2 <- exp(LengthKernel.IG.2[["models"]][[2]][["mu.coefficients"]])
LK.sigma2.2 <- exp(LengthKernel.IG.2[["models"]][[2]][["sigma.coefficients"]])

hist(seeds$`length of kernel`, breaks = 50,freq = FALSE, xlab = "Length of
kernel", main = "Length of kernel: mixture of two Inverse Gaussian
Distribution")
lines(seq(min(seeds$`length of kernel`),max(seeds$`length of
kernel`),length=length(seeds$`length of
kernel`)),LengthKernel.IG.2[["prob"]][1]*dIG(seq(min(seeds$`length of
kernel`),max(seeds$`length of kernel`),length=length(seeds$`length of kernel`)),
mu = LK.mu2.1, sigma = LK.sigma2.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$`length of kernel`),max(seeds$`length of
kernel`),length=length(seeds$`length of
kernel`)),LengthKernel.IG.2[["prob"]][2]*dIG(seq(min(seeds$`length of
kernel`),max(seeds$`length of kernel`),length=length(seeds$`length of kernel`)),
mu = LK.mu2.2, sigma = LK.sigma2.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$`length of kernel`),max(seeds$`length of
kernel`),length=length(seeds$`length of kernel`)),
LengthKernel.IG.2[["prob"]][1]*dIG(seq(min(seeds$`length of
kernel`),max(seeds$`length of kernel`),length=length(seeds$`length of kernel`)),
mu = LK.mu2.1, sigma = LK.sigma2.1) +
LengthKernel.IG.2[["prob"]][2]*dIG(seq(min(seeds$`length of
kernel`),max(seeds$`length of kernel`),length=length(seeds$`length of kernel`)),
mu = LK.mu2.2, sigma = LK.sigma2.2),
      lty = 1, lwd = 3, col = 1)
```

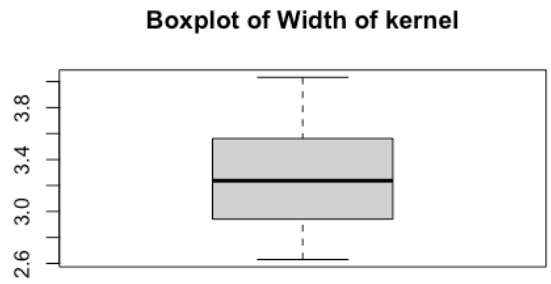


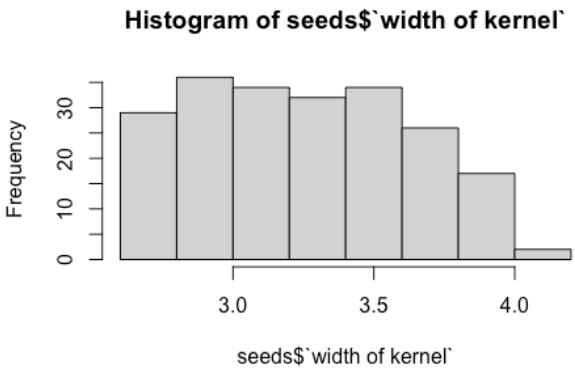

```
> LengthKernel.IG.3 <- gamlssMXfits(n = 5, seeds$`length of kernel`~1, family =
IG, K = 3, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> LengthKernel.IG.3$aic
[1] 214.3505
> LengthKernel.IG.3$sbc
[1] 241.1274
```

In this case, the values of AIC and SBC of the mixture of three Inverse Gaussian are worse than the ones of the mixture of two Inverse Gaussian.

Univariate Analysis: Width of kernel

```
> summary(seeds$`width of kernel`)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.630   2.944   3.237   3.259   3.562   4.033
> sd(seeds$`width of kernel`)
[1] 0.3777144
> var(seeds$`width of kernel`)
[1] 0.1426682

> boxplot(seeds$`width of kernel`, main = "Boxplot of Width of kernel")
> boxplot(seeds$`width of kernel`)$out
numeric(0)
> hist(seeds$`width of kernel`)
```





Exponential

```
> WidthKernel.EXP <- histDist(seeds$`width of kernel`, family = EXP, nbins = 30,
main = "Width of kernel Exponential Distribution")
> WidthKernel.EXP$df.fit
[1] 1
> fitted(WidthKernel.EXP, "mu")[1]
[1] 3.258605
> logLik(WidthKernel.EXP)
'log Lik.' -458.0728 (df=1)
> AIC(WidthKernel.EXP)
[1] 918.1456
> WidthKernel.EXP$sbc
[1] 921.4927
```

Gamma

```
> WidthKernel.GA <- histDist(seeds$`width of kernel`, family=GA, nbins = 30,
main="`Width of kernel Gamma distribution")
> WidthKernel.GA$df.fit
[1] 2
> fitted(WidthKernel.GA, "mu")[1]
[1] 3.258605
> fitted(WidthKernel.GA, "sigma")[1]
[1] 0.115639
> logLik(WidthKernel.GA)
'log Lik.' -92.08132 (df=2)
> AIC(WidthKernel.GA)
[1] 188.1626
> WidthKernel.GA$sbc
[1] 194.8568
```

Inverse Gaussian

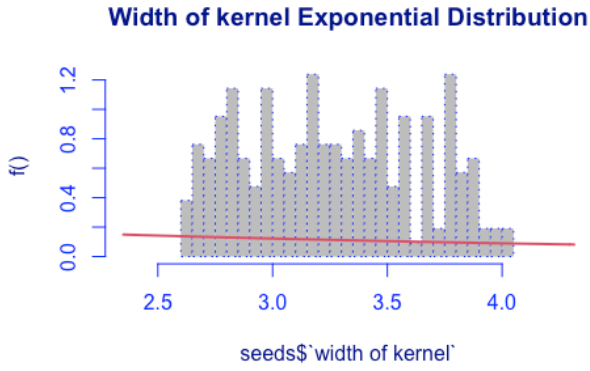
```
> WidthKernel.IG <- histDist(seeds$`width of kernel`, family=IG, nbins = 30,
main="`Width of kernel Inverse Gaussian distribution")
> WidthKernel.IG$df.fit
[1] 2
> fitted(WidthKernel.IG, "mu")[1]
[1] 3.258605
> fitted(WidthKernel.IG, "sigma")[1]
[1] 0.06437053
> logLik(WidthKernel.IG)
'log Lik.' -91.9225 (df=2)
> AIC(WidthKernel.IG)
[1] 187.845
> WidthKernel.IG$sbc
[1] 194.5392
```

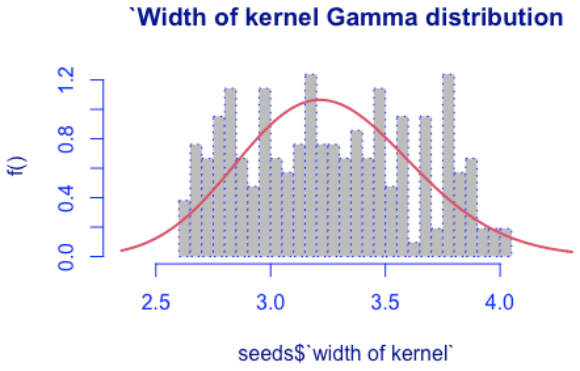
Lognormal

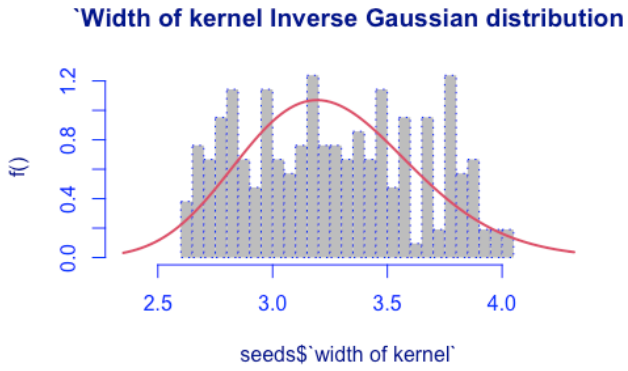
```
> WidthKernel.LOGNO <- histDist(seeds$`width of kernel`, family=LOGNO, nbins =
30, main="Width of kernel Log-Normal distribution")
> WidthKernel.LOGNO$df.fit
[1] 2
> fitted(WidthKernel.LOGNO, "mu")[1]
[1] 1.174598
> fitted(WidthKernel.LOGNO, "sigma")[1]
[1] 0.1158828
> logLik(WidthKernel.LOGNO)
'log Lik.' -92.05367 (df=2)
> AIC(WidthKernel.LOGNO)
[1] 188.1073
> WidthKernel.LOGNO$sbc
[1] 194.8016
```

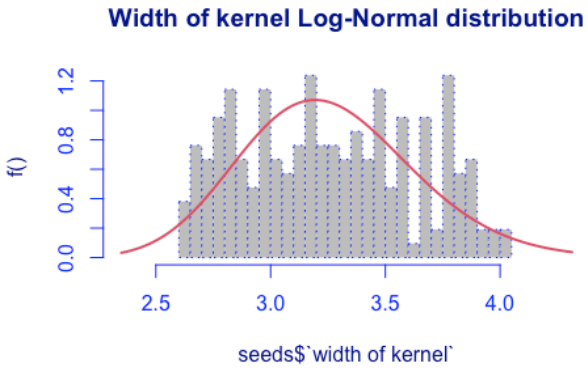
Weibull

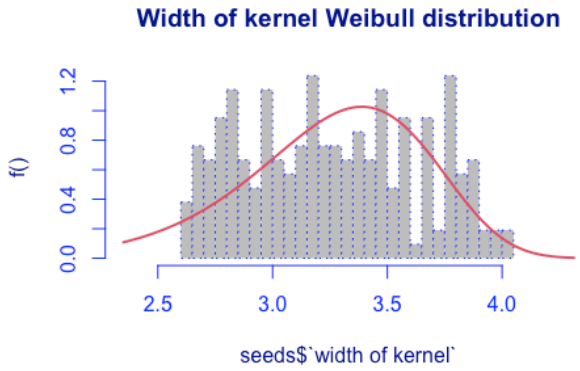
```
> WidthKernel.WEI <- histDist(seeds$`width of kernel`, family=WEI, nbins = 30,
main="Width of kernel Weibull distribution")
> WidthKernel.WEI$df.fit
[1] 2
> fitted(WidthKernel.WEI, "mu")[1]
[1] 3.428685
> fitted(WidthKernel.WEI, "sigma")[1]
[1] 9.514193
> logLik(WidthKernel.WEI)
'log Lik.' -98.62236 (df=2)
> AIC(WidthKernel.WEI)
[1] 201.2447
> WidthKernel.WEI$sbc
[1] 207.9389
```











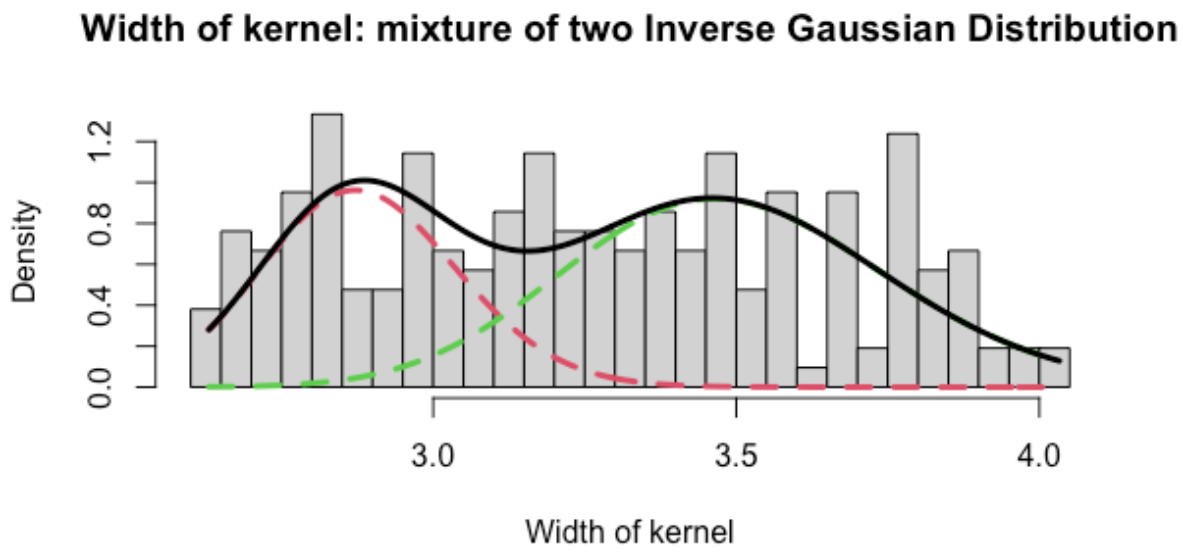

```
> Comparing.WidthKernel <- data.frame(row.names = c("Exponential", "Gamma",
"Inverse Gaussian", "Log-Normal", "Weibull"), AIC=c(AIC(WidthKernel.EXP),
AIC(WidthKernel.GA), AIC(WidthKernel.IG), AIC(WidthKernel.LOGNO),
AIC(WidthKernel.WEI)), SBC=c(WidthKernel.EXP$sbc, WidthKernel.GA$sbc,
WidthKernel.IG$sbc, WidthKernel.LOGNO$sbc, WidthKernel.WEI$sbc),
LL=c(logLik(WidthKernel.EXP), logLik(WidthKernel.GA), logLik(WidthKernel.IG),
logLik(WidthKernel.LOGNO), logLik(WidthKernel.WEI)))
> Comparing.WidthKernel
                     AIC      SBC         LL
Exponential      918.1456 921.4927 -458.07281
Gamma            188.1626 194.8568  -92.08132
Inverse Gaussian 187.8450 194.5392  -91.92250
Log-Normal       188.1073 194.8016  -92.05367
Weibull          201.2447 207.9389  -98.62236
```

The best values of AIC, SBC and LL are the ones of the Inverse Gaussian Distribution.

```
> WidthKernel.IG.2 <- gamlssMXfits(n = 5, seeds$`width of kernel`~1, family =
IG, K = 2, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> WidthKernel.IG.2$aic
[1] 166.7531
> WidthKernel.IG.2$sbc
[1] 183.4886
```

The values of AIC and SBC of the mixture of two Inverse Gaussian are better than the ones observed using only an Inverse Gaussian.

```
WK.mu2.1 <- exp(WidthKernel.IG.2[["models"]][[1]][["mu.coefficients"]])
WK.sigma2.1 <- exp(WidthKernel.IG.2[["models"]][[1]][["sigma.coefficients"]])

WK.mu2.2 <- exp(WidthKernel.IG.2[["models"]][[2]][["mu.coefficients"]])
WK.sigma2.2 <- exp(WidthKernel.IG.2[["models"]][[2]][["sigma.coefficients"]])

hist(seeds$`width of kernel`, breaks = 50,freq = FALSE, xlab = "Width of
kernel", main = "Width of kernel: mixture of two Inverse Gaussian Distribution")
lines(seq(min(seeds$`width of kernel`),max(seeds$`width of
kernel`),length=length(seeds$`width of
kernel`)),WidthKernel.IG.2[["prob"]][1]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu2.1, sigma = WK.sigma2.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$`width of kernel`),max(seeds$`width of
kernel`),length=length(seeds$`width of
kernel`)),WidthKernel.IG.2[["prob"]][2]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu2.2, sigma = WK.sigma2.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$`width of kernel`),max(seeds$`width of
kernel`),length=length(seeds$`width of kernel`)),
WidthKernel.IG.2[["prob"]][1]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu2.1, sigma = WK.sigma2.1) +
WidthKernel.IG.2[["prob"]][2]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu2.2, sigma = WK.sigma2.2),
      lty = 1, lwd = 3, col = 1)
```

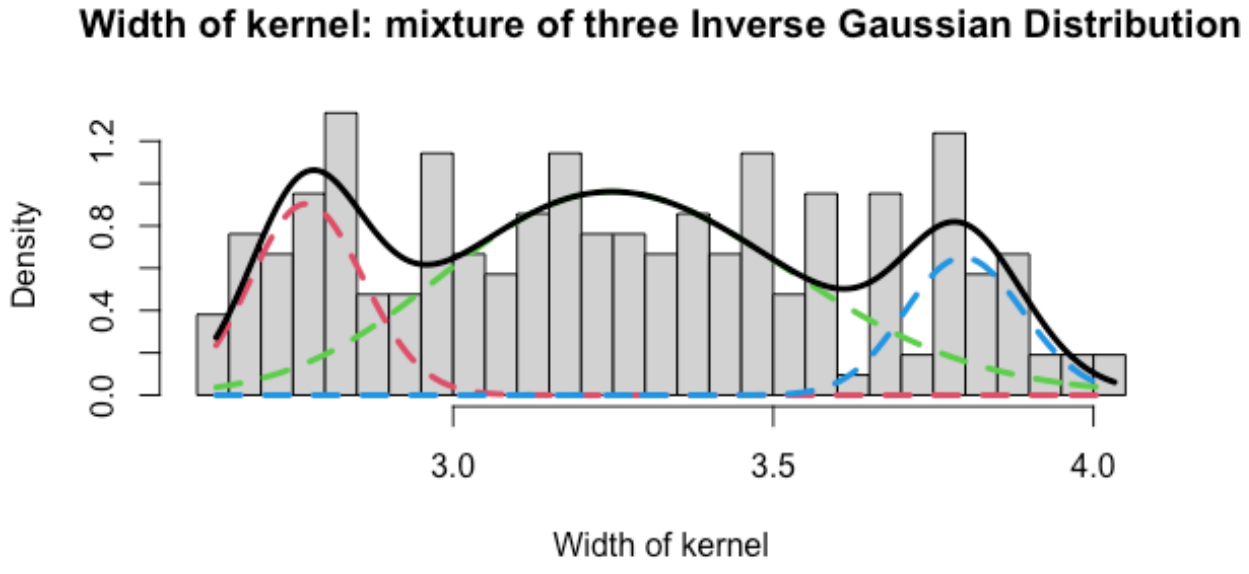



```
> WidthKernel.IG.3 <- gamlssMXfits(n = 5, seeds$`width of kernel`~1, family =
IG, K = 3, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> WidthKernel.IG.3$aic
[1] 154.7135
> WidthKernel.IG.3$sbc
[1] 181.4903
```

The values of AIC and SBC in this case are even better with a mixture of three Inverse Gaussian.

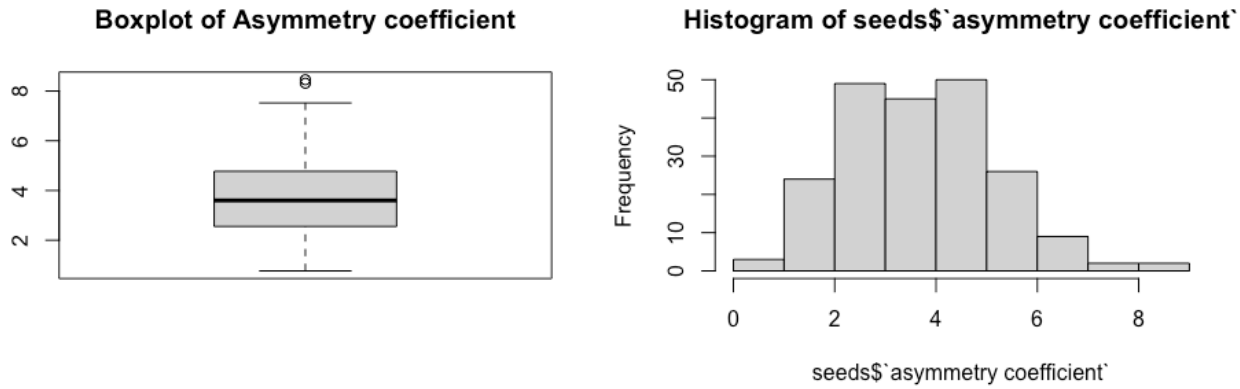
```
WK.mu3.1 <- exp(WidthKernel.IG.3[["models"]][[1]][["mu.coefficients"]])
WK.sigma3.1 <- exp(WidthKernel.IG.3[["models"]][[1]][["sigma.coefficients"]])

WK.mu3.2 <- exp(WidthKernel.IG.3[["models"]][[2]][["mu.coefficients"]])
WK.sigma3.2 <- exp(WidthKernel.IG.3[["models"]][[2]][["sigma.coefficients"]])

WK.mu3.3 <- exp(WidthKernel.IG.3[["models"]][[3]][["mu.coefficients"]])
WK.sigma3.3 <- exp(WidthKernel.IG.3[["models"]][[3]][["sigma.coefficients"]])
```

```
hist(seeds$`width of kernel`, breaks = 50,freq = FALSE, xlab = "Width of
kernel", main = "Width of kernel: mixture of three Inverse Gaussian
Distribution")
lines(seq(min(seeds$`width of kernel`),max(seeds$`width of
kernel`),length=length(seeds$`width of
kernel`)),WidthKernel.IG.3[["prob"]][1]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu3.1, sigma = WK.sigma3.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$`width of kernel`),max(seeds$`width of
kernel`),length=length(seeds$`width of
kernel`)),WidthKernel.IG.3[["prob"]][2]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu3.2, sigma = WK.sigma3.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$`width of kernel`),max(seeds$`width of
kernel`),length=length(seeds$`width of
kernel`)),WidthKernel.IG.3[["prob"]][3]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu3.3, sigma = WK.sigma3.3),lty=2,lwd=3,col=4)
lines(seq(min(seeds$`width of kernel`),max(seeds$`width of
kernel`),length=length(seeds$`width of kernel`)),
WidthKernel.IG.3[["prob"]][1]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu3.1, sigma = WK.sigma3.1) +
WidthKernel.IG.3[["prob"]][2]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu3.2, sigma = WK.sigma3.2) +
WidthKernel.IG.3[["prob"]][3]*dIG(seq(min(seeds$`width of
kernel`),max(seeds$`width of kernel`),length=length(seeds$`width of kernel`)),
mu = WK.mu3.3, sigma = WK.sigma3.3) ,
      lty = 1, lwd = 3, col = 1)
```



Univariate Analysis: Asymmetry coefficient

```
> summary(seeds$`asymmetry coefficient`)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.7651  2.5615  3.5990  3.7002  4.7687  8.4560
> sd(seeds$`asymmetry coefficient`)
[1] 1.503557
> var(seeds$`asymmetry coefficient`)
[1] 2.260684

> boxplot(seeds$`asymmetry coefficient`, main = "Boxplot of Asymmetry
coefficient")
> boxplot(seeds$`asymmetry coefficient`)$out
[1] 8.456 8.315
> hist(seeds$`asymmetry coefficient`)
```



In this case, there are two outliers, but it could not be a problem for our analysis.

Exponential

```
> AsymmetryCoefficient.EXP <- histDist(seeds$`asymmetry coefficient`, family =
EXP, nbins = 30, main = "Asymmetry coefficient Exponential Distribution")
> AsymmetryCoefficient.EXP$df.fit
[1] 1
> fitted(AsymmetryCoefficient.EXP, "mu")[1]
[1] 3.700201
> logLik(AsymmetryCoefficient.EXP)
'log Lik.' -484.7613 (df=1)
> AIC(AsymmetryCoefficient.EXP)
[1] 971.5226
> AsymmetryCoefficient.EXP$sbc
[1] 974.8697
```

Gamma

```
> AsymmetryCoefficient.GA <- histDist(seeds$`asymmetry coefficient`, family=GA,
nbins = 30, main="Asymmetry coefficient Gamma distribution")
> AsymmetryCoefficient.GA$df.fit
[1] 2
> fitted(AsymmetryCoefficient.GA, "mu")[1]
[1] 3.700201
> fitted(AsymmetryCoefficient.GA, "sigma")[1]
[1] 0.4256079
> logLik(AsymmetryCoefficient.GA)
'log Lik.' -380.0827 (df=2)
> AIC(AsymmetryCoefficient.GA)
[1] 764.1653
> AsymmetryCoefficient.GA$sbc
[1] 770.8596
```

Inverse Gaussian

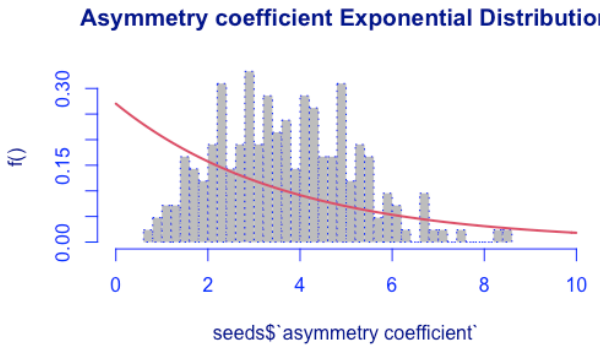
```
> AsymmetryCoefficient.IG <- histDist(seeds$`asymmetry coefficient`, family=IG,
nbins = 30, main="Asymmetry coefficient Inverse Gaussian distribution")
> AsymmetryCoefficient.IG$df.fit
[1] 2
> fitted(AsymmetryCoefficient.IG, "mu")[1]
[1] 3.70022
> fitted(AsymmetryCoefficient.IG, "sigma")[1]
[1] 0.2498304
> logLik(AsymmetryCoefficient.IG)
'log Lik.' -389.4664 (df=2)
> AIC(AsymmetryCoefficient.IG)
[1] 782.9328
> AsymmetryCoefficient.IG$sbc
[1] 789.627
```

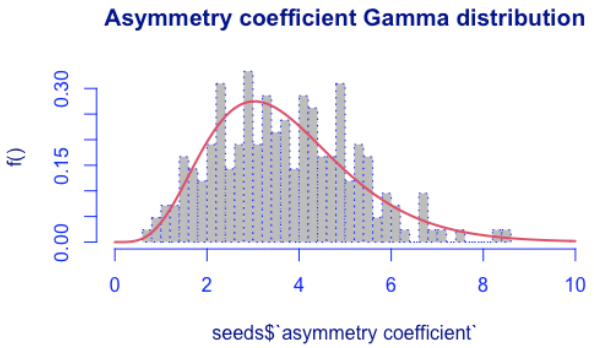
Lognormal

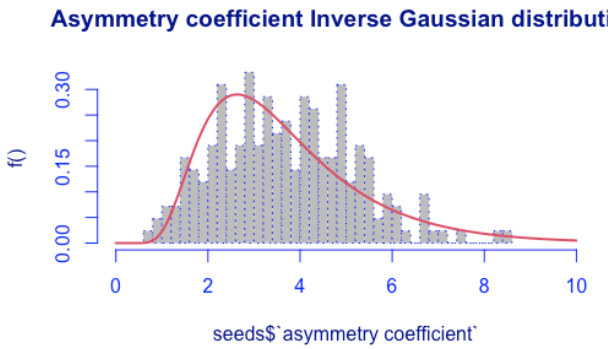
```
> AsymmetryCoefficient.LOGNO <- histDist(seeds$`asymmetry coefficient`,
family=LOGNO, nbins = 30, main="Asymmetry coefficient Log-Normal distribution")
> AsymmetryCoefficient.LOGNO$df.fit
[1] 2
> fitted(AsymmetryCoefficient.LOGNO, "mu")[1]
[1] 1.215092
> fitted(AsymmetryCoefficient.LOGNO, "sigma")[1]
[1] 0.4548616
> logLik(AsymmetryCoefficient.LOGNO)
'log Lik.' -387.7155 (df=2)
> AIC(AsymmetryCoefficient.LOGNO)
[1] 779.431
> AsymmetryCoefficient.LOGNO$sbc
[1] 786.1252
```

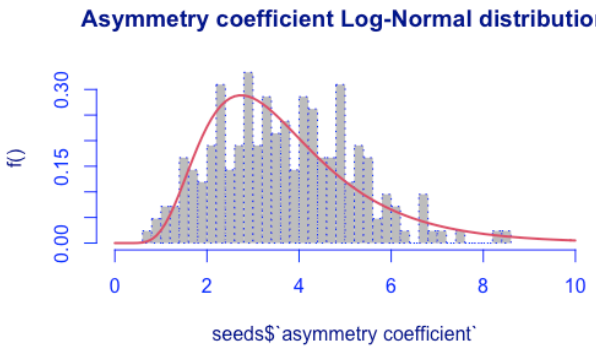
Weibull

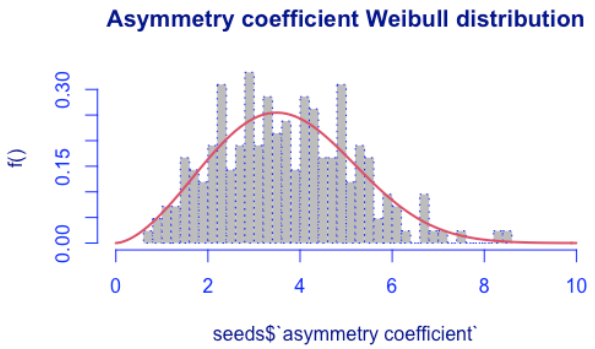
```
> AsymmetryCoefficient.WEI <- histDist(seeds$`asymmetry coefficient`,
family=WEI, nbins = 30, main="Asymmetry coefficient Weibull distribution")
> AsymmetryCoefficient.WEI$df.fit
[1] 2
> fitted(AsymmetryCoefficient.WEI, "mu")[1]
[1] 4.167869
> fitted(AsymmetryCoefficient.WEI, "sigma")[1]
[1] 2.65879
> logLik(AsymmetryCoefficient.WEI)
'log Lik.' -378.3609 (df=2)
> AIC(AsymmetryCoefficient.WEI)
[1] 760.7218
> AsymmetryCoefficient.WEI$sbc
[1] 767.416
```











```
> Comparing.AsymmetryCoefficient <- data.frame(row.names = c("Exponential",
"Gamma", "Inverse Gaussian", "Log-Normal", "Weibull"),
AIC=c(AIC(AsymmetryCoefficient.EXP), AIC(AsymmetryCoefficient.GA),
AIC(AsymmetryCoefficient.IG), AIC(AsymmetryCoefficient.LOGNO),
AIC(AsymmetryCoefficient.WEI)), SBC=c(AsymmetryCoefficient.EXP$sbc,
AsymmetryCoefficient.GA$sbc, AsymmetryCoefficient.IG$sbc,
AsymmetryCoefficient.LOGNO$sbc, AsymmetryCoefficient.WEI$sbc),
LL=c(logLik(AsymmetryCoefficient.EXP), logLik(AsymmetryCoefficient.GA),
logLik(AsymmetryCoefficient.IG), logLik(AsymmetryCoefficient.LOGNO),
logLik(AsymmetryCoefficient.WEI)))
> Comparing.AsymmetryCoefficient
                     AIC      SBC        LL
Exponential      971.5226 974.8697 -484.7613
Gamma            764.1653 770.8596 -380.0827
Inverse Gaussian 782.9328 789.6270 -389.4664
Log-Normal       779.4310 786.1252 -387.7155
Weibull          760.7218 767.4160 -378.3609
```

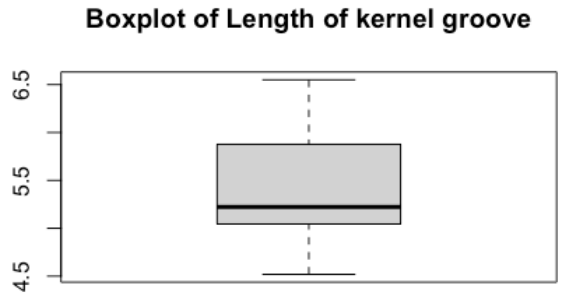
We can observe the best values of AIC, SBC and LL on the Weibull Distribution. So, I tried to fit the mixtures of two and three Weibull Distribution to our data.

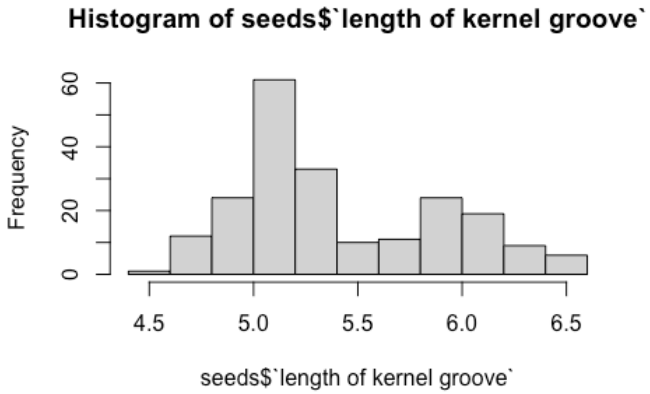
```
> AsymmetryCoefficient.WEI.2 <- gamlssMXfits(n = 5, seeds$`asymmetry
coefficient`~1, family = WEI, K = 2, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> AsymmetryCoefficient.WEI.2$aic
[1] 766.2034
> AsymmetryCoefficient.WEI.2$sbc
[1] 782.9389

> AsymmetryCoefficient.WEI.3 <- gamlssMXfits(n = 5, seeds$`asymmetry
coefficient`~1, family = WEI, K = 3, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> AsymmetryCoefficient.WEI.3$aic
[1] 768.678
> AsymmetryCoefficient.WEI.3$sbc
[1] 795.4548
```

The values observed on the mixtures are worse than the values observed using only a Weibull Distribution.

Univariate Analysis: Length of kernel groove

```
> summary(seeds$`length of kernel groove`)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.519   5.045   5.223   5.408   5.877   6.550
> sd(seeds$`length of kernel groove`)
[1] 0.4914805
> var(seeds$`length of kernel groove`)
[1] 0.2415531

> boxplot(seeds$`length of kernel groove`, main = "Boxplot of Length of kernel
groove")
> boxplot(seeds$`length of kernel groove`)$out
numeric(0)
> hist(seeds$`length of kernel groove`)
```





Exponential

```
> KernelGroove.EXP <- histDist(seeds$`length of kernel groove`, family = EXP,
nbins = 30, main = "Length of kernel groove Exponential Distribution")
> KernelGroove.EXP$df.fit
[1] 1
> fitted(KernelGroove.EXP, "mu")[1]
[1] 5.408071
> logLik(KernelGroove.EXP)
'log Lik.' -564.4574 (df=1)
> AIC(KernelGroove.EXP)
[1] 1130.915
> KernelGroove.EXP$sbc
[1] 1134.262
```

Gamma

```
> KernelGroove.GA <- histDist(seeds$`length of kernel groove`, family=GA, nbins
= 30, main="Length of kernel groove Gamma distribution")
> KernelGroove.GA$df.fit
[1] 2
> fitted(KernelGroove.GA, "mu")[1]
[1] 5.408071
> fitted(KernelGroove.GA, "sigma")[1]
[1] 0.08944192
> logLik(KernelGroove.GA)
'log Lik.' -144.8986 (df=2)
> AIC(KernelGroove.GA)
[1] 293.7972
> KernelGroove.GA$sbc
[1] 300.4914
```

Inverse Gaussian

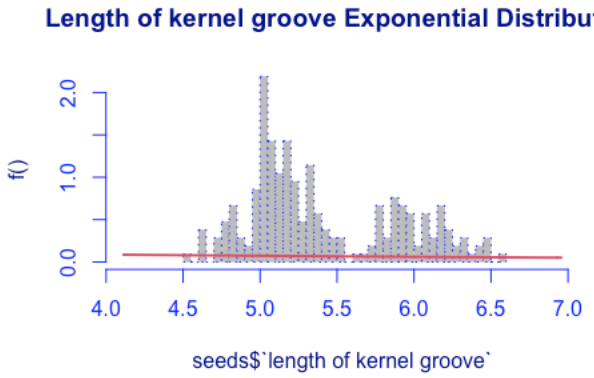
```
> KernelGroove.IG <- histDist(seeds$`length of kernel groove`, family=IG, nbins
= 30, main="Length of kernel groove Inverse Gaussian distribution")
> KernelGroove.IG$df.fit
[1] 2
> fitted(KernelGroove.IG, "mu")[1]
[1] 5.408071
> fitted(KernelGroove.IG, "sigma")[1]
[1] 0.03831204
> logLik(KernelGroove.IG)
'log Lik.' -143.3847 (df=2)
> AIC(KernelGroove.IG)
[1] 290.7693
> KernelGroove.IG$sbc
[1] 297.4636
```

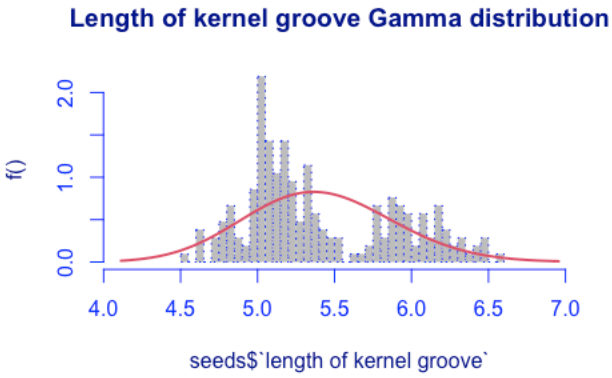
Lognormal

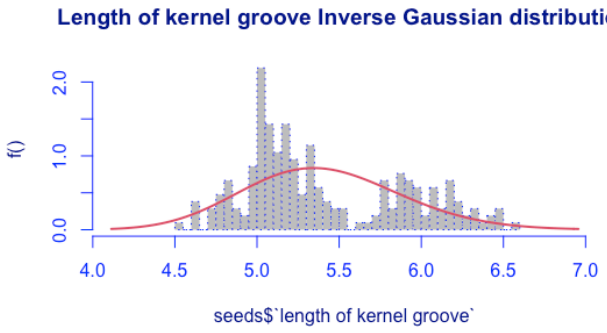
```
> KernelGroove.LOGNO <- histDist(seeds$`length of kernel groove`, family=LOGNO,
nbins = 30, main="Length of kernel groove Log-Normal distribution")
> KernelGroove.LOGNO$df.fit
[1] 2
> fitted(KernelGroove.LOGNO, "mu")[1]
[1] 1.683887
> fitted(KernelGroove.LOGNO, "sigma")[1]
[1] 0.08894813
> logLik(KernelGroove.LOGNO)
'log Lik.' -143.4539 (df=2)
> AIC(KernelGroove.LOGNO)
[1] 290.9079
> KernelGroove.LOGNO$sbc
[1] 297.6021
```

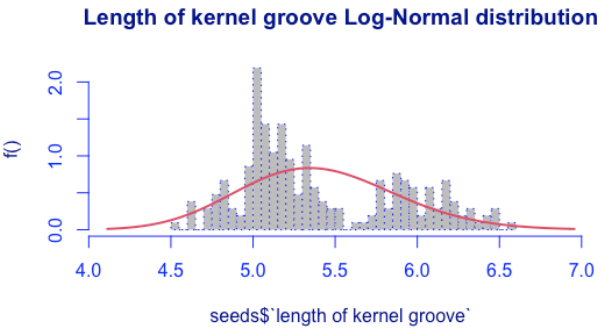
Weibull

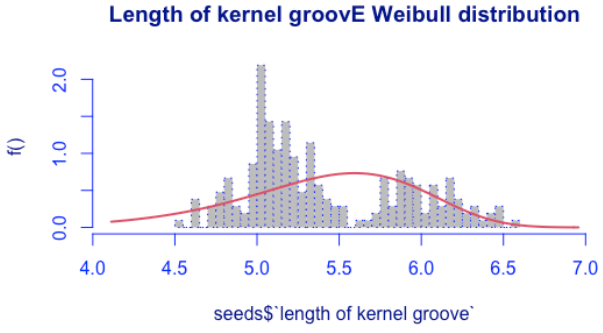
```
> KernelGroove.WEI <- histDist(seeds$`length of kernel groove`, family=WEI,
nbins = 30, main="Length of kernel groove Weibull distribution")
> KernelGroove.WEI$df.fit
[1] 2
> fitted(KernelGroove.WEI, "mu")[1]
[1] 5.640896
> fitted(KernelGroove.WEI, "sigma")[1]
[1] 11.1889
> logLik(KernelGroove.WEI)
'log Lik.' -164.9332 (df=2)
> AIC(KernelGroove.WEI)
[1] 333.8664
> KernelGroove.WEI$sbc
[1] 340.5606
```











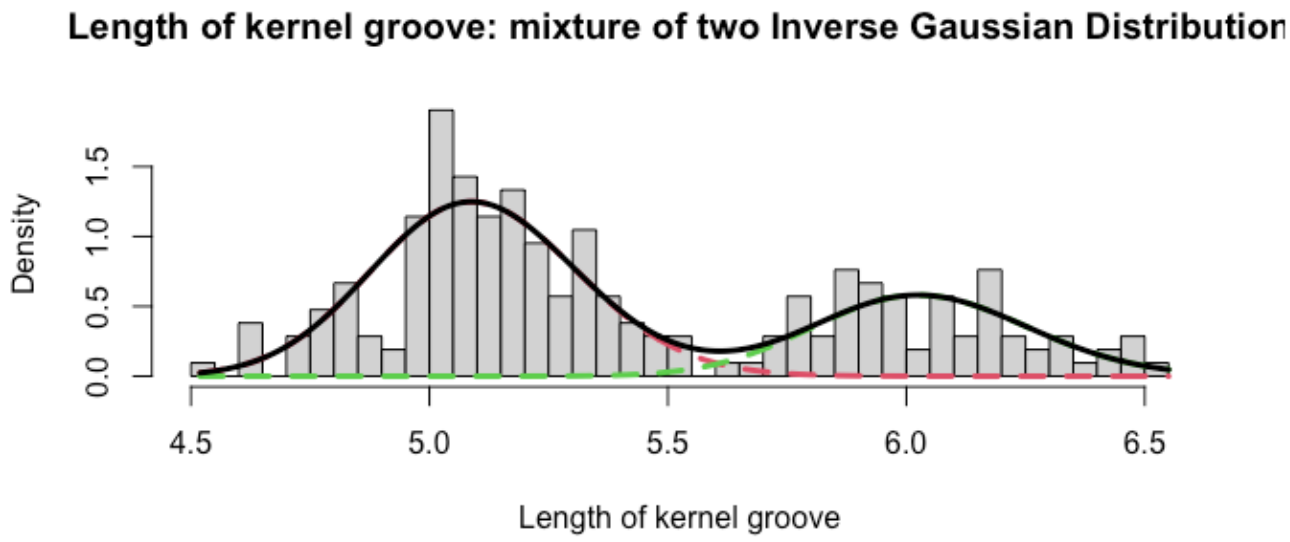
```
> Comparing.KernelGroove <- data.frame(row.names = c("Exponential", "Gamma",
"Inverse Gaussian", "Log-Normal", "Weibull"), AIC=c(AIC(KernelGroove.EXP),
AIC(KernelGroove.GA), AIC(KernelGroove.IG), AIC(KernelGroove.LOGNO),
AIC(KernelGroove.WEI)), SBC=c(KernelGroove.EXP$sbc, KernelGroove.GA$sbc,
KernelGroove.IG$sbc, KernelGroove.LOGNO$sbc, KernelGroove.WEI$sbc),
LL=c(logLik(KernelGroove.EXP), logLik(KernelGroove.GA), logLik(KernelGroove.IG),
logLik(KernelGroove.LOGNO), logLik(KernelGroove.WEI)))
> Comparing.KernelGroove
                     AIC       SBC        LL
Exponential      1130.9149 1134.2620 -564.4574
Gamma             293.7972  300.4914 -144.8986
Inverse Gaussian  290.7693  297.4636 -143.3847
Log-Normal        290.9079  297.6021 -143.4539
Weibull           333.8664  340.5606 -164.9332
```

We can observe the best values of AIC, SBC and LL on the Inverse Gaussian Distribution. So, I tried to fit the mixtures of two and three Inverse Gaussian Distributions to our data.

```
> KernelGroove.IG.2 <- gamlssMXfits(n = 5, seeds$`length of kernel groove`~1,
family = IG, K = 2, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> KernelGroove.IG.2$aic
[1] 217.0728
> KernelGroove.IG.2$sbc
[1] 233.8083
```

The values are significantly better than the previous.

```
KG.mu2.1 <- exp(KernelGroove.IG.2[["models"]][[1]][["mu.coefficients"]])
KG.sigma2.1 <- exp(KernelGroove.IG.2[["models"]][[1]][["sigma.coefficients"]])

KG.mu2.2 <- exp(KernelGroove.IG.2[["models"]][[2]][["mu.coefficients"]])
KG.sigma2.2 <- exp(KernelGroove.IG.2[["models"]][[2]][["sigma.coefficients"]])

hist(seeds$`length of kernel groove`, breaks = 50,freq = FALSE, xlab = "Length
of kernel groove", main = "Length of kernel groove: mixture of two Inverse
Gaussian Distribution")
lines(seq(min(seeds$`length of kernel groove`),max(seeds$`length of kernel
groove`),length=length(seeds$`length of kernel
groove`)),KernelGroove.IG.2[["prob"]][1]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu2.1, sigma = KG.sigma2.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$`length of kernel groove`),max(seeds$`length of kernel
groove`),length=length(seeds$`length of kernel
groove`)),KernelGroove.IG.2[["prob"]][2]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu2.2, sigma = KG.sigma2.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$`length of kernel groove`),max(seeds$`length of kernel
groove`),length=length(seeds$`length of kernel groove`)),
KernelGroove.IG.2[["prob"]][1]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu2.1, sigma = KG.sigma2.1) +
KernelGroove.IG.2[["prob"]][2]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu2.2, sigma = KG.sigma2.2),
          lty = 1, lwd = 3, col = 1)
```

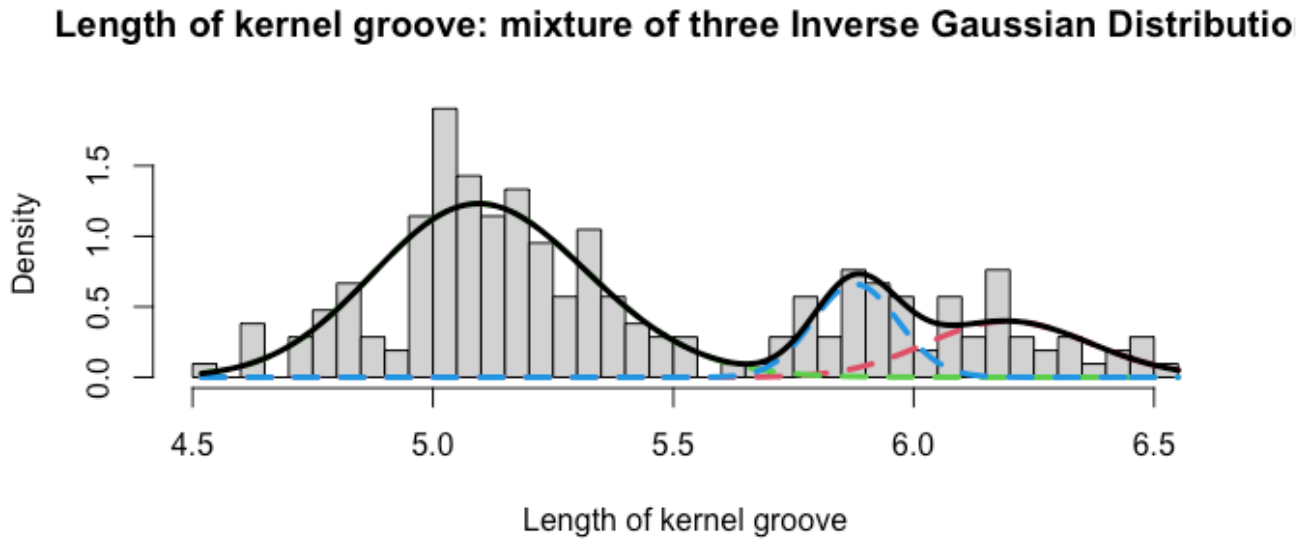



```
> KernelGroove.IG.3 <- gamlssMXfits(n = 5, seeds$`length of kernel groove`~1,
family = IG, K = 3, data = NULL)
model= 1
model= 2
model= 3
model= 4
model= 5
> KernelGroove.IG.3$aic
[1] 219.39
> KernelGroove.IG.3$sbc
[1] 246.1669
```

In this case, the value of AIC is a little bit better than the one observed with the mixture of two Inverse Gaussian, but the value of SBC is worse than the previous. It could be better to use the mixture of two Inverse Gaussian.

```
KG.mu3.1 <- exp(KernelGroove.IG.3[["models"]][[1]][["mu.coefficients"]])
KG.sigma3.1 <- exp(KernelGroove.IG.3[["models"]][[1]][["sigma.coefficients"]])

KG.mu3.2 <- exp(KernelGroove.IG.3[["models"]][[2]][["mu.coefficients"]])
KG.sigma3.2 <- exp(KernelGroove.IG.3[["models"]][[2]][["sigma.coefficients"]])

KG.mu3.3 <- exp(KernelGroove.IG.3[["models"]][[3]][["mu.coefficients"]])
KG.sigma3.3 <- exp(KernelGroove.IG.3[["models"]][[3]][["sigma.coefficients"]])
```

```
hist(seeds$`length of kernel groove`, breaks = 50,freq = FALSE, xlab = "Length
of kernel groove", main = "Length of kernel groove: mixture of three Inverse
Gaussian Distribution")
lines(seq(min(seeds$`length of kernel groove`),max(seeds$`length of kernel
groove`),length=length(seeds$`length of kernel
groove`)),KernelGroove.IG.3[["prob"]][1]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu3.1, sigma = KG.sigma3.1),lty=2,lwd=3,col=2)
lines(seq(min(seeds$`length of kernel groove`),max(seeds$`length of kernel
groove`),length=length(seeds$`length of kernel
groove`)),KernelGroove.IG.3[["prob"]][2]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu3.2, sigma = KG.sigma3.2),lty=2,lwd=3,col=3)
lines(seq(min(seeds$`length of kernel groove`),max(seeds$`length of kernel
groove`),length=length(seeds$`length of kernel
groove`)),KernelGroove.IG.3[["prob"]][3]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu3.3, sigma = KG.sigma3.3),lty=2,lwd=3,col=4)
lines(seq(min(seeds$`length of kernel groove`),max(seeds$`length of kernel
groove`),length=length(seeds$`length of kernel groove`)),
KernelGroove.IG.3[["prob"]][1]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu3.1, sigma = KG.sigma3.1) +
KernelGroove.IG.3[["prob"]][2]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu3.2, sigma = KG.sigma3.2) +
KernelGroove.IG.3[["prob"]][3]*dIG(seq(min(seeds$`length of kernel
groove`),max(seeds$`length of kernel groove`),length=length(seeds$`length of
kernel groove`)), mu = KG.mu3.3, sigma = KG.sigma3.3) ,
      lty = 1, lwd = 3, col = 1)
```



Univariate Analysis: Variety

The last attribute is Variety. As we seen, Variety is a factor with three levels: Kama, Rose and Canadian. This will be useful for the Cluster Analysis.

```
> summary(seeds$variety)
    Kama     Rosa Canadian
      70       70       70
> str(seeds$variety)
 Factor w/ 3 levels "Kama","Rosa",..: 1 1 1 1 1 1 1 1 1 1 ...
```

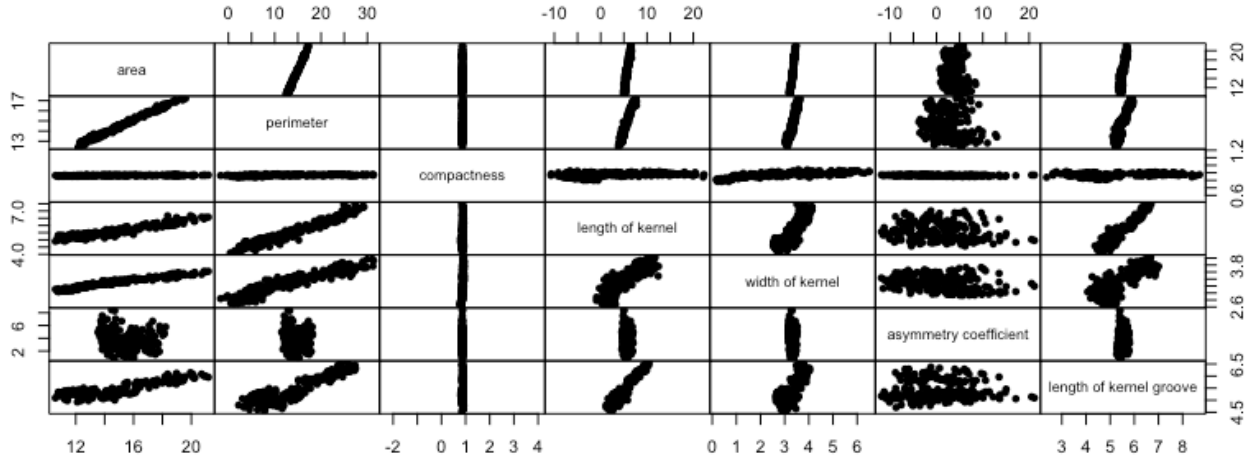
CHAPTER 3: PRINCIPAL COMPONENT ANALYSIS

The PCA is useful in this case because of the high number of variables, that would require a very complex analysis if we only consider the bivariate plot of every couple of variables.
Before proceeding with the PCA, I had to prepare the data, excluding the factor variety.

```
> numerical.seeds <- seeds[,-8]
> head(numerical.seeds)
   area perimeter compactness length of kernel width of kernel
1 15.26     14.84      0.8710            5.763           3.312
2 14.88     14.57      0.8811            5.554           3.333
3 14.29     14.09      0.9050            5.291           3.337
4 13.84     13.94      0.8955            5.324           3.379
5 16.14     14.99      0.9034            5.658           3.562
6 14.38     14.21      0.8951            5.386           3.312
  asymmetry coefficient length of kernel groove
1                 2.221                   5.220
2                 1.018                   4.956
3                 2.699                   4.825
4                 2.259                   4.805
5                 1.355                   5.175
6                 2.462                   4.956
```

Now, I could create the bivariate scatter plots in the original space. Without PCA, this matrix of bivariate scatterplots could be the only way to have information about our dataset, but it could be very difficult due to the big number of plots and to the fact that each plot contains only a small part of the total information contained in the dataset.

```
> pairs(numerical.seeds, gap=0, pch=16, asp=1)
```



It’s useful to evaluate the correlation between variable before to proceed with the Principal Component Analysis. This can be done with the cor function. As can be seen on the next data frame, there is a strong correlation between some variables, like area and perimeter.

```
> cor(numerical.seeds)
                              area  perimeter compactness length of kernel
area                     1.0000000  0.9943409   0.6082884        0.9499854
perimeter                0.9943409  1.0000000   0.5292436        0.9724223
compactness              0.6082884  0.5292436   1.0000000        0.3679151
length of kernel         0.9499854  0.9724223   0.3679151        1.0000000
width of kernel          0.9707706  0.9448294   0.7616345        0.8604149
asymmetry coefficient   -0.2295723 -0.2173404  -0.3314709       -0.1715624
length of kernel groove  0.8636927  0.8907839   0.2268248        0.9328061
                        width of kernel asymmetry coefficient
area                          0.9707706           -0.22957233
perimeter                     0.9448294           -0.21734037
compactness                   0.7616345           -0.33147087
length of kernel              0.8604149           -0.17156243
width of kernel               1.0000000           -0.25803655
asymmetry coefficient        -0.2580365            1.00000000
length of kernel groove       0.7491315           -0.01107902
                        length of kernel groove
area                                 0.86369275
perimeter                            0.89078390
compactness                          0.22682482
length of kernel                     0.93280609
width of kernel                      0.74913147
asymmetry coefficient               -0.01107902
length of kernel groove              1.00000000
```

The next step is to compute the eigen decomposition, in order to obtain the eigen vectors and the eigen values. We first have to obtain the covariance matrix of the scaled data.

```
> scaled_seeds <- scale(numerical.seeds)
> cov_seeds <- cov(scaled_seeds)
> eigen_seeds <- eigen(cov_seeds)
> str(eigen_seeds)
List of 2
 $ values : num [1:7] 5.0312 1.1976 0.678 0.0684 0.0187 ...
 $ vectors: num [1:7, 1:7] -0.444 -0.442 -0.277 -0.424 -0.433 ...
 - attr(*, "class")= chr "eigen"
```

The eigen vectors correspond to the loading of the variables for the PCA, so we can extract them.

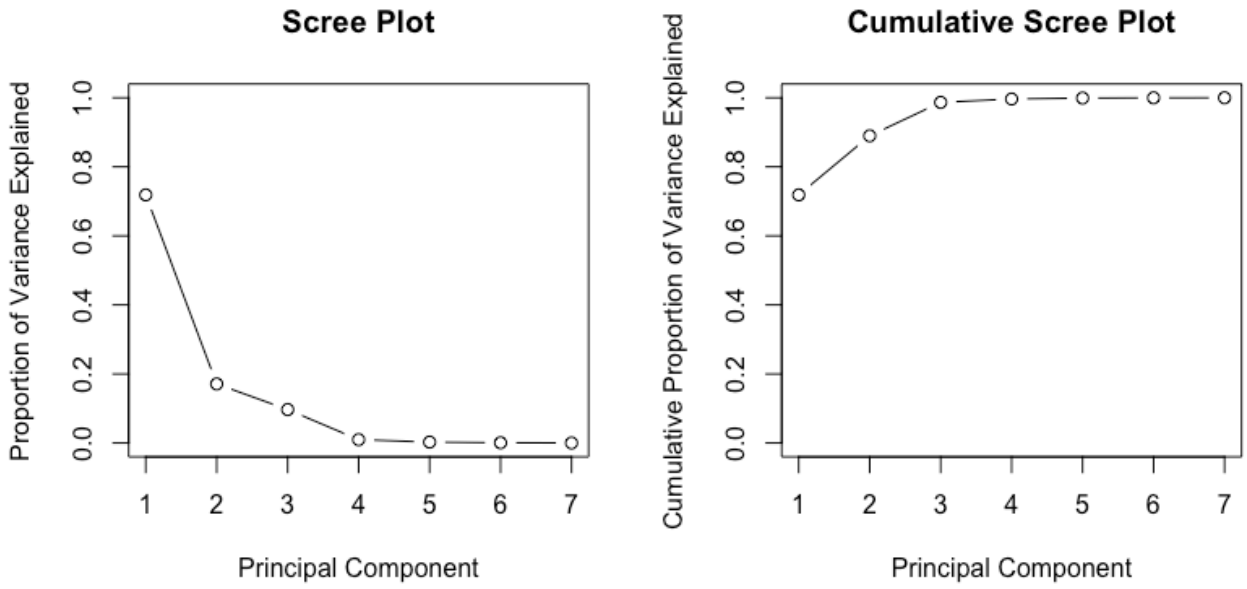
```
> loadings_seeds <- eigen_seeds$vectors
> rownames(loadings_seeds) <- colnames(scaled_seeds)
> colnames(loadings_seeds) <- c("PC1", "PC2", "PC3", "PC4","PC5","PC6","PC7")
> head(loadings_seeds)
                              PC1         PC2         PC3         PC4
area                  -0.4444735  0.02656355 -0.02587094  0.19363997
perimeter             -0.4415715  0.08400282  0.05983912  0.29545659
compactness           -0.2770174 -0.52915125 -0.62969178 -0.33281640
length of kernel      -0.4235633  0.20597518  0.21187966  0.26340659
width of kernel       -0.4328187 -0.11668963 -0.21648338  0.19963039
asymmetry coefficient  0.1186925  0.71688203 -0.67950584  0.09246481
                              PC5         PC6          PC7
area                  -0.20441167 -0.42643686  0.734805689
perimeter             -0.17427591 -0.47623853 -0.670751532
compactness            0.33265481 -0.14162884 -0.072552703
length of kernel       0.76609839  0.27357647  0.046276051
width of kernel       -0.46536555  0.70301171 -0.039289079
asymmetry coefficient  0.03625822 -0.01964186 -0.003723456
```

Now we have to find the number of PCs that explain the right proportion of the variability of the data. We will use three methods: CPVE (Cumulative Proportion of Variance Explained), the scree plot and the Kaiser's rule.

```
> PVE <- eigen_seeds$values/sum(eigen_seeds$values)
> cumsum(PVE)
[1] 0.7187430 0.8898249 0.9866825 0.9964488 0.9991222 0.9998839 1.0000000
```

The CPVE method require that the PVE has to be at least the 80% of the total variability of the data. According to this method, the best choice in respect of information and user-friendliness is 2 PCs.

```
> plot.PVE <- plot(PVE, main = "Scree Plot", xlab = "Principal Component", ylab
= "Proportion of Variance Explained", ylim = c(0,1), type = "b")
> plot.cumsumPVE <- plot(cumsum(PVE),main = "Cumulative Scree Plot",  xlab =
"Principal Component", ylab = "Cumulative Proportion of Variance Explained",
ylim = c(0,1), type = "b")
```



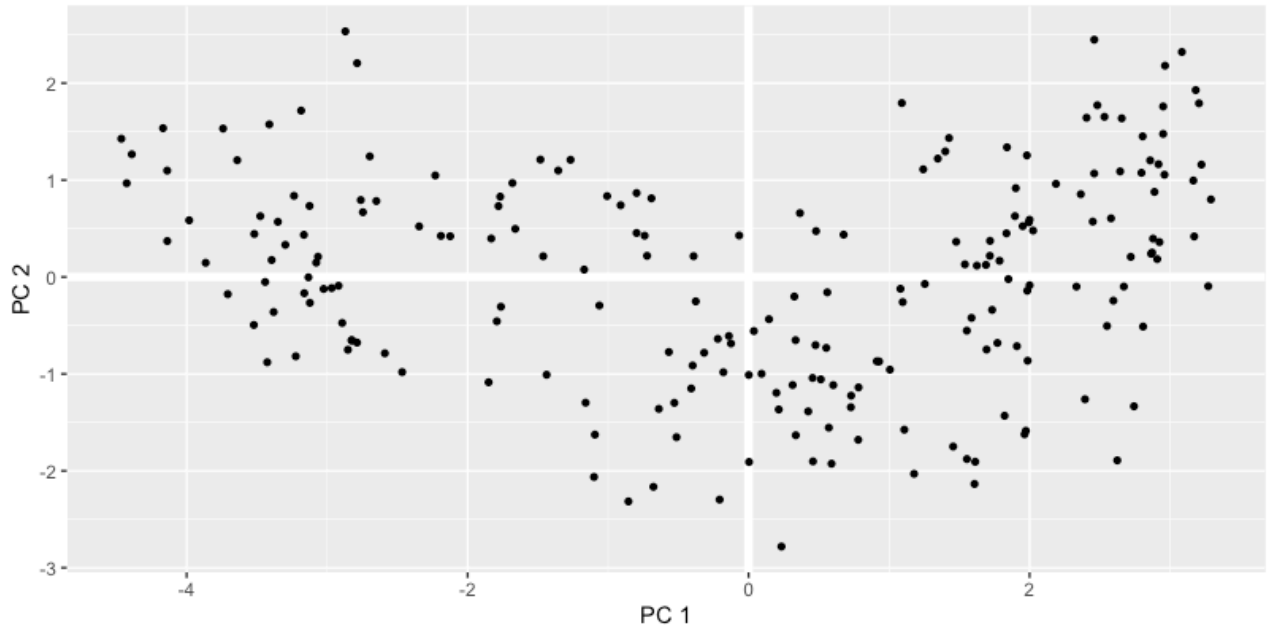
The Scree Plot shows us two main elbows, in correspondence of the first 2 and 3 PCs. In my opinion, it doesn't make sense to use 3 PCs because we would have to add a variable for a little improvement of the CPVE. So, I prefer to consider 2 as a best solution.

```
> eigen_seeds$values[eigen_seeds$values>1]
[1] 5.031201 1.197573
```

The Kaiser's rule requires that the value of the variance explained by the PCs has to be greater than 1, so we have to consider only these values. According to the three methods used, the best solution is to choose the first 2 PCs.

Now I can compute the first two PCs and plot them using the ggplot function.

```
> PC1 <- scaled_seeds %*% loadings_seeds[,1]
> PC2 <- scaled_seeds %*% loadings_seeds[,2]
>
> PC <- data.frame(PC1, PC2)
>
> ggplot(PC, aes(PC1, PC2)) +
+  modelr::geom_ref_line(h = 0) +
+  modelr::geom_ref_line(v = 0) +
+  geom_point(pch=16) +
+  xlab("PC 1") +
+  ylab("PC 2")
```

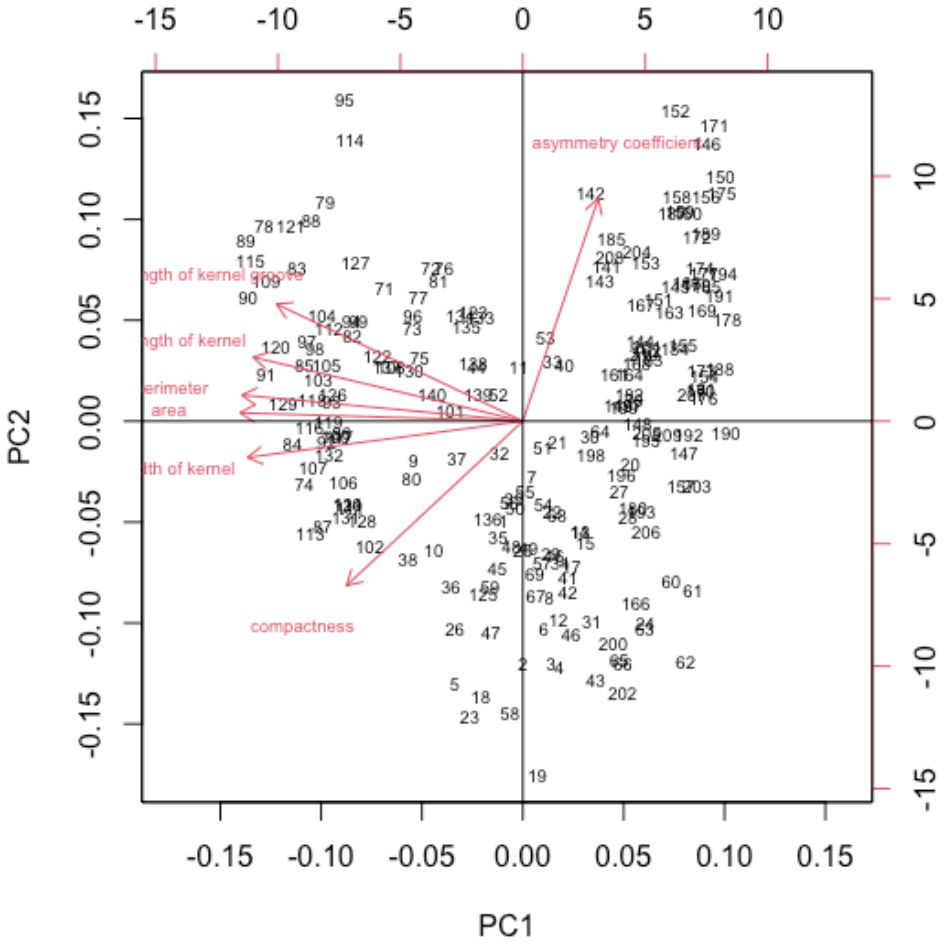


In this plot, every dot represents an observation, so it contains all the values of the variables for this observation. This means that the dots that are near to each other presents similar values. So, the aim of the analysis is to avoid the creation of multiple bivariate plots, enclosing the most of the variability in only one plot. In this case, we captured the essence of data using only two PCs. Then, I have to use the function prcomp to obtain the PCA object of our data because I need it to create the biplot.

```
> PCA.seeds <- prcomp(numerical.seeds, scale = T)
```

The biplot is a plot that merge a usual PCA plot and the loading plot. We can see in the bottom and left axis, respectively, the first and the second PCs. In respect of the previous plot, here we can see how the loadings influence the first two PCs.

```
> biplot(PCA.seeds, cex = 0.6)
> abline(h=0)
> abline(v=0)
```



In this plot, the PCs are displayed as axis, the sample units are represented by dots and the loading of the PCs are represented by arrows. The lengths of the arrows are correlated to the measurement of each observation for all the variables. The angle between two arrows represents the correlation between the variables. An angle close to 0° indicates a positive correlation, while 180° indicates a negative correlation. For example, we can see that compactness and asymmetry coefficient are negative correlated, while perimeter and area are positive correlated. An angle of 90°, like for compactness and length of kernel groove indicates that there is no correlation between the variable. The same argument can be applied for the angle between the arrows and the axis. So, we can see that the variables that mostly influence the first PC are area and perimeter, while the variable that mostly explain the second PC is asymmetry coefficient. We can also know the values of each element for a variable or for the PCs by projecting it on them.

CHAPTER 4: CLUSTER ANALYSIS

The purpose of the Cluster Analysis is to identify pattern of elements with similar characteristics in a dataset. In our case, there is a clear reference that is the variety of the kernel, that could identify some common characteristics.

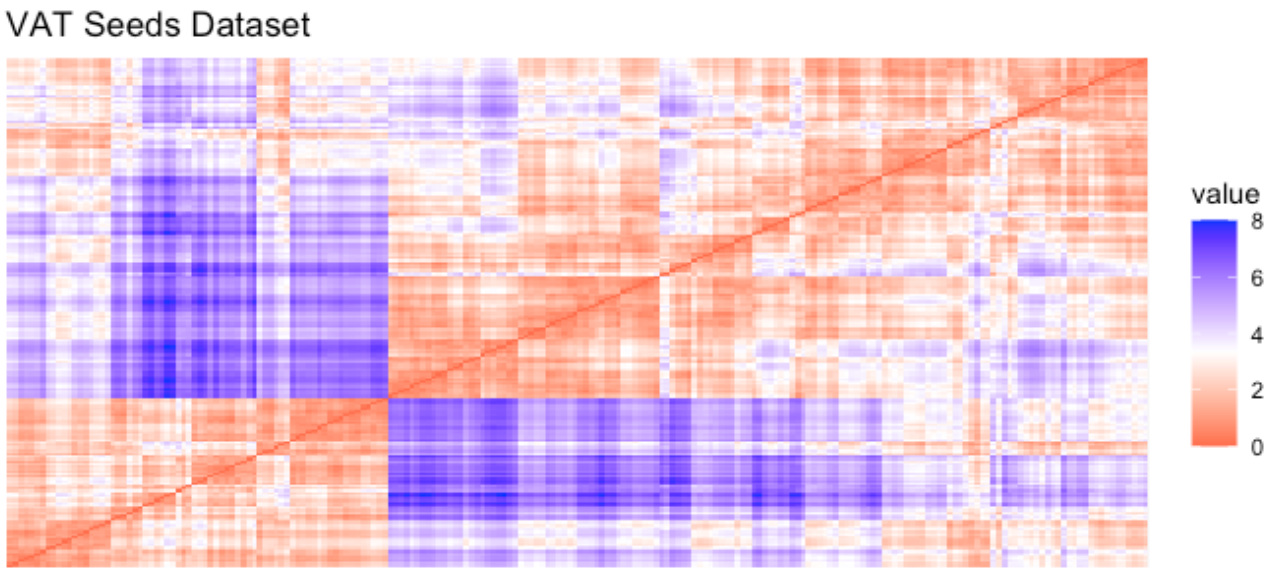
First of all, we have to assess the cluster tendency to evaluate if the clustering could be useful. It will be done using the Hopkins statistic and the VAT (Visual Assessment of cluster Tendency).

The Hopkins statistic is computed using the Hopkins function of the clustertend package.

```
> hopkins(scaled_seeds, n = nrow(scaled_seeds) -1)
$H
[1] 0.19128
```

A value of 0.19128 indicates that there is a good cluster tendency in the Seeds dataset.

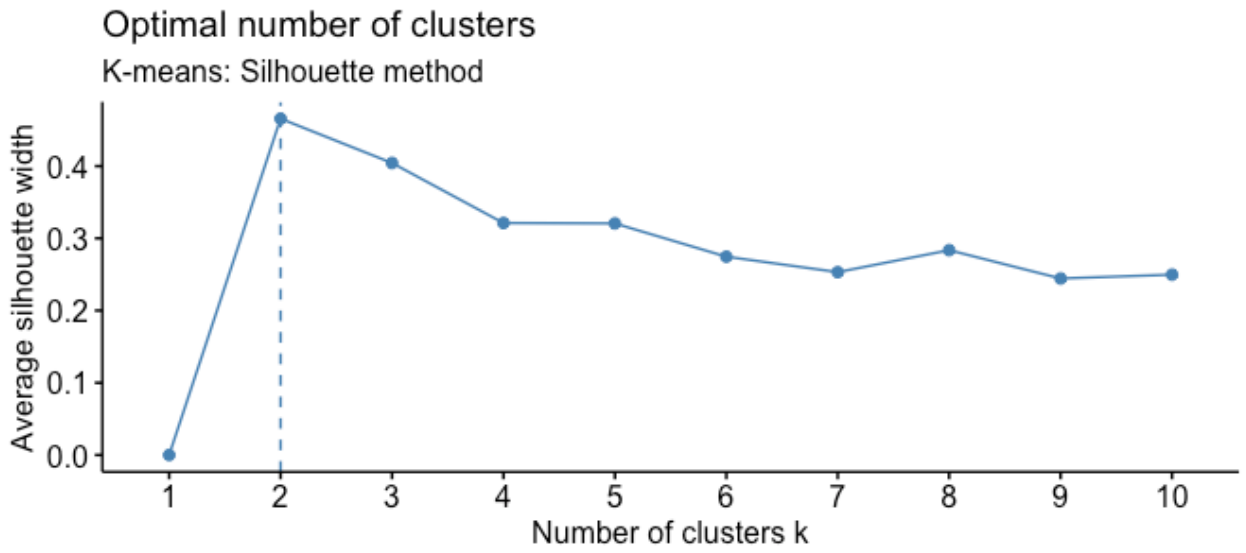
```
> fviz_dist(dist(scaled_seeds), show_labels = FALSE) +
+   labs(title = "VAT Seeds Dataset")
```



This plot has to be interpreted considering that colors that tend to red indicate a high similarity and colors that tend to blue indicate low similarity. In this case, the Vat method shows that it's possible to have different clusters. Then, we can proceed with the different clustering algorithms, deciding first the optimal number of clusters for each algorithm, proceeding with the clustering and then evaluate the best algorithm for our data.

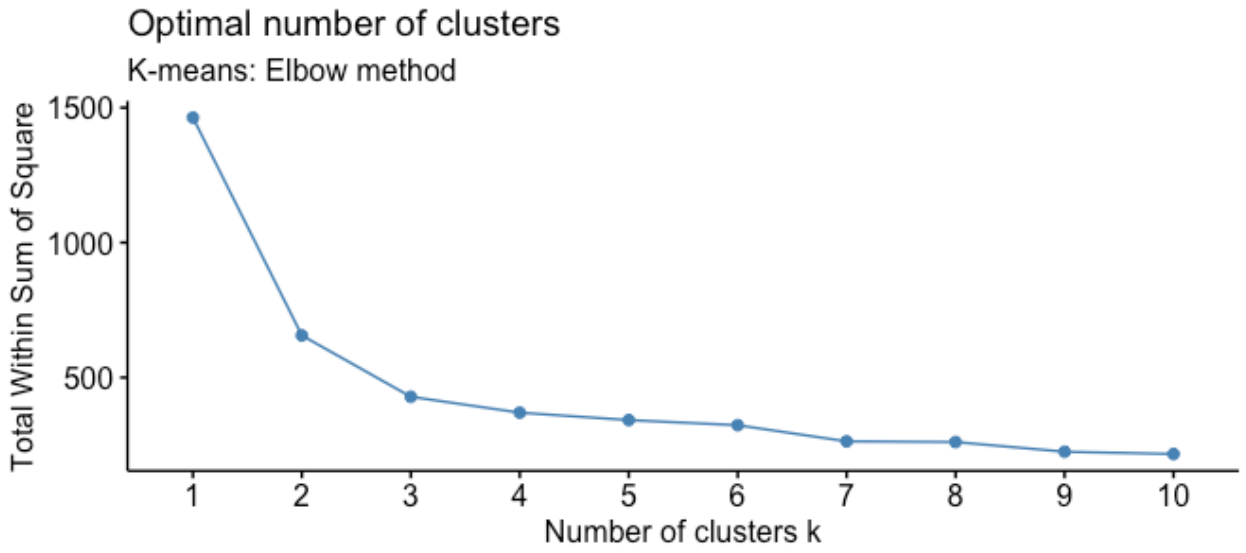
Optimal number of clusters according to K-means algorithm

```
> fviz_nbclust(scaled_seeds, kmeans, method = "silhouette") +
+   labs(subtitle = "K-means: Silhouette method")
```



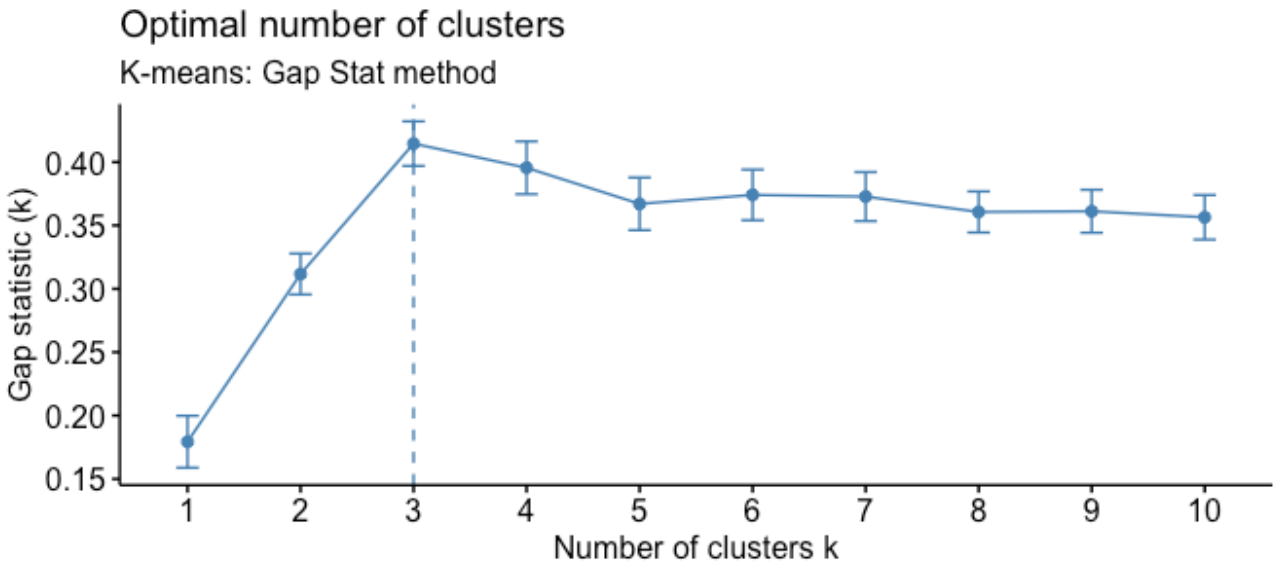
The Silhouette plot show how the average silhouette width change when k increases. The silhouette width is a measure of how an element of the cluster is similar to the other elements in its cluster compared to the elements belonging to the other clusters. It can assume values in the range [-1; +1], when values closer to 1 indicates a better cluster cohesion.

```
> fviz_nbclust(scaled_seeds, kmeans, method = "wss") +
+   labs(subtitle = "K-means: Elbow method")
```



The previous plot it's useful to show how change the WSS (within sum of square) when k increases. In this case, the plot shows two main elbows: the first in correspondence of two clusters and the second in correspondence of three clusters. In my opinion, the second elbow is better because an additional cluster would add only a little improvement to the WSS.

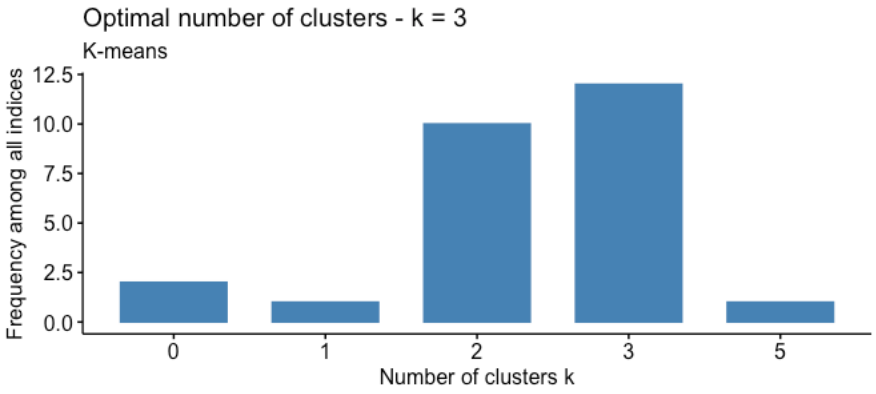
```
> fviz_nbclust(scaled_seeds, kmeans, method = "gap_stat") +
+   labs(subtitle = "K-means: Gap Stat method")
```




```
> nk.km <- NbClust(scaled_seeds, diss = NULL, distance = "euclidean", min.nc =
2, max.nc = 5, method = "kmeans")
*** : The Hubert index is a graphical method of determining the number of
clusters.
In the plot of Hubert index, we seek a significant knee that corresponds to a
significant increase of the value of the measure i.e the significant peak in
Hubert index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
In the plot of D index, we seek a significant knee (the significant peak in
Dindex second differences plot) that corresponds to a significant increase of
the value of the measure.

*******************************************************************
* Among all indices:
* 10 proposed 2 as the best number of clusters
* 12 proposed 3 as the best number of clusters
* 1 proposed 5 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3


*******************************************************************
```

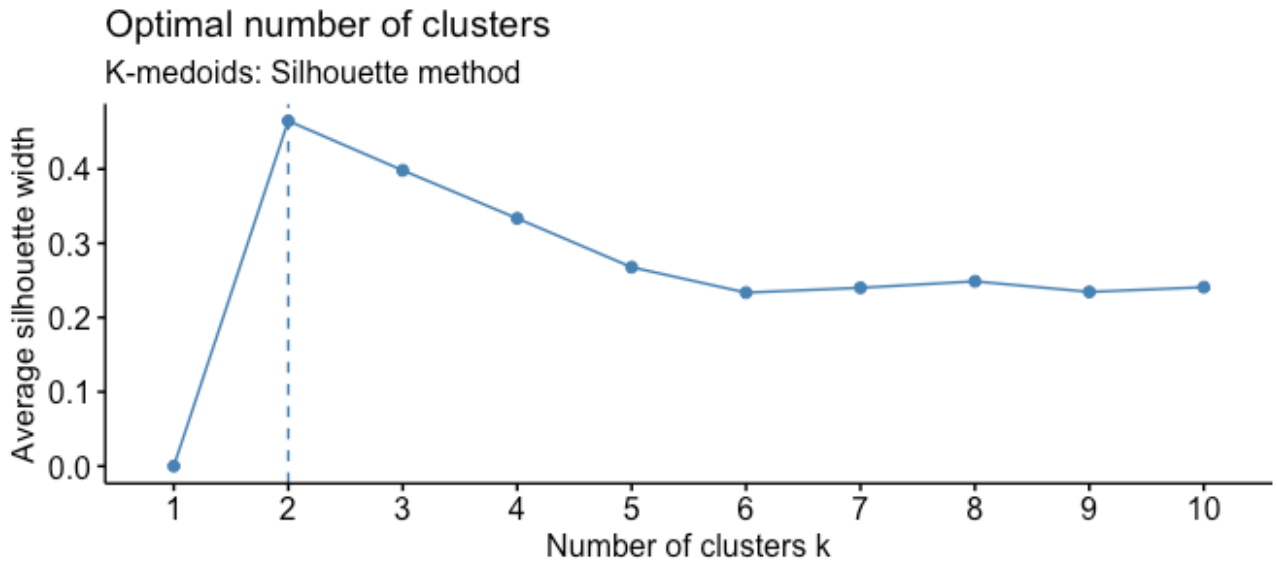
According to all the method used, except the Silhouette one, the best number of K-means clusters is 3.

```
> fviz_nbclust(nk.km) +
+   labs(subtitle = "K-means")
```

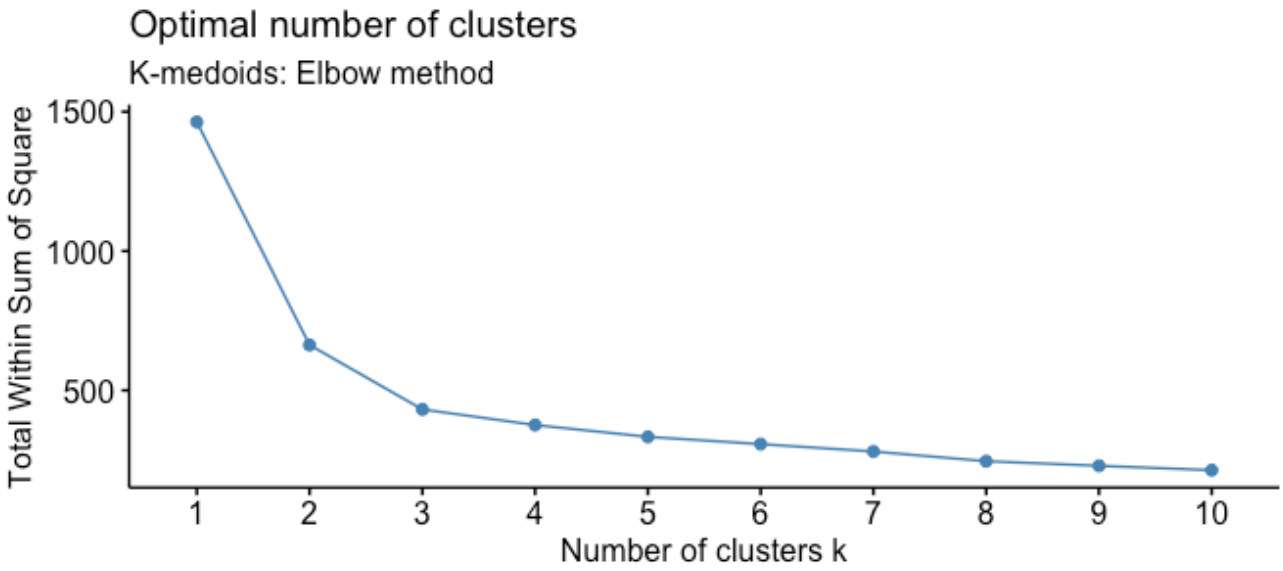


Optimal number of clusters according to K-medoids algorithm

```
> fviz_nbclust(scaled_seeds, pam, method = "silhouette") +
+   labs(subtitle = "K-medoids: Silhouette method")
```

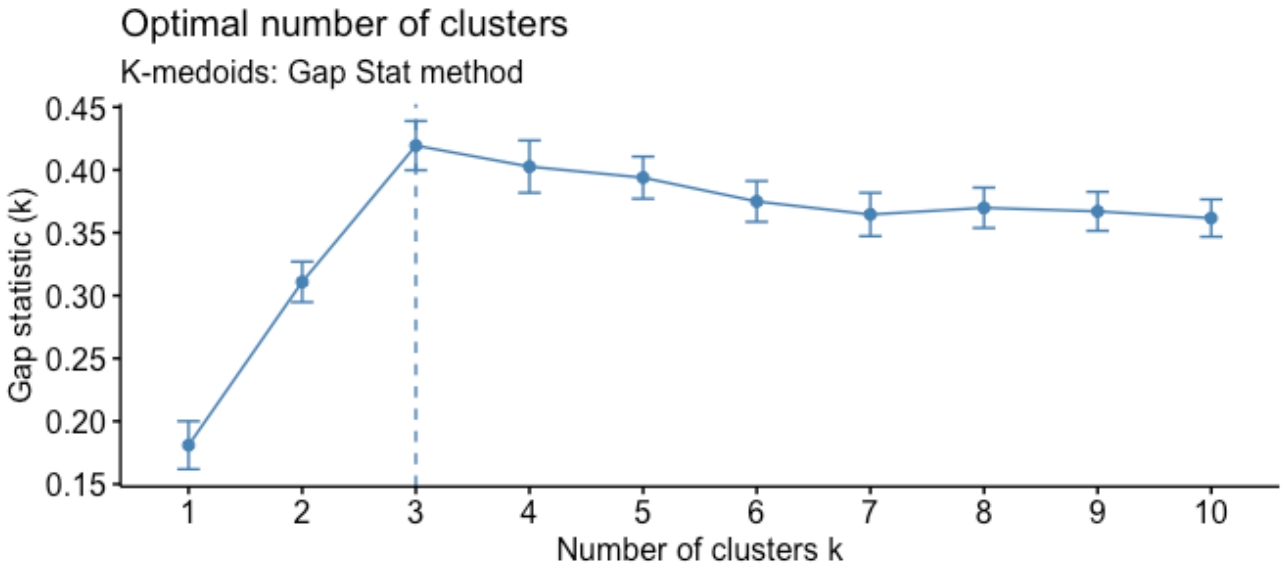


```
> fviz_nbclust(scaled_seeds, pam, method = "wss") +
+   labs(subtitle = "K-medoids: Elbow method")
```



The previous plot presents two main elbows, like the one of K-means. The interpretation is the same.

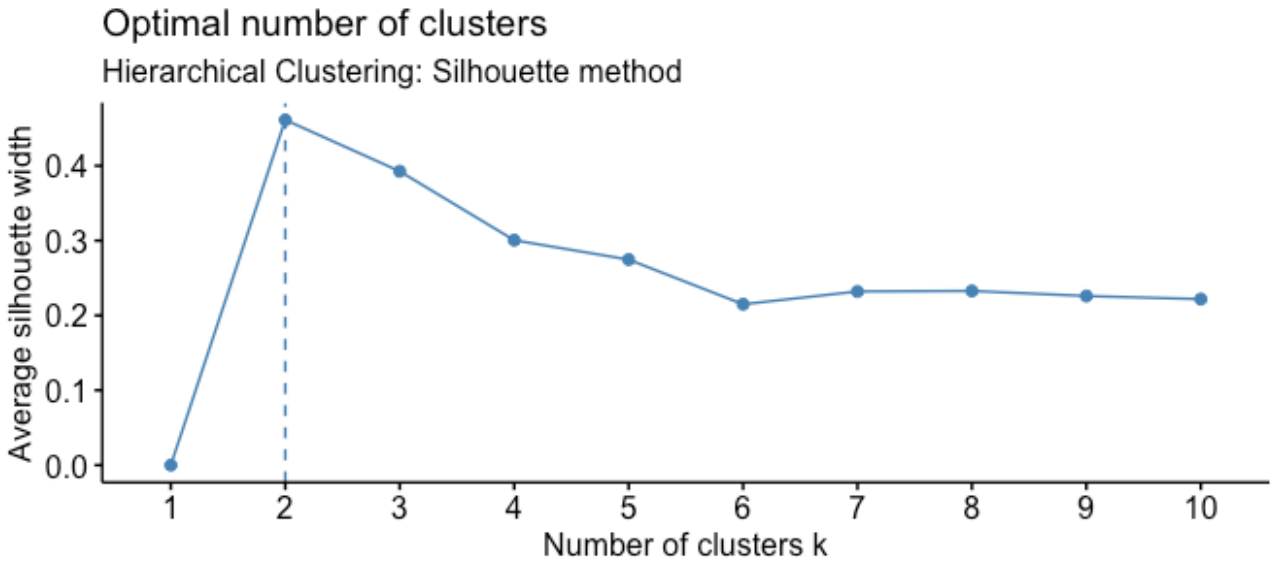
```
> fviz_nbclust(scaled_seeds, pam, method = "gap_stat") +
+   labs(subtitle = "K-medoids: Gap Stat method")
```



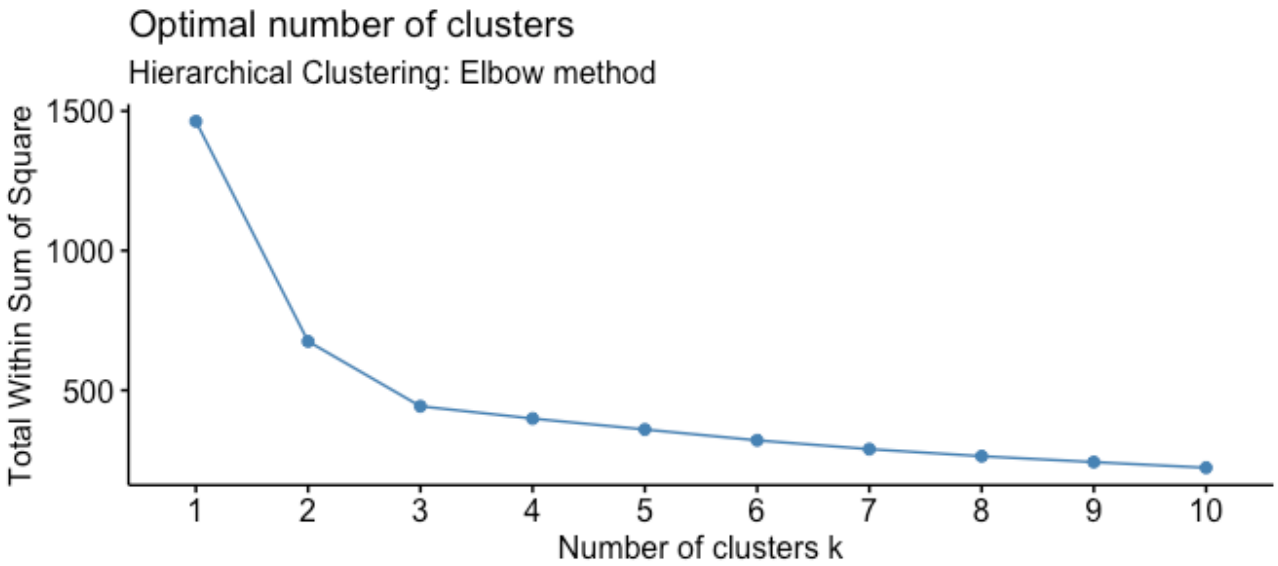
According to all the method used, except the Silhouette one, the best number of K-medoids clusters is 3.

Optimal number of clusters according to Hierarchical Clustering algorithm

```
> fviz_nbclust(scaled_seeds, hcut, method = "silhouette") +
+   labs(subtitle = "Hierarchical Clustering: Silhouette method")
```

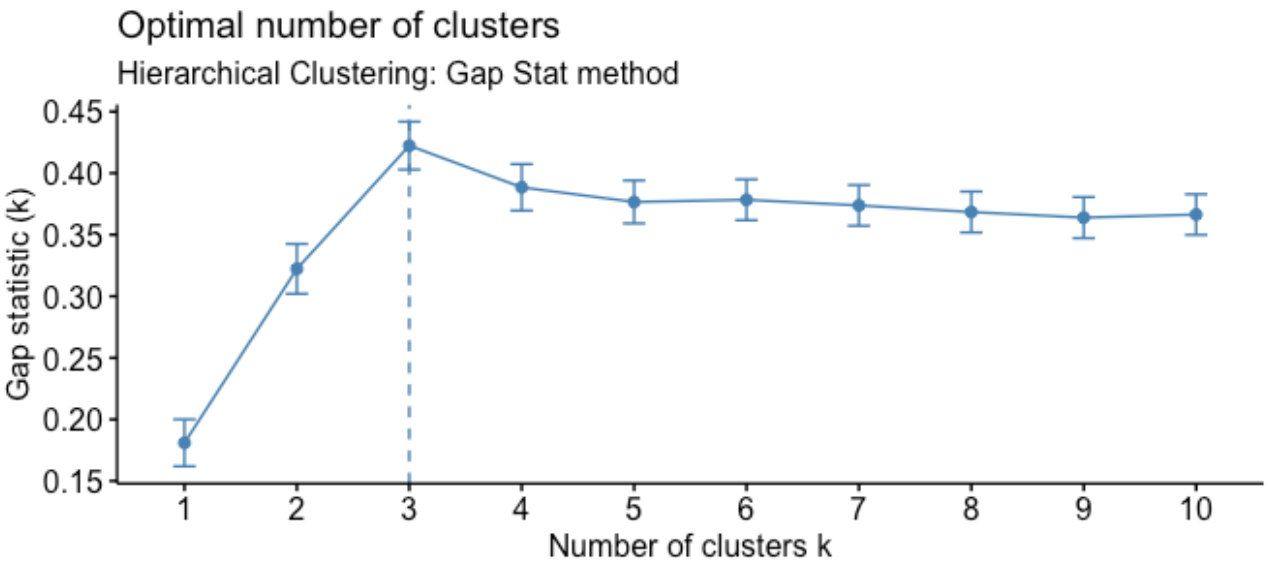



```
> fviz_nbclust(scaled_seeds, hcut, method = "wss") +
+   labs(subtitle = "Hierarchical Clustering: Elbow method")
```



Also in this case, 3 is the best value of k according to the Elbow method.

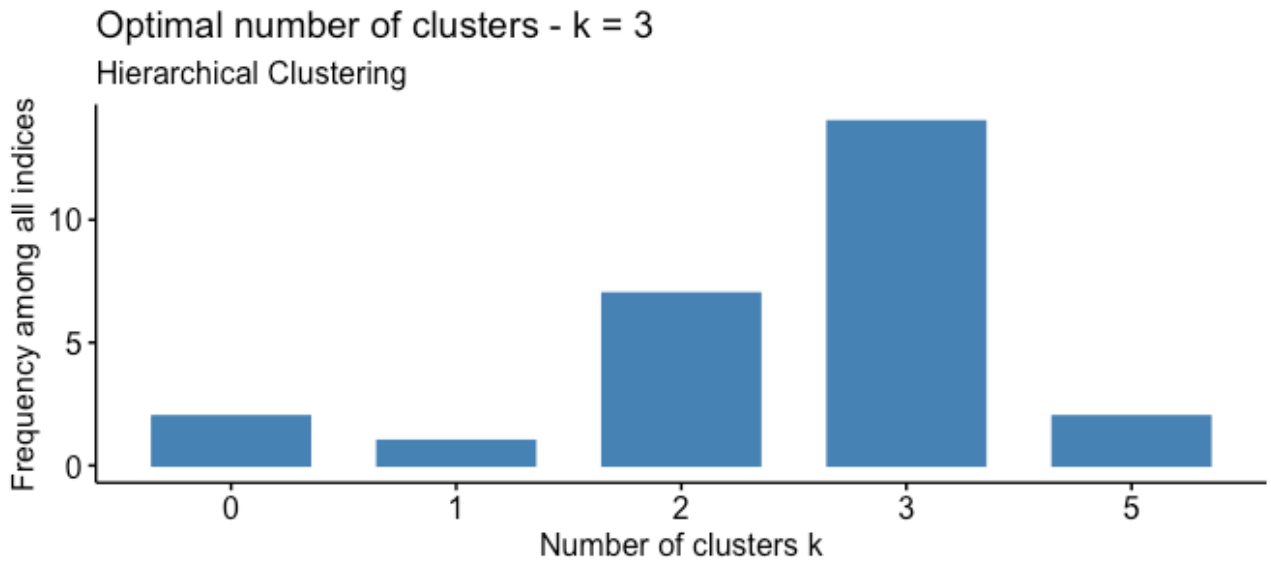
```
> fviz_nbclust(scaled_seeds, hcut, method = "gap_stat") +
+   labs(subtitle = "Hierarchical Clustering: Gap Stat method")
```




```
> nk.hc <- NbClust(scaled_seeds, diss = NULL, distance = "euclidean", min.nc =
2, max.nc = 5, method = "ward.D2")
*** : The Hubert index is a graphical method of determining the number of
clusters.
                In the plot of Hubert index, we seek a significant knee that
corresponds to a
                significant increase of the value of the measure i.e the
significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the
significant peak in Dindex
                second differences plot) that corresponds to a significant
increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 7 proposed 2 as the best number of clusters
* 14 proposed 3 as the best number of clusters
* 2 proposed 5 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3


*******************************************************************
```

According to all the method used, except the Silhouette one, the best number of clusters is 3.

```
> fviz_nbclust(nk.hc) +
+   labs(subtitle = "Hierarchical Clustering")
```



Now I can proceed and generate the clusters with all the previous method. The numbers of clusters will be 2 and 3 and then, with a process of Cluster Validation, I'll find the best method.

Generating K-means Clusters

```
> set.seed(123)
> km.res2 <- kmeans(scaled_seeds, 2, nstart = 25)
> km.res3 <- kmeans(scaled_seeds, 3, nstart = 25)
```

```
> aggregate(numerical.seeds, by=list(cluster=km.res2$cluster), mean)
  cluster     area perimeter compactness length of kernel width of kernel asymmetry coefficient length of kernel groove
1       1 18.15857  16.05481   0.8838169         6.127429        3.660519              3.480417                5.971740
2       2 12.93060  13.69346   0.8635774         5.339699        3.025917              3.827444                5.081737
> aggregate(numerical.seeds, by=list(cluster=km.res3$cluster), mean)
  cluster     area perimeter compactness length of kernel width of kernel asymmetry coefficient length of kernel groove
1       1 11.85694  13.24778   0.8482528         5.231750        2.849542              4.742389                5.101722
2       2 14.43789  14.33775   0.8815972         5.514577        3.259225              2.707341                5.120803
3       3 18.49537  16.20343   0.8842104         6.175687        3.697537              3.632373                6.041701
```

Generated the clusters, I can add a column in the original dataset containing the number of the cluster in which the observations belong.

```
> seeds.km.2 <- cbind(seeds, cluster = km.res2$cluster)
> seeds.km.3 <- cbind(seeds, cluster = km.res3$cluster)
```

Another useful information is the size of every cluster, because we know that each variety has 70 observation.

```
> km.res2$size
[1]  77 133
> km.res3$size
[1] 72 71 67
```

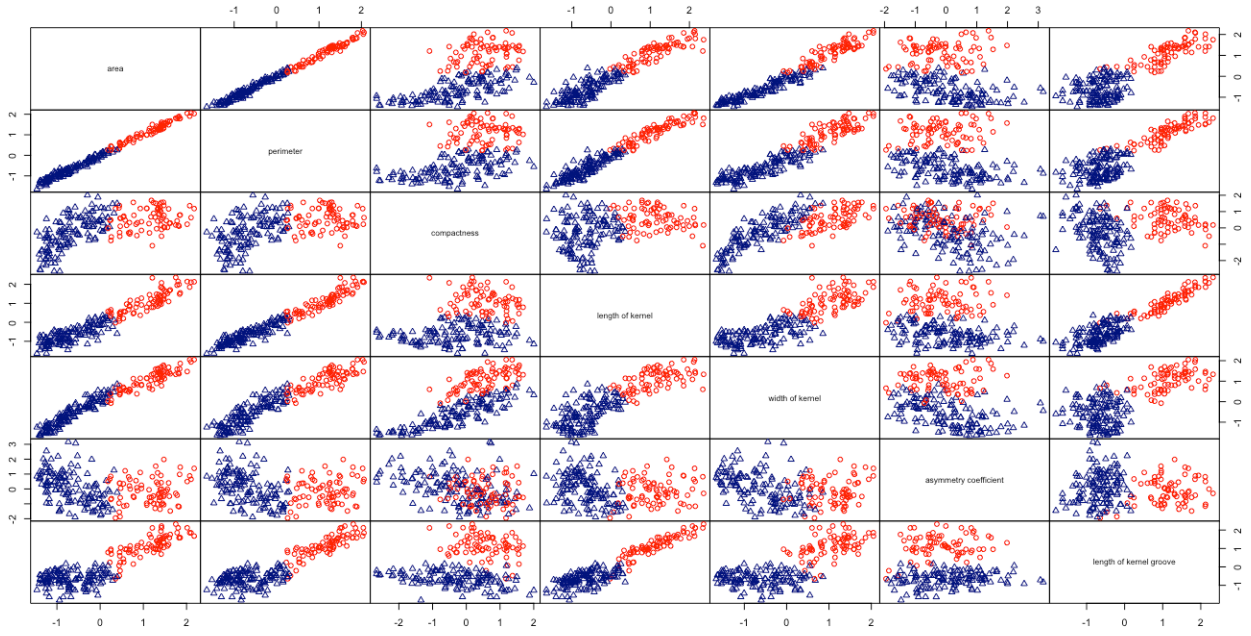
Here we can see how the observation belonging to each variety are distributed in the clusters.

```
> table(seeds$variety, km.res2$cluster)

            1  2
  Kama      9 61
  Rosa     68  2
  Canadian  0 70

> table(seeds$variety, km.res3$cluster)

            1  2  3
  Kama      6 62  2
  Rosa      0  5 65
  Canadian 66  4  0
```

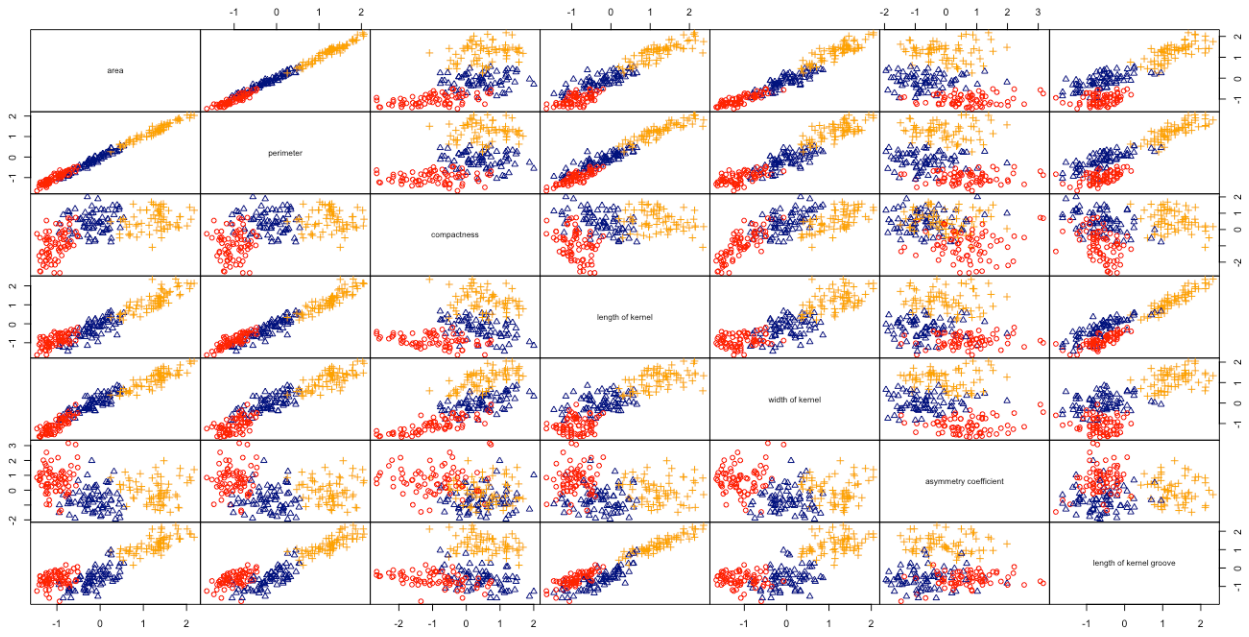
```
> km.res2$centers
       area perimeter compactness length of kernel width of kernel asymmetry coefficient length of kernel groove
1  1.1379346  1.145151   0.5424726        1.1260130       1.0640703           -0.14617607               1.1468793
2 -0.6588042 -0.662982  -0.3140631       -0.6519023      -0.6160407            0.08462825              -0.6639828
> km.res3$centers
       area  perimeter compactness length of kernel width of kernel asymmetry coefficient length of kernel groove
1 -1.0277967 -1.0042491  -0.9626050       -0.8955451    -1.082995635            0.69314821              -0.6233191
2 -0.1407831 -0.1696372   0.4485346       -0.2571999     0.001643014           -0.66034079              -0.5844965
3  1.2536860  1.2589580   0.5591283        1.2349319     1.162075101           -0.04511157               1.2892273
```

Plotting K-means clusters in the original space

```
> km.cl.2 <- km.res2$cluster
> pairs(scaled_seeds, gap=0, pch=km.cl.2, col=c("red", "navy")[km.cl.2])
```

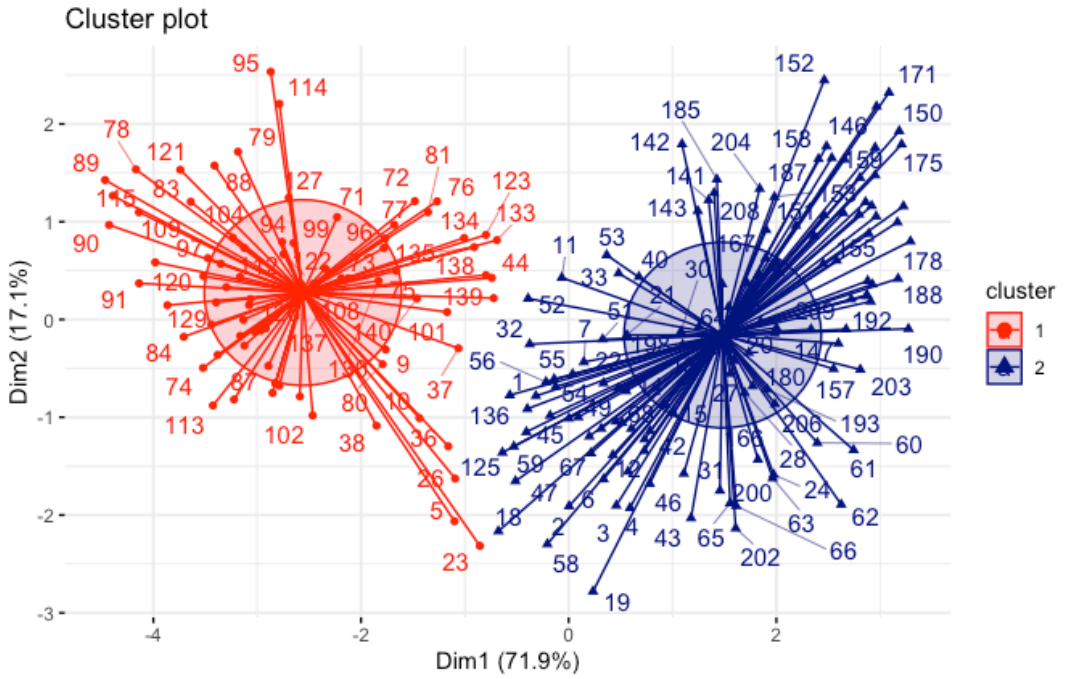


```
> km.cl.3 <- km.res3$cluster
> pairs(scaled_seeds, gap=0, pch=km.cl.3, col=c("red", "navy", "orange")
[km.cl.3])
```

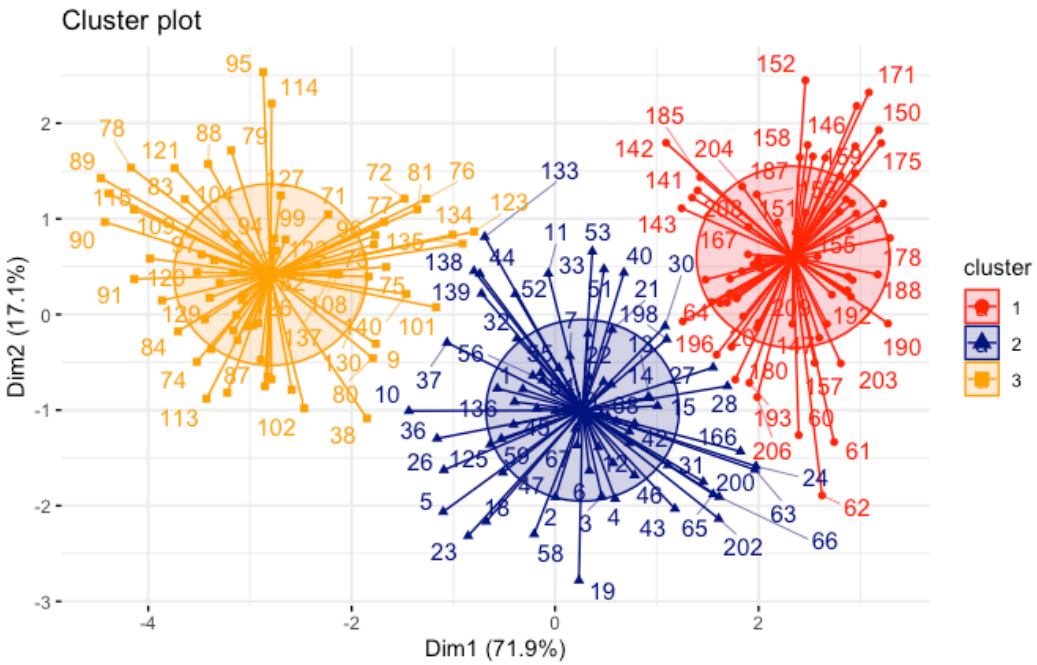


Plotting K-means clusters in the first two PCs

```
> fviz_cluster(km.res2,
+              data = scaled_seeds,
+              palette = c("red", "navy"),
+              ellipse.type = "euclid",
+              star.plot = TRUE,
+              repel = TRUE,
+              ggtheme = theme_minimal()
+ )
```



```
> fviz_cluster(km.res3,
+              data = scaled_seeds,
+              palette = c("red", "navy", "orange"),
+              ellipse.type = "euclid",
+              star.plot = TRUE,
+              repel = TRUE,
+              ggtheme = theme_minimal()
+ )
```



Generating K-medoids Clusters

```
> pam.res2 <- pam(scaled_seeds, 2)
> pam.res3 <- pam(scaled_seeds, 3)

> seeds.pam.2 <- cbind(seeds, cluster = pam.res2$cluster)
> seeds.pam.3 <- cbind(seeds, cluster = pam.res3$cluster)
```

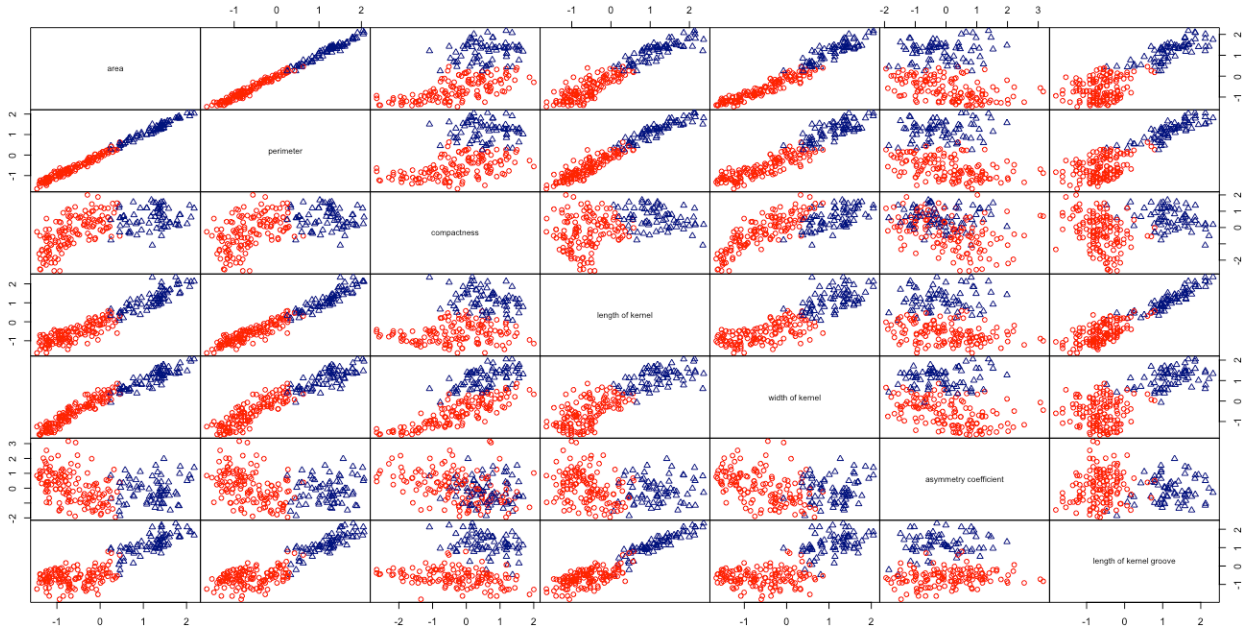
```
> pam.res2$medoids
          area  perimeter compactness length of kernel width of kernel asymmetry coefficient length of kernel groove
[1,] -0.5593443 -0.5507722  -0.1268999       -0.5270878      -0.4993316             0.3038122              -0.6512393
[2,]  1.3618163  1.3252442   0.8295350        1.4523126       1.1500626            -0.3080701               1.3122160
> pam.res3$medoids
           area  perimeter compactness length of kernel width of kernel asymmetry coefficient length of kernel groove
[1,] -0.01976967 -0.0300819   0.4613499       -0.1885358      0.08576648            -0.6625628              -0.6044419
[2,]  1.40993126  1.4247880   0.5036700        1.3981443      1.31420772            -0.2209434               1.5055095
[3,] -0.96144769 -0.8800322  -1.2441514       -0.8159854     -1.08972470             0.8565016              -0.7366954
```

```
> table(seeds$variety, pam.res2$cluster)

             1  2
  Kama      63  7
  Rosa       5 65
  Canadian 70  0
```

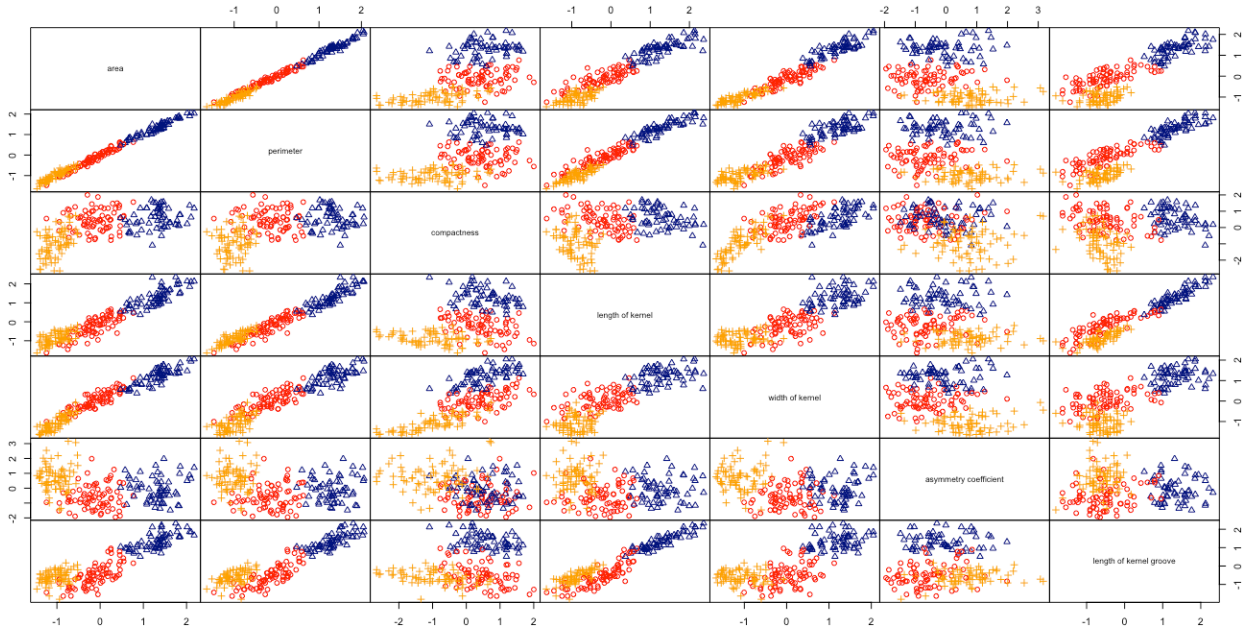
```
> table(seeds$variety, pam.res3$cluster)

             1  2  3
  Kama      64  1  5
  Rosa       9 61  0
  Canadian   4  0 66
```

Plotting K-medoids clusters in the original space

```
> pam.cl.2 <- pam.res2$clustering
> pairs(scaled_seeds, gap=0, pch=pam.cl.2, col=c("red", "navy")[pam.cl.2])
```



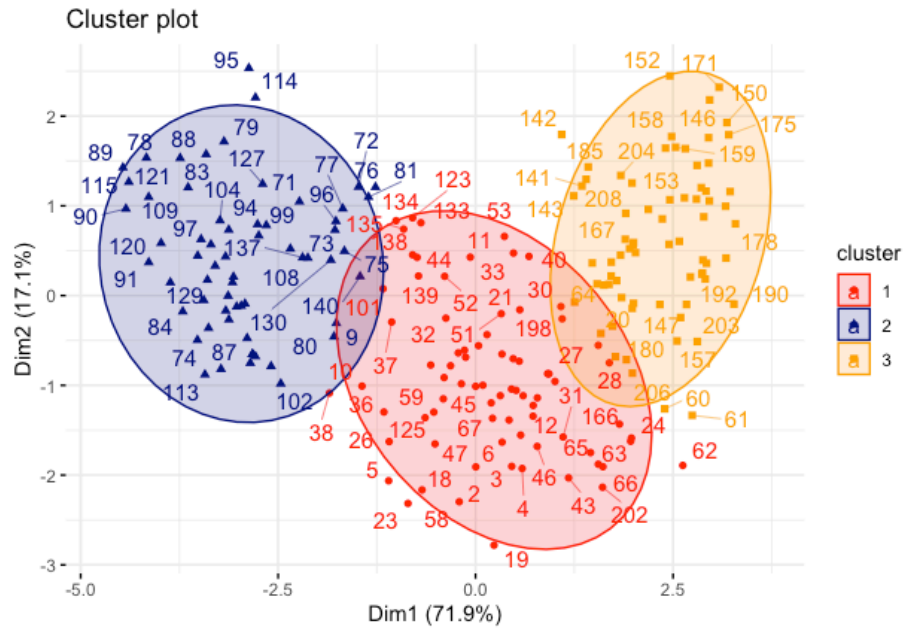

```
> pam.cl.3 <- pam.res3$clustering
> pairs(scaled_seeds, gap=0, pch=pam.cl.3, col=c("red", "navy", "orange")
[pam.cl.3])
```



Plotting K-medoids clusters in the first two PCs

```
> fviz_cluster(pam.res2,
+              palette = c("red", "navy"),
+              ellipse.type = "t",
+              repel = TRUE,
+              ggtheme = theme_minimal()
+ )
```


```
> fviz_cluster(pam.res3,
+              palette = c("red", "navy", "orange"),
+              ellipse.type = "t",
+              repel = TRUE,
+              ggtheme = theme_minimal()
+ )
```



Generating Hierarchical Clusters

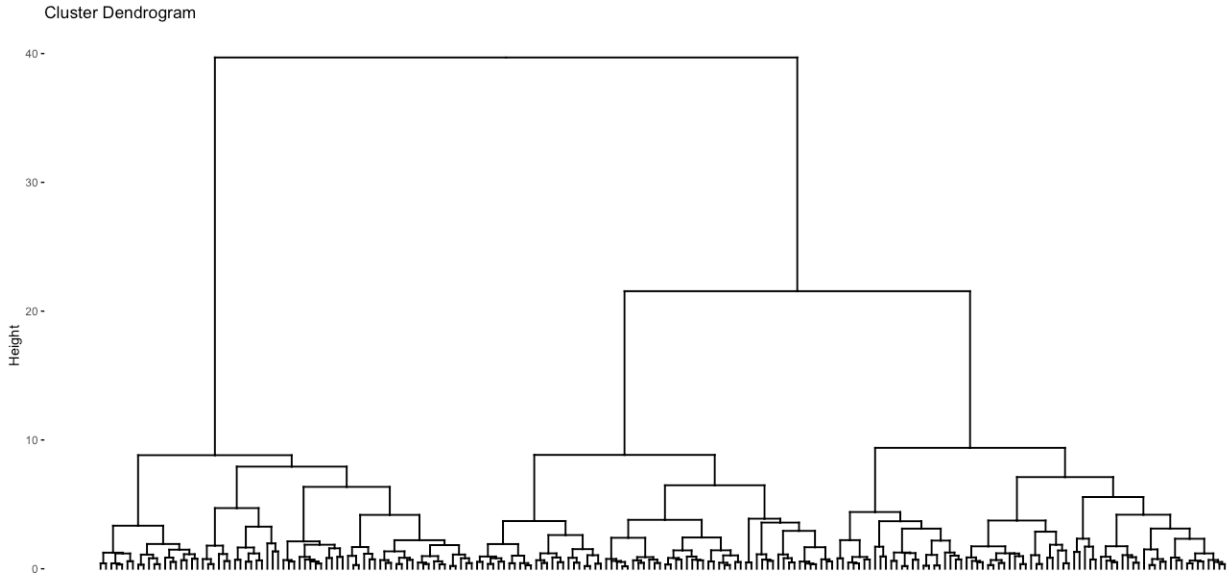
First of all, we have to compute the dissimilarity matrix of our data.

```
> diss.seeds <- dist(scaled_seeds, method = "euclidean")
```

Then, we can try with all the Hierarchical Clustering method, in order to compute and find the best Cophenetic distance, a measure of how similar are two units, in order to be grouped in the same cluster.

Ward.D2 method

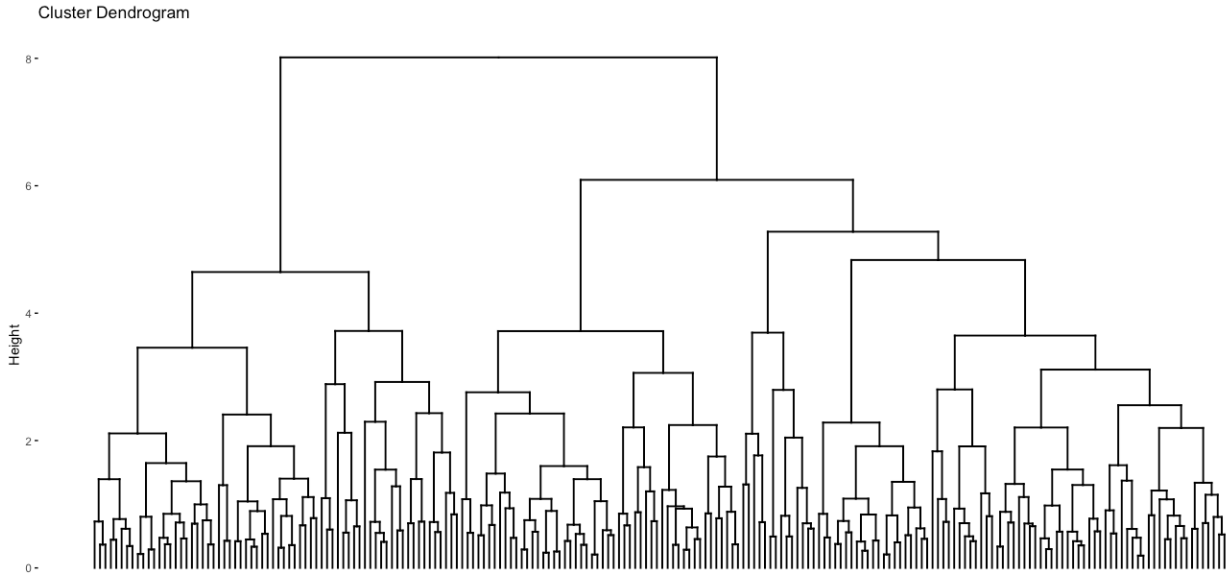
```
> ward.d2.seeds <- hclust(d = diss.seeds, method = "ward.D2")
> fviz_dend(ward.d2.seeds, show_labels = F)
```




```
> coph.ward.seeds <- cophenetic(ward.d2.seeds)
> cor(diss.seeds, coph.ward.seeds)
[1] 0.7285529
```

Complete linkage method

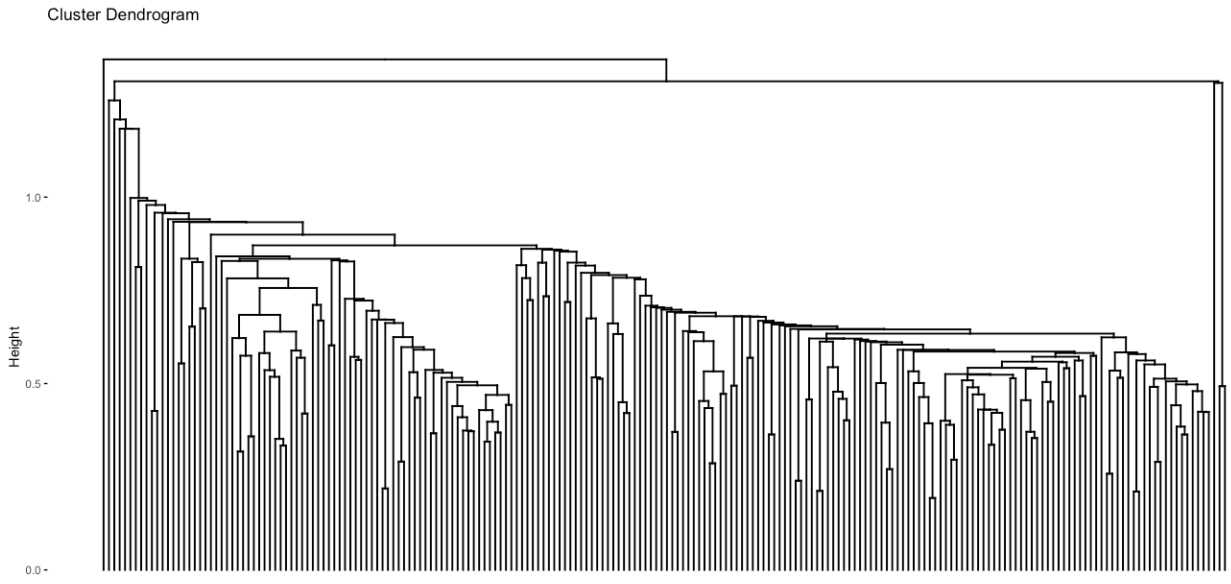
```
> comp.seeds <- hclust(d = diss.seeds, method = "complete")
> fviz_dend(comp.seeds, show_labels = F)
```




```
> coph.comp.seeds <- cophenetic(comp.seeds)
> cor(diss.seeds, coph.comp.seeds)
[1] 0.7129629
```

Single linkage method

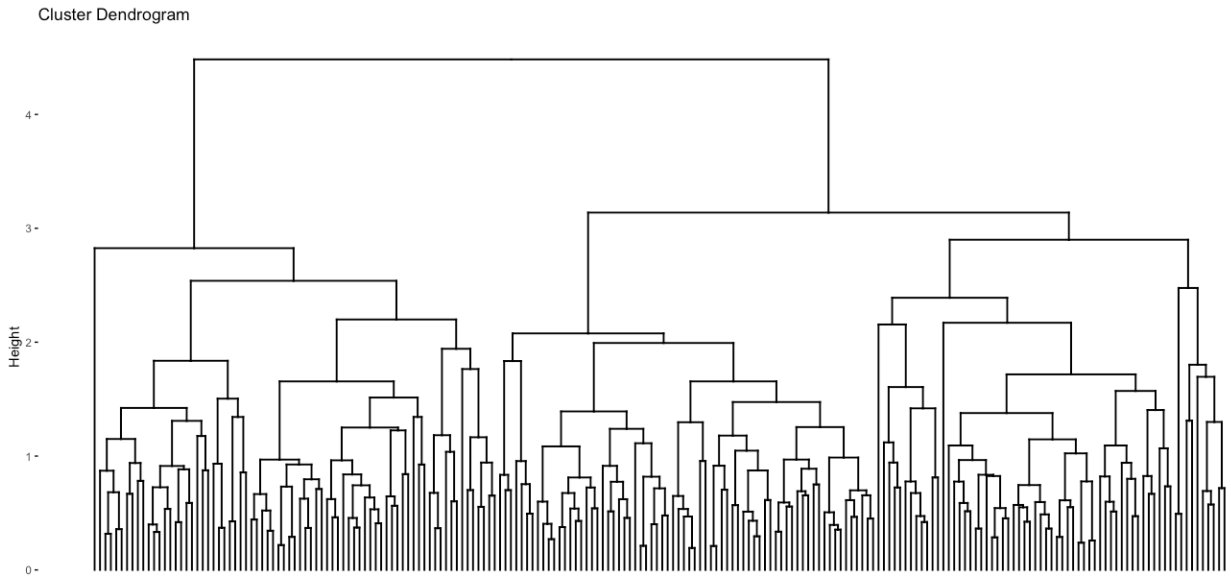
```
> single.seeds <- hclust(d = diss.seeds, method = "single")
> fviz_dend(single.seeds, show_labels = F)
```



```
> coph.single.seeds <- cophenetic(single.seeds)
> cor(diss.seeds, coph.single.seeds)
[1] 0.4266838
```

Average linkage method

```
> avg.seeds <- hclust(d = diss.seeds, method = "average")
> fviz_dend(avg.seeds, show_labels = F)
```



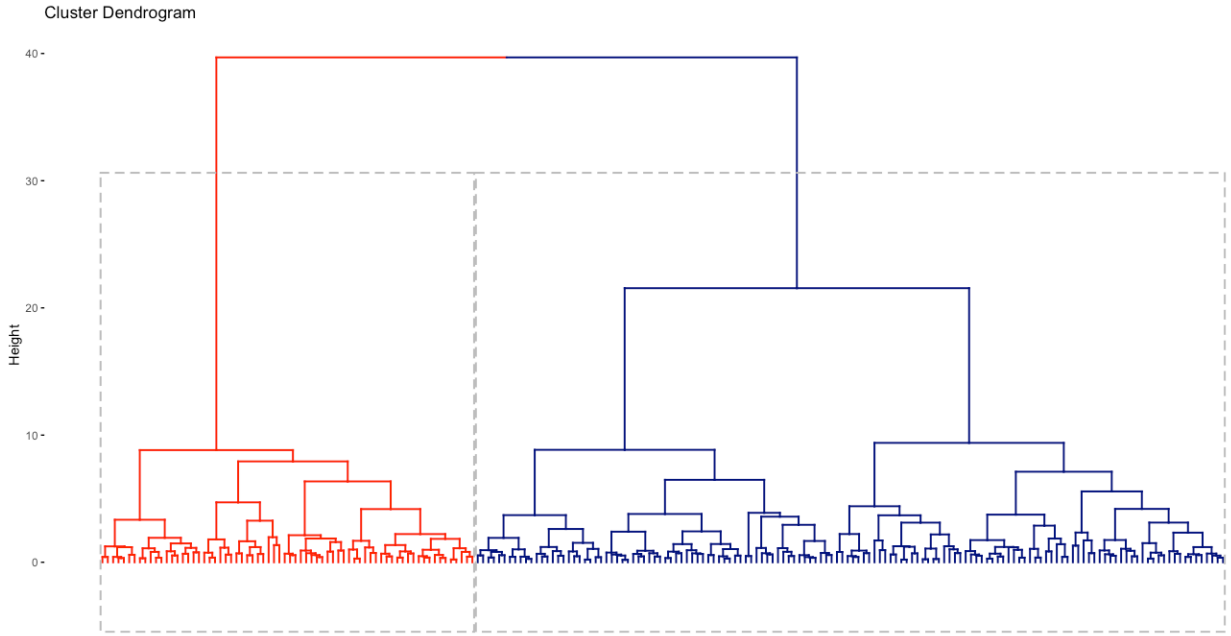

```
> coph.avg.seeds <- cophenetic(avg.seeds)
> cor(diss.seeds, coph.avg.seeds)
[1] 0.7146889
```

The best value of Cophenetic distance is the one obtained with the Ward.D2 method.

Hierarchical Clustering with k = 2

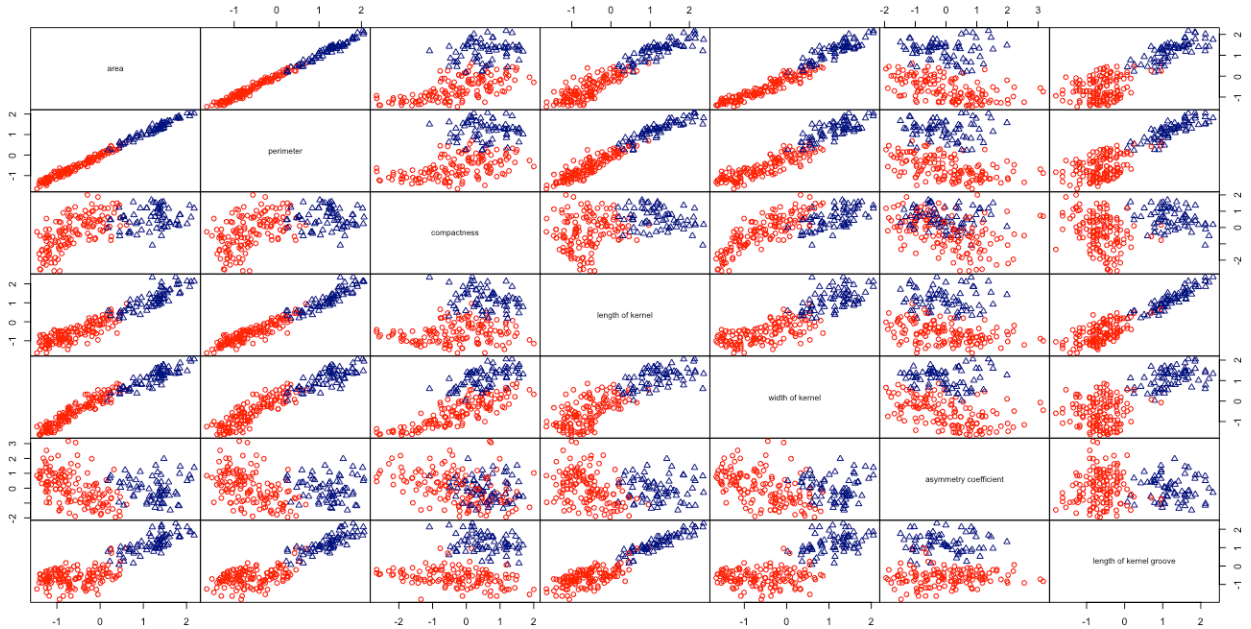
```
> hc.seeds.2 <- cutree(ward.d2.seeds, k = 2)
> table(hc.seeds.2)
hc.seeds.2
  1   2
140  70
> table(seeds$variety, hc.seeds.2)
          hc.seeds.2
            1  2
  Kama     66  4
  Rosa      4 66
  Canadian 70  0
```

```
> fviz_dend(ward.d2.seeds,
+           k = 2,
+           show_labels = F,
+           k_colors = c("red", "navy"),
+           rect = T)
```



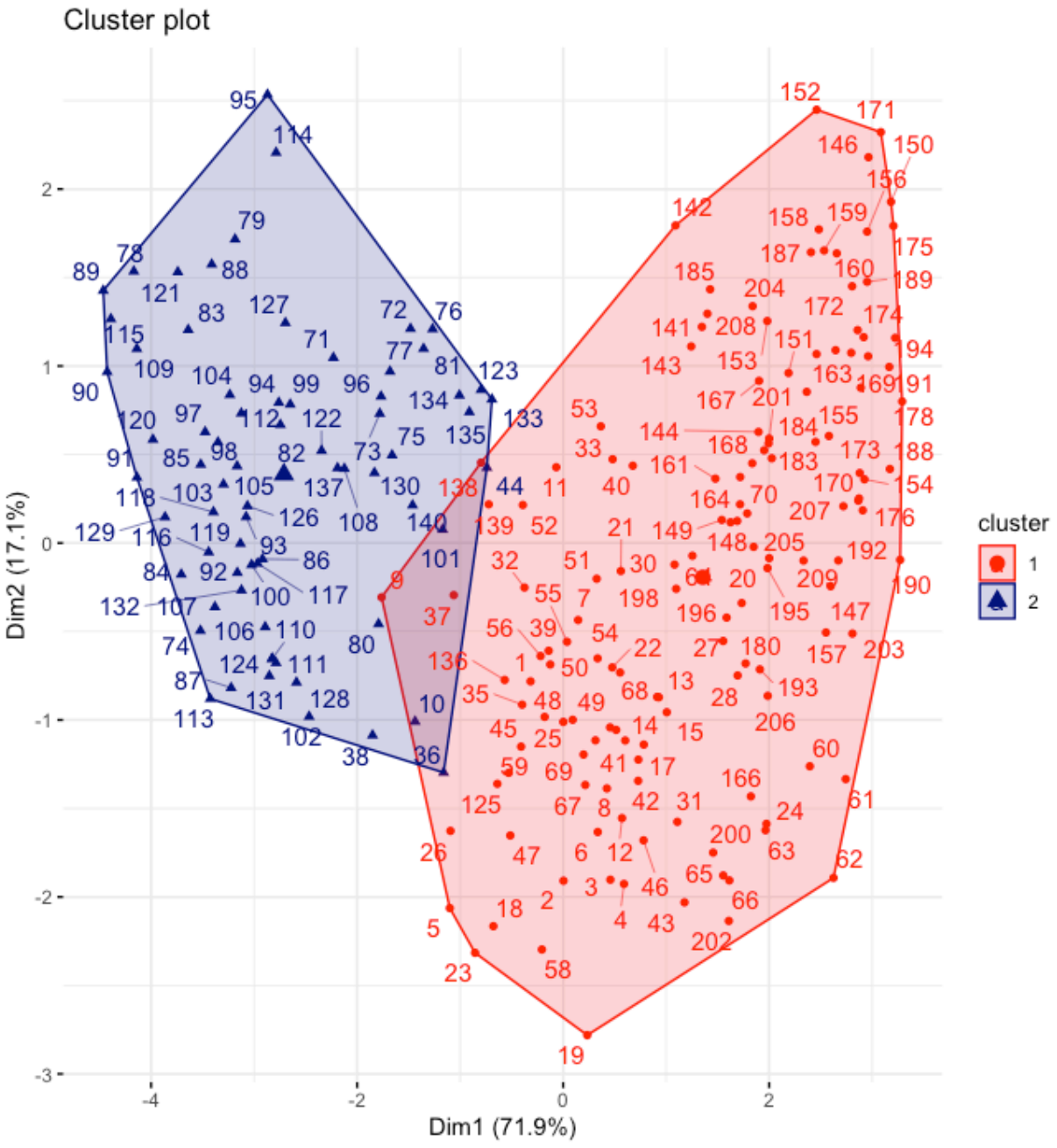
Plotting Clusters in the original space

```
> pairs(scaled_seeds, gap=0, pch=hc.seeds.2, col=c("red", "navy")[hc.seeds.2])
```



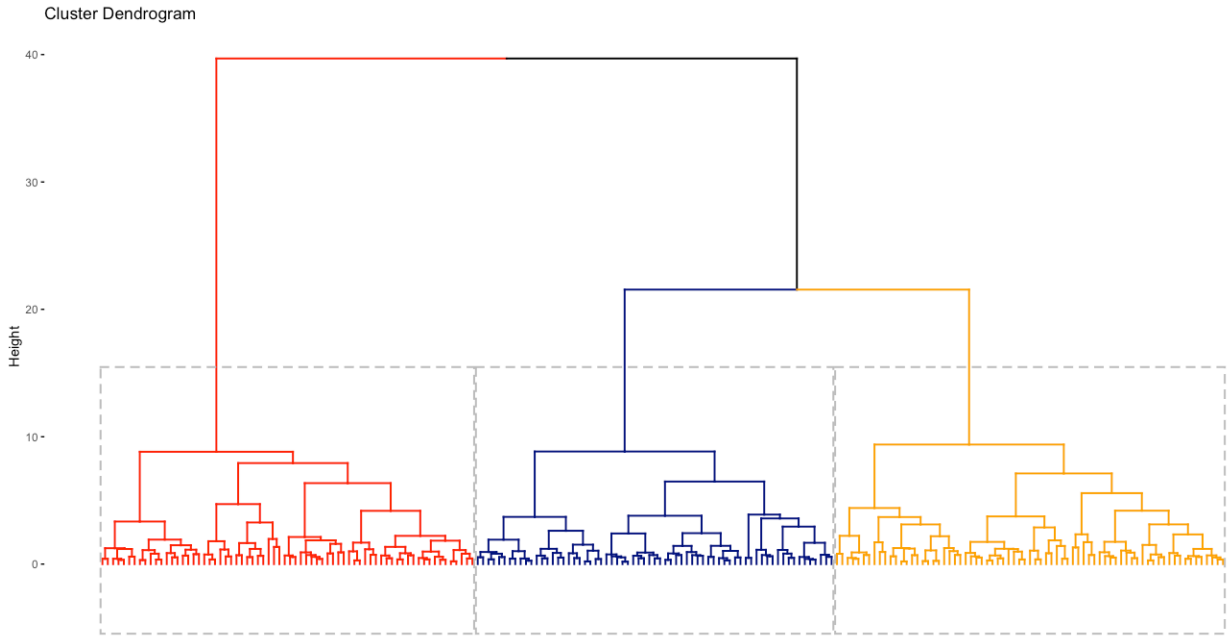
Plotting Clusters in the first two PCs

```
> fviz_cluster(list(data = scaled_seeds, cluster = hc.seeds.2),
+              palette = c("red", "navy"),
+              ellipse.type = "convex",
+              repel = TRUE,
+              ggtheme = theme_minimal()
+ )
```



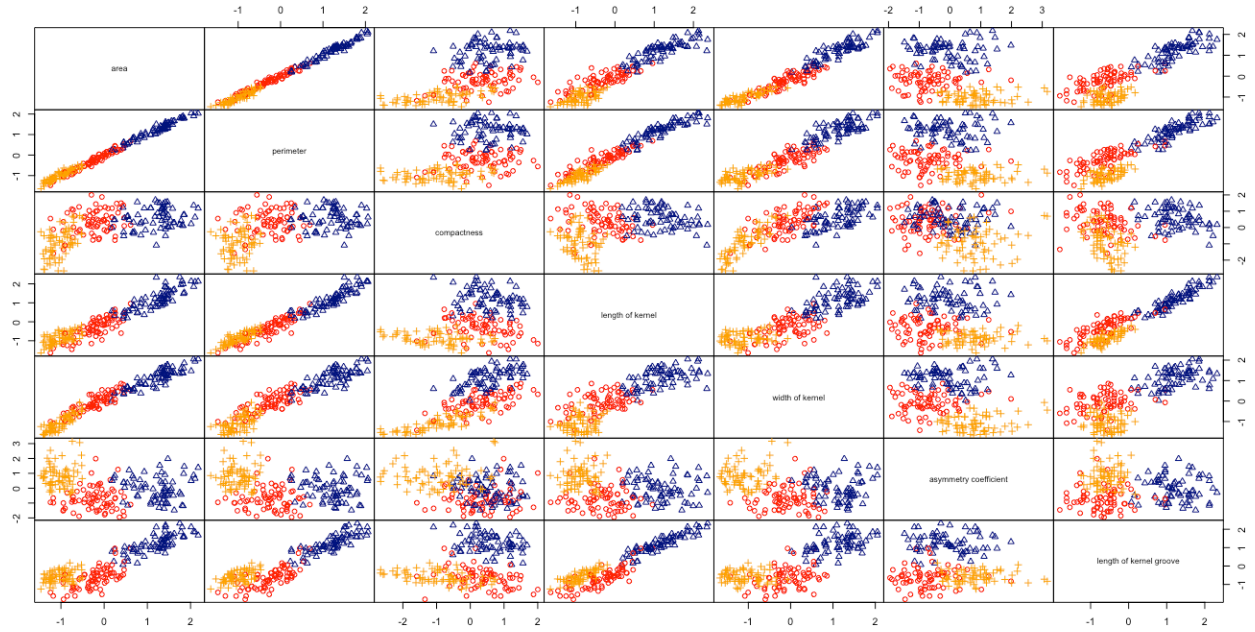
Hierarchical Clustering with k = 3

```
> hc.seeds.3 <- cutree(ward.d2.seeds, k = 3)
> table(hc.seeds.3)
hc.seeds.3
 1  2  3
73 70 67
> table(seeds$variety, hc.seeds.3)
          hc.seeds.3
            1  2  3
  Kama     64  4  2
  Rosa      4 66  0
  Canadian  5  0 65

> fviz_dend(ward.d2.seeds,
+           k = 3,
+           show_labels = F,
+           k_colors = c("red", "navy", "orange"),
+           rect = T)
```



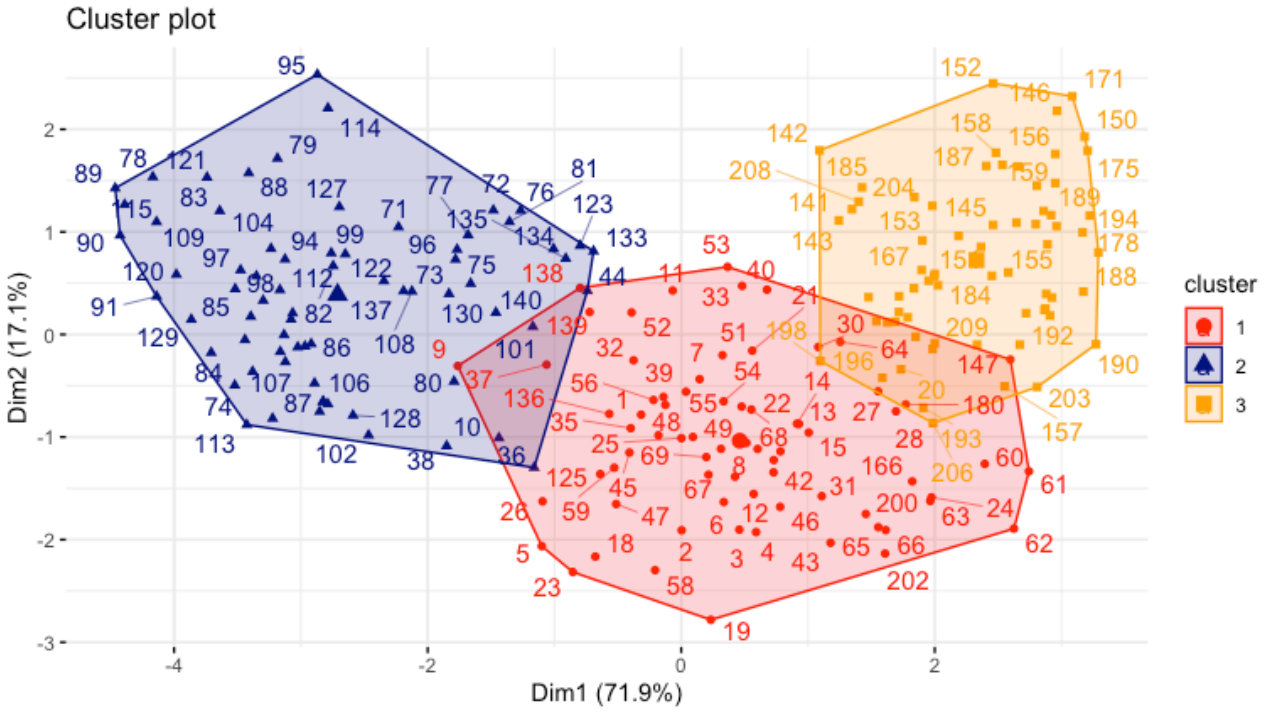
Plotting Clusters in the original space

```
> pairs(scaled_seeds, gap=0, pch=hc.seeds.3, col=c("red", "navy",
"orange")[hc.seeds.3])
```



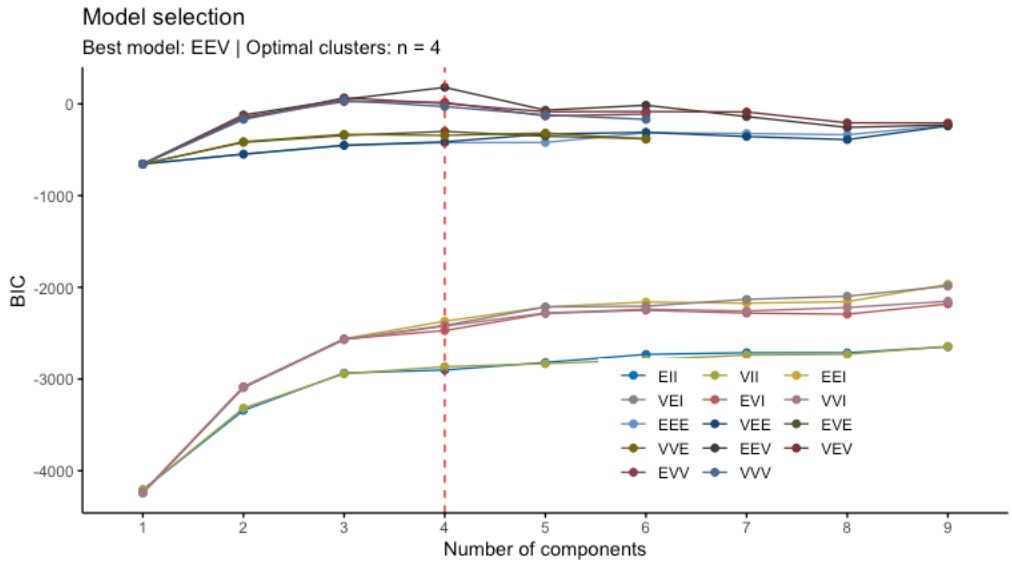
Plotting Clusters in the first two PCs

```
> fviz_cluster(list(data = scaled_seeds, cluster = hc.seeds.3),
+              palette = c("red", "navy", "orange"),
+              ellipse.type = "convex",
+              repel = TRUE,
+              ggtheme = theme_minimal()
+ )
```



Model Based Clustering

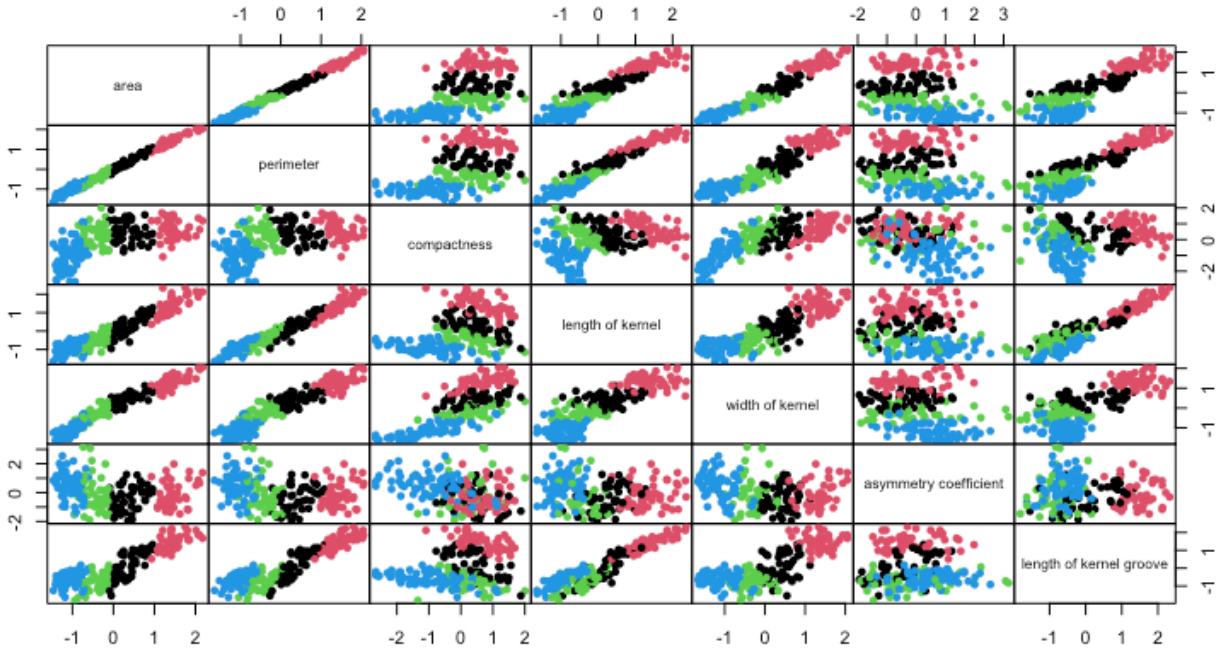
```
> model.seeds <- Mclust(scaled_seeds, G = 1:9, modelNames = NULL)
> summary(model.seeds$BIC)
Best BIC values:
            EEV,4      VEV,3      EEV,3
BIC      179.9839   66.00606   48.81966
BIC diff   0.0000 -113.97782 -131.16422
> summary(model.seeds)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust EEV (ellipsoidal, equal volume and shape) model with 4 components:

 log-likelihood   n  df      BIC      ICL
       416.1655 210 122 179.9839 172.5306

Clustering table:
 1  2  3  4
50 49 45 66

> table(seeds$variety, model.seeds$classification)

             1  2  3  4
  Kama      29  0 38  3
  Rosa      21 49  0  0
  Canadian   0  0  7 63

> adjustedRandIndex(seeds$variety, model.seeds$classification)
[1] 0.5795368

> fviz_mclust(model.seeds, "BIC", palette = "jco")
```



According to this analysis, the best Gaussian mixture model for our data is the EEV, characterized by equal volume, equal shape and variable orientation, with 4 clusters.

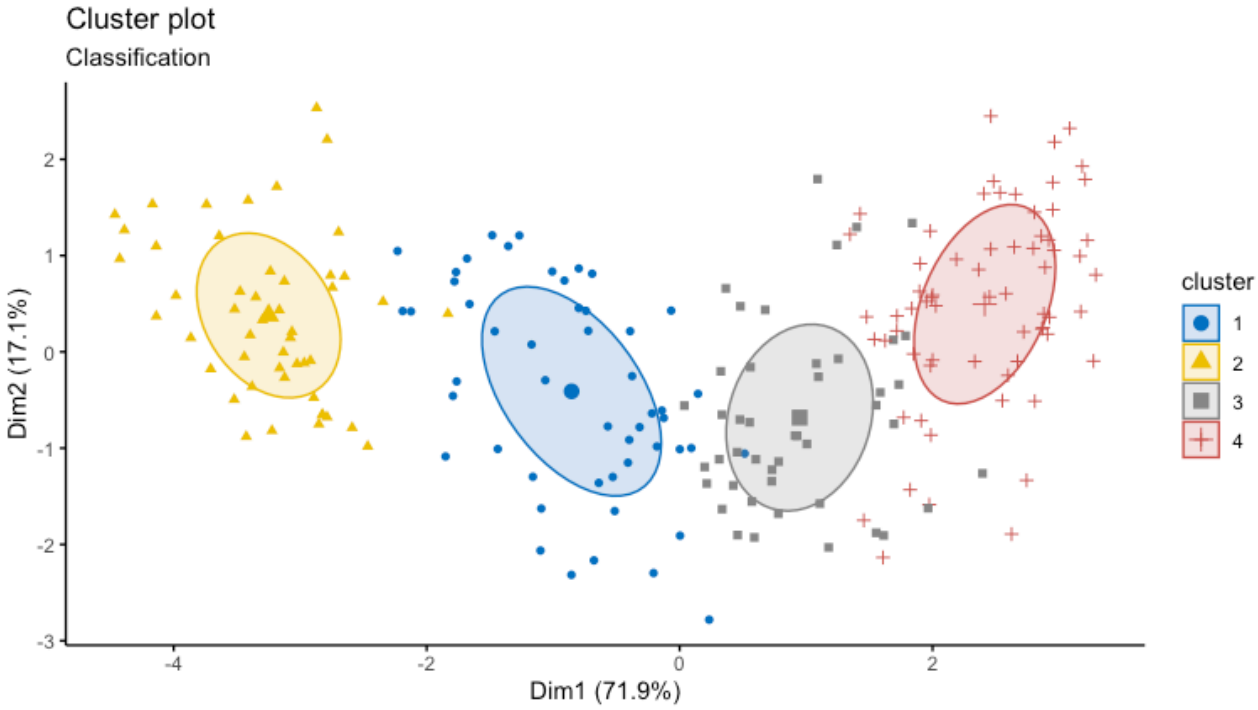
Plotting the clusters in the original space

```
> pairs(scaled_seeds, gap=0, pch = 16, col = model.seeds$classification)
```

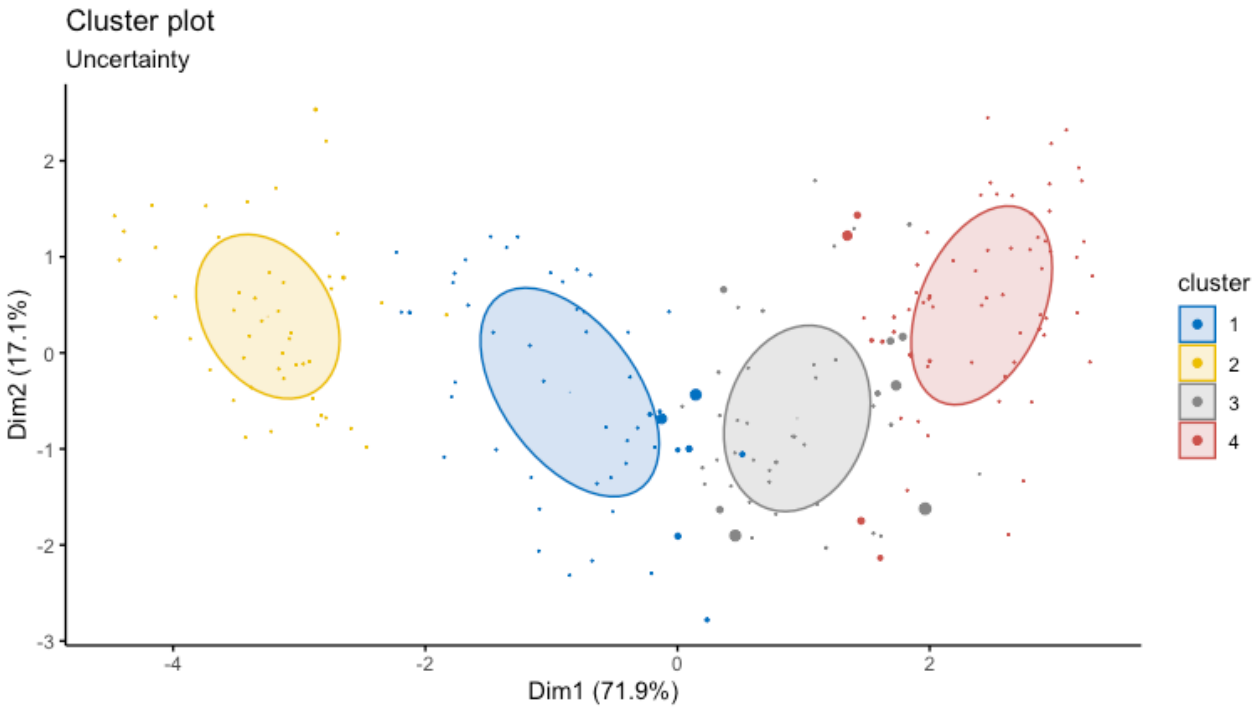


Plotting the clusters in the first two PCs

```
> fviz_mclust(model.seeds, "classification", geom = "point", pointsize = 1.5,
palette = "jco")
```



```
> fviz_mclust(model.seeds, "uncertainty", palette = "jco")
```



Cluster Validation

The Cluster Validation will be conducted using two criterions: internal and external. The internal validation allows us to know which is the best method and the optimal number of clusters, without using external informations.

```
> clmethods <- c("kmeans", "pam", "hierarchical", "model")
> ClusterValidation <- clValid(scaled_seeds, nClust = 2:5, clMethods =
clmethods, validation = "internal")
> summary(ClusterValidation)

Clustering Methods:
 kmeans pam hierarchical model

Cluster sizes:
 2 3 4 5

Validation Measures:
                                    2        3        4        5

kmeans       Connectivity   16.8964  39.5476  63.6409  79.9706
             Dunn            0.0870   0.1188   0.0888   0.0814
             Silhouette      0.4658   0.4007   0.3379   0.2881
pam          Connectivity   28.5028  41.4413  55.4944  73.2921
             Dunn            0.0588   0.1115   0.1114   0.0816
             Silhouette      0.4648   0.3982   0.3335   0.2678
hierarchical Connectivity   17.7754  39.1413  42.9798  46.1087
             Dunn            0.1132   0.0997   0.1058   0.1058
             Silhouette      0.4413   0.3760   0.3549   0.2752
model        Connectivity   33.3202  54.4615  90.3782 124.8873
             Dunn            0.0440   0.0710   0.0675   0.0713
             Silhouette      0.4429   0.3728   0.2572   0.2023
```

```
Optimal Scores:

             Score   Method Clusters
Connectivity 16.8964 kmeans 2
Dunn          0.1188 kmeans 3
Silhouette    0.4658 kmeans 2
```

We could easily see that, for example, the Model Based Clustering wasn't a good method. But I needed a process of Cluster Validation to have certain information about the internal validity of the other method. The results of this process show that, according to all the optimal scores, the K-means is the best method, but the response about the best number of clusters is not so clear. From the previous analysis I thought that the optimal number of clusters was 3, but the Cluster Validation provided a different result. In conclusion, I think that three is a good number of clusters, but two is the best from an internal point of view. It's important because in our case, we know that the observations are categorized into three variables.

```
> table(seeds$variety, km.res2$cluster)

            1  2
  Kama      9 61
  Rosa     68  2
  Canadian  0 70
```

We can see with the function table that all Canadian seeds and most of the Kama belong to the second cluster. Now, I can proceed with the external validation. This analysis is useful because we know in which variety the observations belong, so I can confront the clusters obtained by the different methods with the information about variety. First of all, I had to threat variety as a vector of number.

```
variety <- as.numeric(seeds$variety)
```

Then, I used the cluster.stats function of the fpc library to obtain the information I needed: the Corrected Rand Index and the Meila's Variation Index. The first is a measure of similarity between the two methods analyzed, while the second tells us how different they are. So, according to the external cluster validation, the best clustering method is the one with the highest Corrected Rand index and the lowest Meila's Variation Index.

```
> km.stats2 <- cluster.stats(d = dist(numerical.seeds), variety,
km.res2$cluster)
> km.stats3 <- cluster.stats(d = dist(numerical.seeds), variety,
km.res3$cluster)

> pam.stats2 <- cluster.stats(d = dist(numerical.seeds), variety,
pam.res2$cluster)
> pam.stats3 <- cluster.stats(d = dist(numerical.seeds), variety,
pam.res3$cluster)

> hc.stats2 <- cluster.stats(d = dist(numerical.seeds), variety, hc.seeds.2)
> hc.stats3 <- cluster.stats(d = dist(numerical.seeds), variety, hc.seeds.3)
```

Now, I could easily extract the two indexes. For example:

```
> km.stats2$corrected.rand
[1] 0.4805276
> km.stats2$vi
[1] 0.783723
```

The previous are, respectively, the values of the Corrected Rand Index and Meila's Variation Index for the K-means algorithm with k = 2. I computed all the indexes for all the algorithms used and put them into a table to simplify the analysis.

```
> external.validation <- data.frame(c(km.stats2$corrected.rand, km.stats2$vi),
c(km.stats3$corrected.rand, km.stats3$vi), c(pam.stats2$corrected.rand,
pam.stats2$vi), c(pam.stats3$corrected.rand, pam.stats3$vi),
c(hc.stats2$corrected.rand, hc.stats2$vi), c(hc.stats3$corrected.rand,
hc.stats3$vi))
> colnames(external.validation) <- c("K-means 2", "K-means 3", "K-medoids 2",
"K-medoids 3", "HC 2", "HCl 3")
> rownames(external.validation) <- c("Rand", "VI")
> external.validation
     K-means 2 K-means 3 K-medoids 2 K-medoids 3      HC 2     HCl 3
Rand 0.4805276 0.7732937   0.4440428   0.7469561 0.4762028 0.7969983
VI   0.7837230 0.5978132   0.8439676   0.6264608 0.7541415 0.5495578
```

The table shows that Hierarchical Clustering algorithm with k = 3 is the best method, according to the external validation. This means that the clusters created with this method are the most similar to the variety of the seeds dataframe.