P6

brunop31

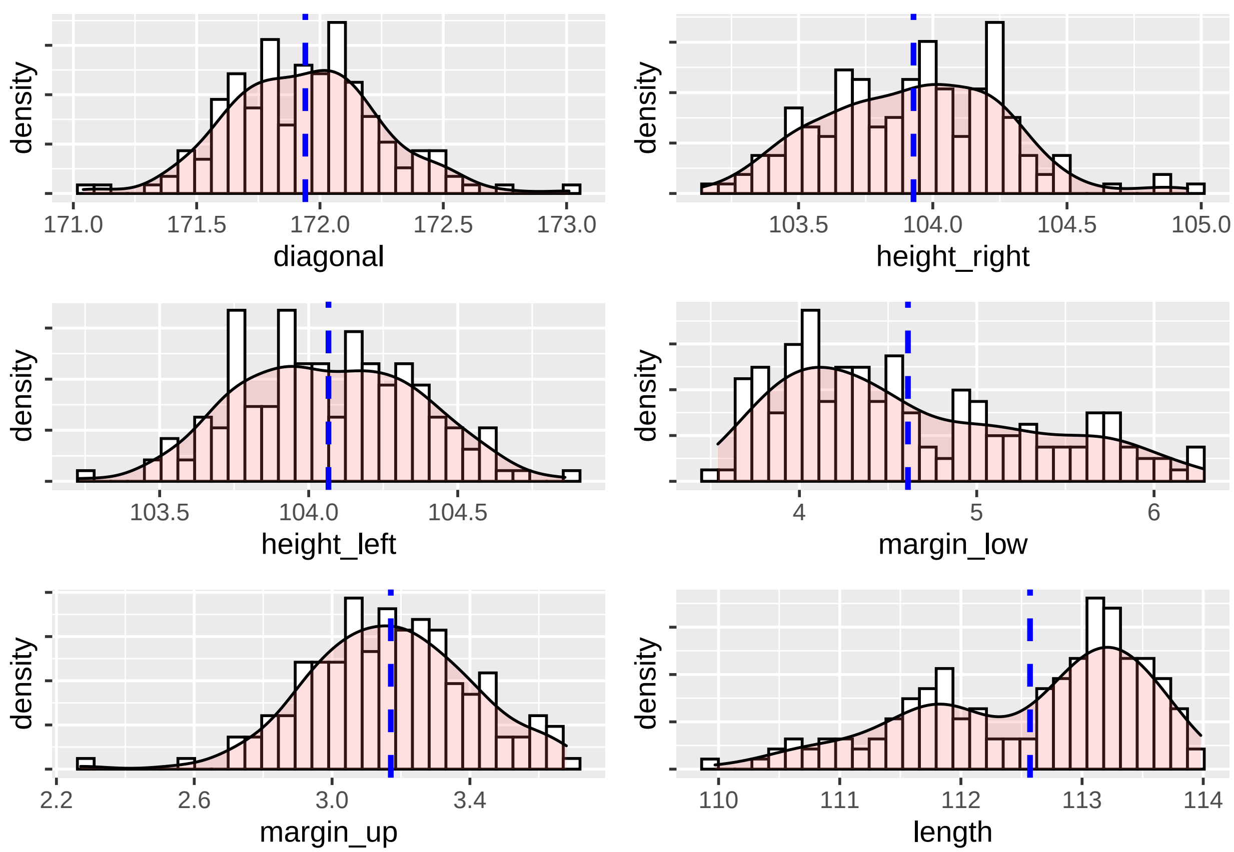
22/06/2020

```r
library(dplyr)
library(FactoMineR)
library(factoextra)
library(gridExtra)
library(corrplot)
library(caret)
library(MASS)
library(tibble)
library(ggpubr)

select <- dplyr::select
```

```r
###Je récupère les données
df<-read.csv("billets.csv", encoding = "UTF-8")

###Je trace les courbe de densité des billets pour chaque variable
u<-ggplot(df, aes(diagonal))+
geom_histogram(aes(y=..density..), bins = 30,
colour="black", fill="white")+
geom_density(alpha=.2, fill="#FF6666")+
geom_vline(aes(xintercept=mean(diagonal)),
color="blue", linetype="dashed", size=1)+
theme(axis.text.y = element_blank())

v<-ggplot(df, aes(height_right))+
geom_histogram(aes(y=..density..), bins = 30,
colour="black", fill="white")+
geom_density(alpha=.2, fill="#FF6666")+
geom_vline(aes(xintercept=mean(height_right)),
color="blue", linetype="dashed", size=1)+
theme(axis.text.y = element_blank())

w<-ggplot(df, aes(height_left))+
geom_histogram(aes(y=..density..), bins = 30,
colour="black", fill="white")+
geom_density(alpha=.2, fill="#FF6666")+
geom_vline(aes(xintercept=mean(height_left)),
color="blue", linetype="dashed", size=1)+
theme(axis.text.y = element_blank())

x<-ggplot(df, aes(margin_low))+
```

```
geom_histogram(aes(y=..density..), bins = 30,
colour="black", fill="white")+
geom_density(alpha=.2, fill="#FF6666")+
geom_vline(aes(xintercept=mean(margin_low)),
color="blue", linetype="dashed", size=1)+
theme(axis.text.y = element_blank())

y<-ggplot(df, aes(margin_up))+
geom_histogram(aes(y=..density..), bins = 30,
colour="black", fill="white")+
geom_density(alpha=.2, fill="#FF6666")+
geom_vline(aes(xintercept=mean(margin_up)),
color="blue", linetype="dashed", size=1)+
theme(axis.text.y = element_blank())

z<-ggplot(df, aes(length))+
geom_histogram(aes(y=..density..), bins = 30,
colour="black", fill="white")+
geom_density(alpha=.2, fill="#FF6666")+
geom_vline(aes(xintercept=mean(length)),
color="blue", linetype="dashed", size=1)+
theme(axis.text.y = element_blank())
```

```
grid.arrange(u,v,w,x,y,z, ncol=2, nrow = 3)
```



```r
###Je test la normalité de mes données
shapiro.test(df$diagonal)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$diagonal
## W = 0.99318, p-value = 0.6107
```

```r
shapiro.test(df$height_left)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$height_left
## W = 0.99272, p-value = 0.5533
```

```r
shapiro.test(df$height_right)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$height_right
## W = 0.98812, p-value = 0.1625
```

```r
shapiro.test(df$margin_low)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$margin_low
## W = 0.9354, p-value = 6.226e-07
```

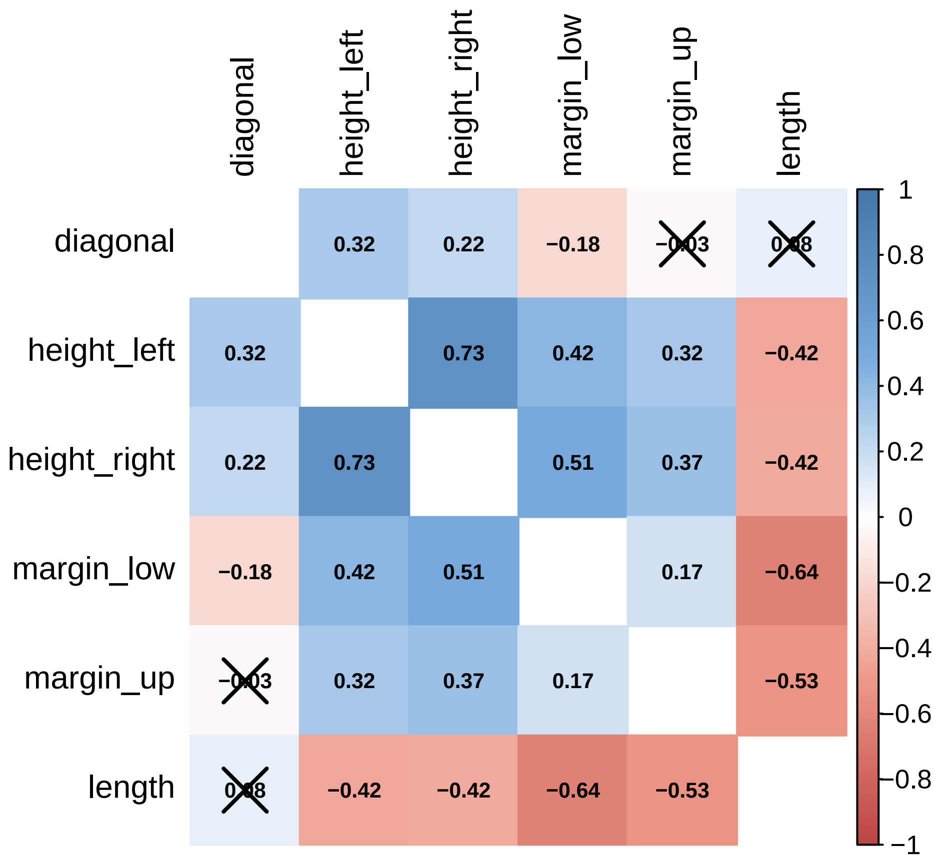
```r
shapiro.test(df$margin_up)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$margin_up
## W = 0.98892, p-value = 0.2044
```

```r
shapiro.test(df$length)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$length
## W = 0.93246, p-value = 3.715e-07
```

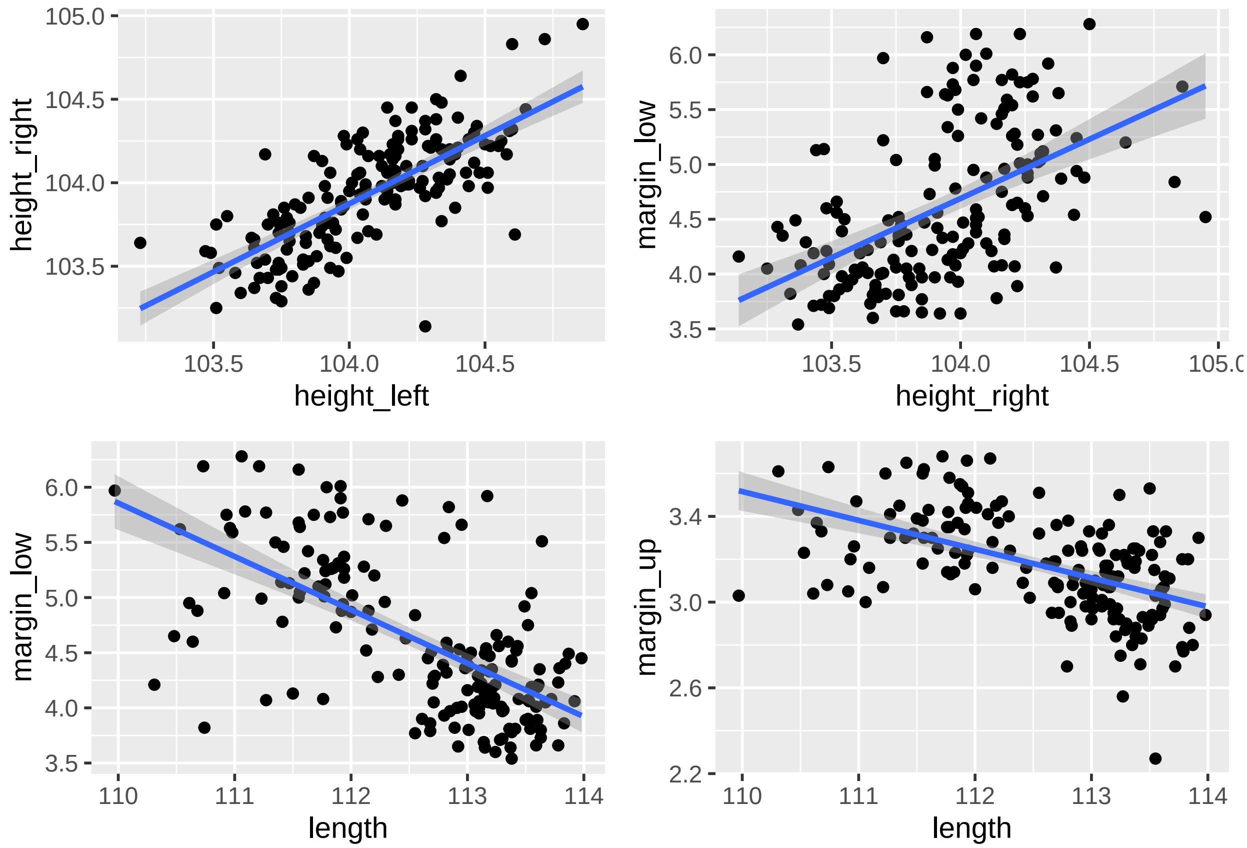
```r
###je réalise la matrice des corrélation de ces variables
df_cor<-select(df,-"is_genuine")
cormat <- cor(df_cor, method = "pearson")
p.mat <- cor.mtest(df_cor)$p
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD",
"#4477AA"))
corrplot(cormat, method = "color", col = col(200),
type = "full", order = "original", number.cex = .7,
addCoef.col = "black", # Add coefficient of correlation
tl.col = "black", tl.srt = 90, # Text label color and rotation
# Combine with significance
p.mat = p.mat, sig.level = 0.05,
# hide correlation coefficient on the principal diagonal
diag = FALSE)
```




```r
###J'observe les liens entre mes variables les plus corrélées
a<-ggplot(df, aes(height_left, height_right))+geom_point()+
geom_smooth(method = lm)
b<-ggplot(df, aes(height_right,margin_low))+geom_point()+
geom_smooth(method = lm)
c<-ggplot(df, aes(length, margin_low))+geom_point()+
geom_smooth(method = lm)
d<-ggplot(df, aes(length, margin_up))+geom_point()+
geom_smooth(method = lm)
```

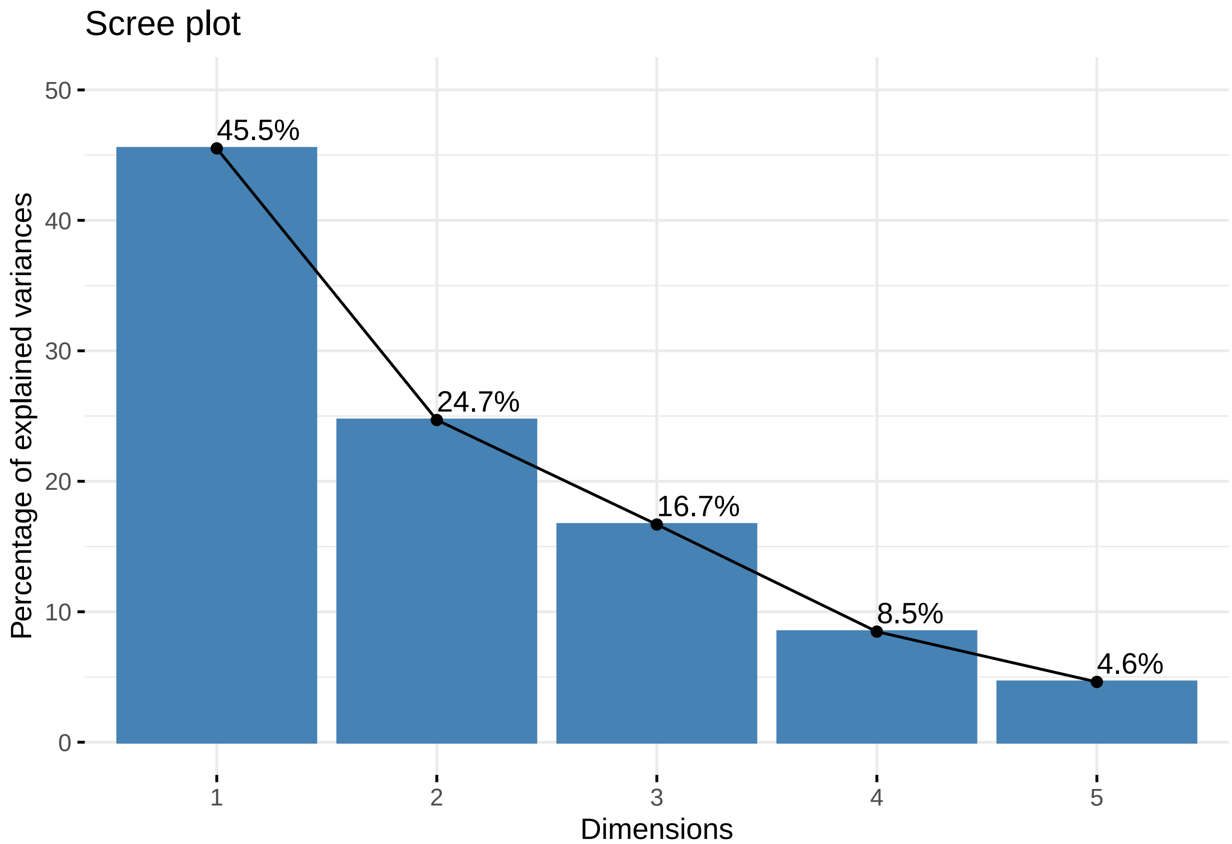
```r
grid.arrange(a,b,c,d, ncol=2, nrow = 2)
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



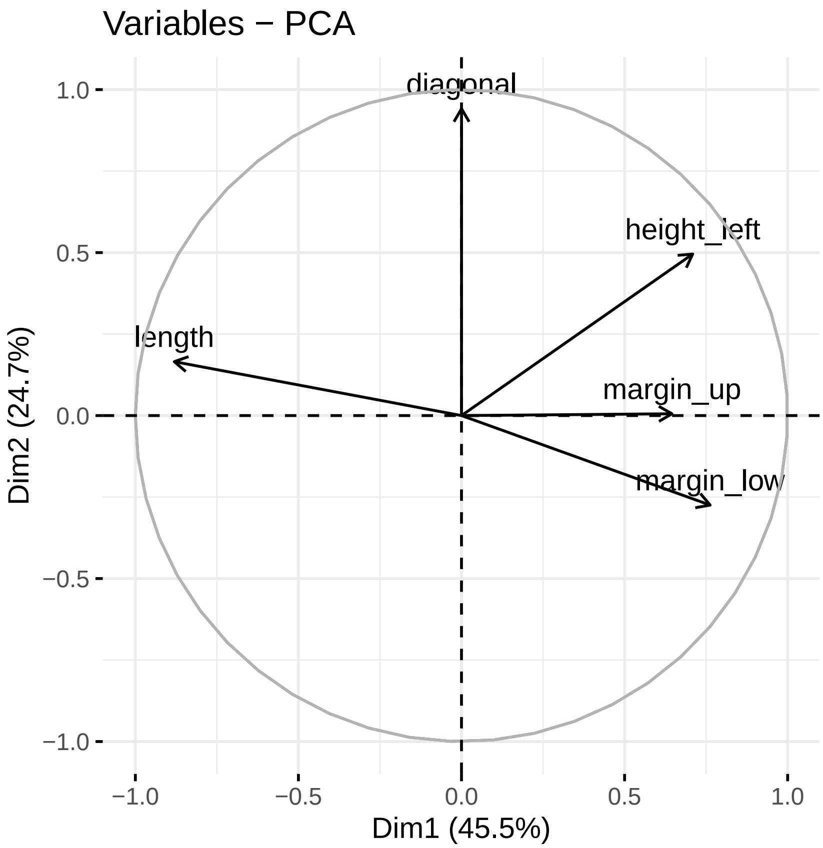

```r
###J'effectue une pca sur l'ensemble de mes données
res.pca <- PCA(df_cor%>%select(-"height_right"), graph = FALSE, scale.unit = TRUE, ncp = 3)
```

```r
###J'observe le pourcentage de variance expliquée de chacun de mes
###axes obtenue par pca
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 50))
```

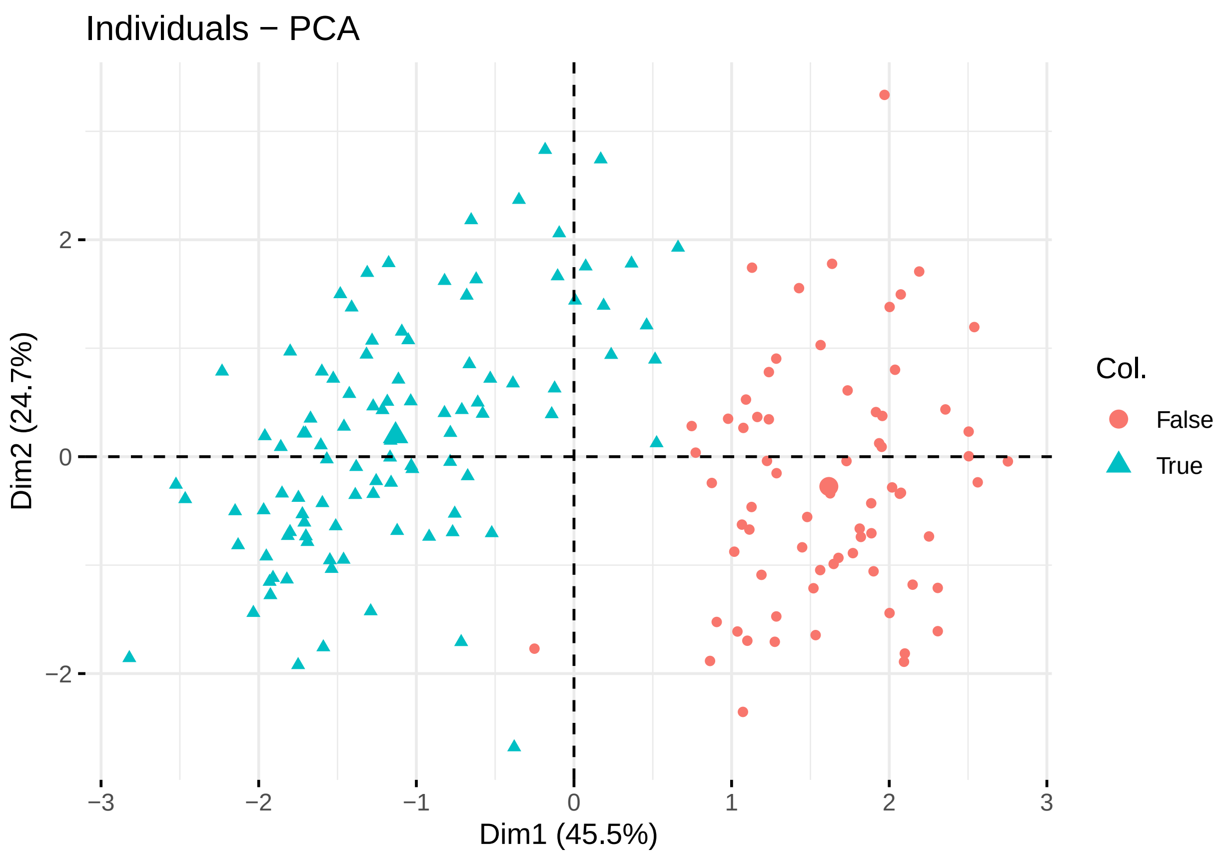



```
###Je choisis d'oberver 3 plans (axe 1 et 2 // 2 et 3 // 1 et 3)
```

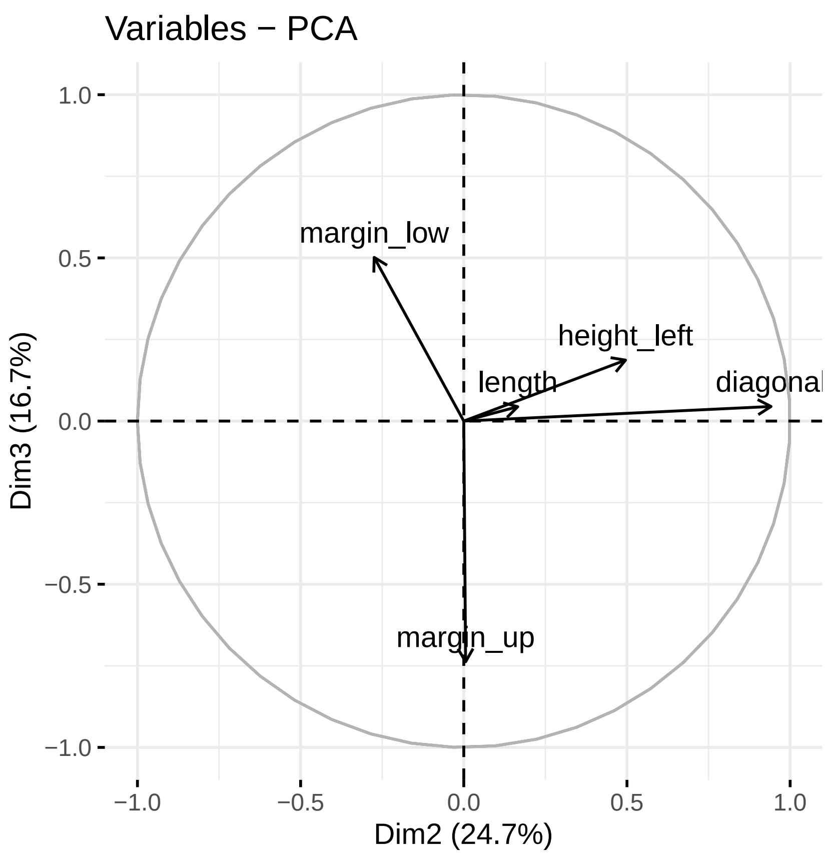
```
###cercle de corrélation du premier plan
fviz_pca_var(res.pca)
```



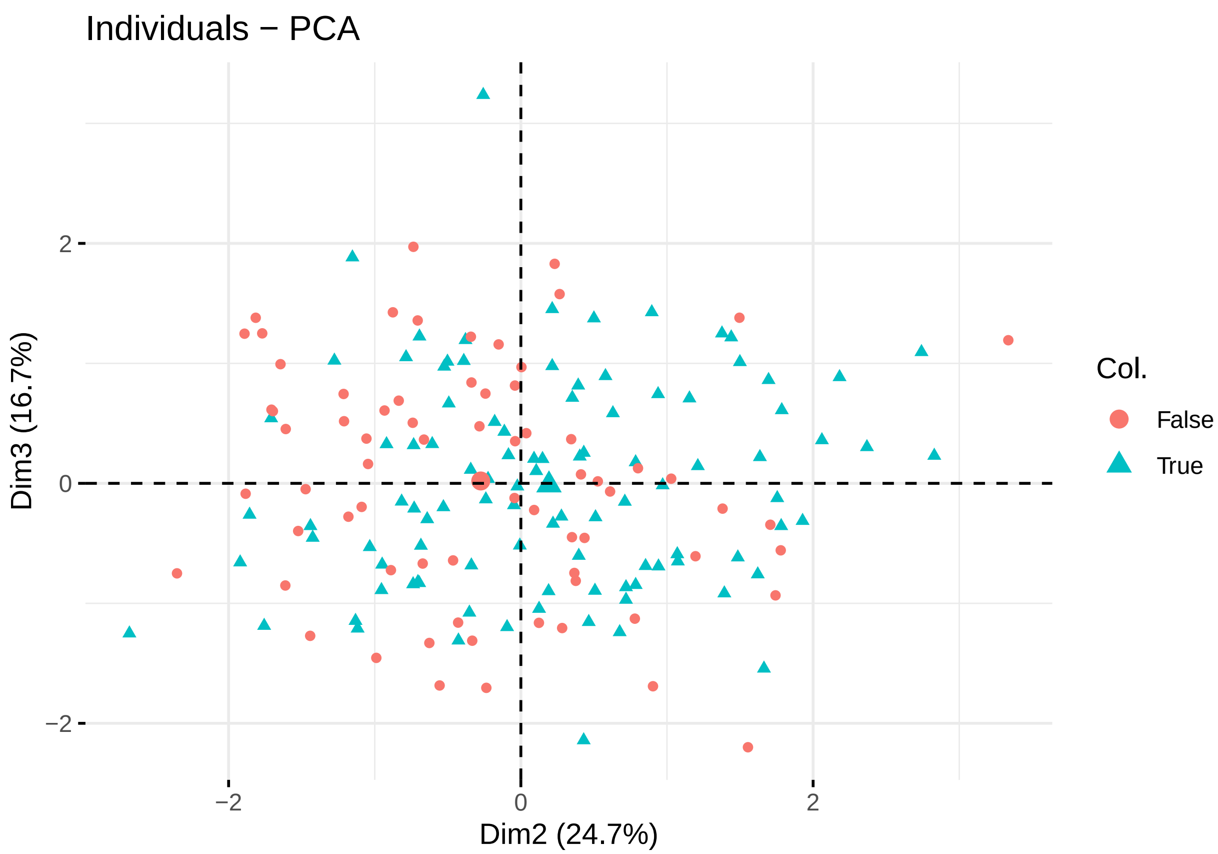

```r
###premier plan (1 et 2)
fviz_pca_ind(res.pca,geom.ind = "point", col.ind = df$is_genuine)
```



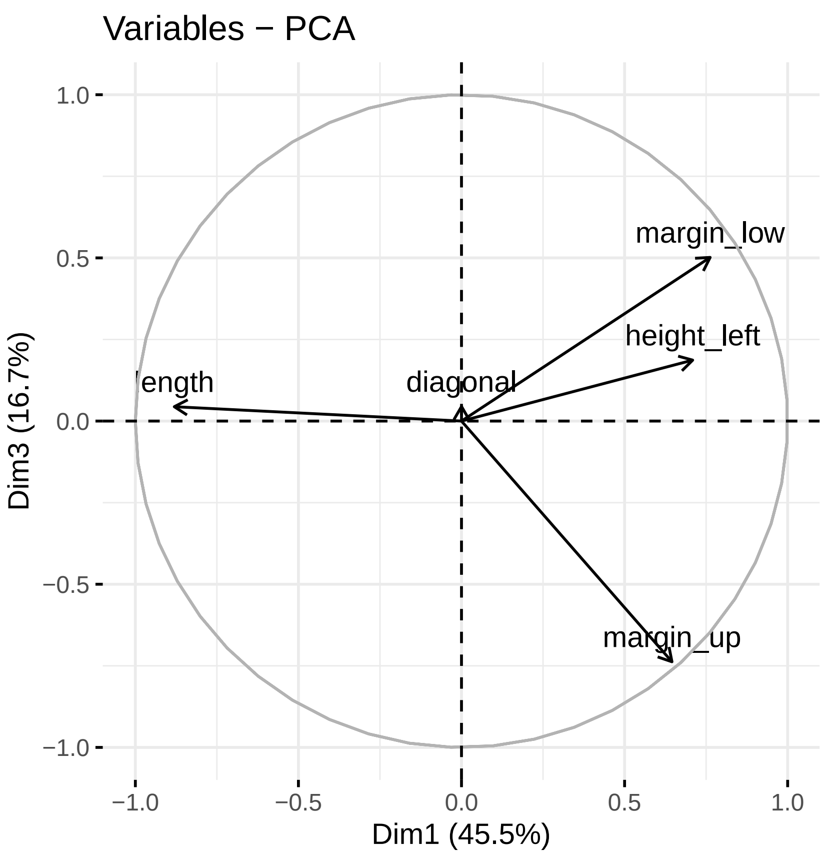

```r
###cercle de corrélation du deuxieme plan
fviz_pca_var(res.pca, axes = c(2,3))
```



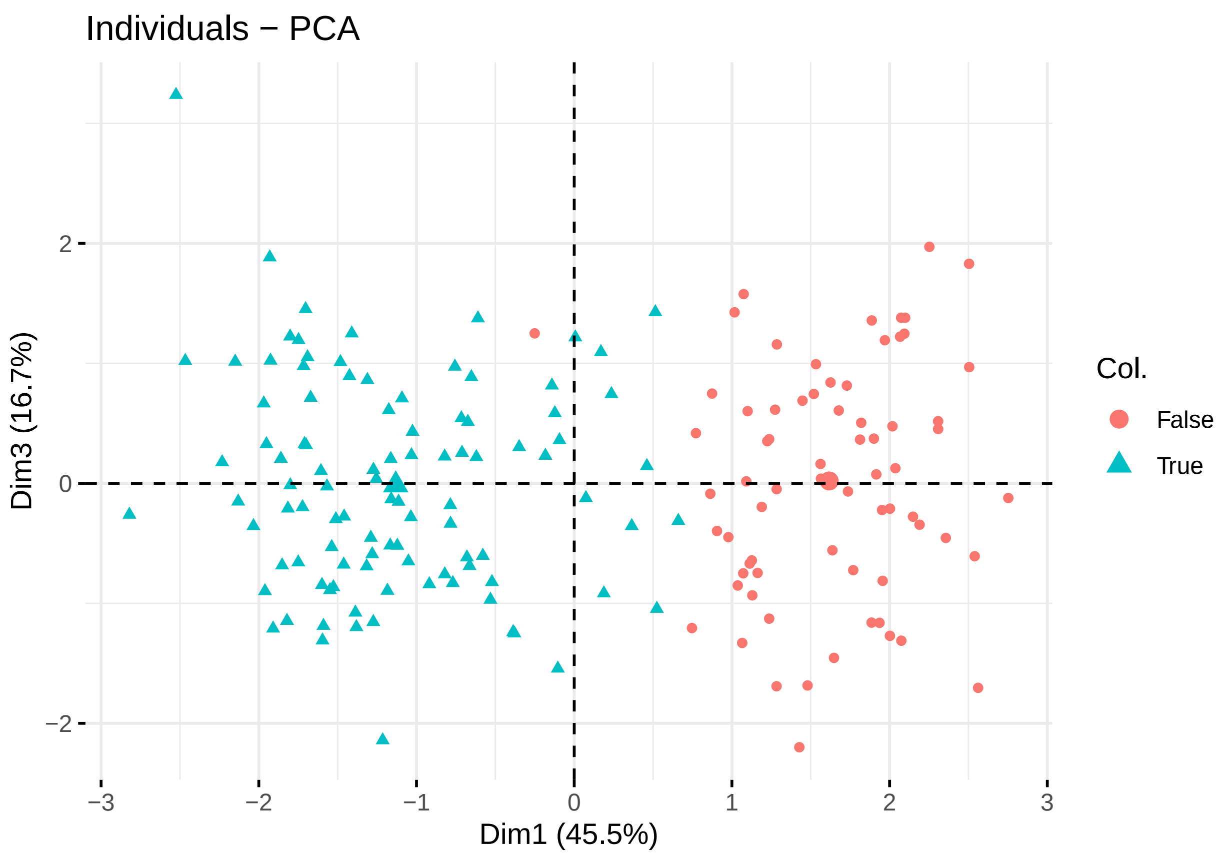

```r
###premier plan (2 et 3)
fviz_pca_ind(res.pca,geom.ind = "point", col.ind = df$is_genuine,
axes = c(2,3))
```




```r
###cercle de corrélation du troisième plan
fviz_pca_var(res.pca, axes = c(1,3))
```




```r
###troisieme plan (1 et 3)
fviz_pca_ind(res.pca,geom.ind = "point", col.ind = df$is_genuine,
axes = c(1,3))
```

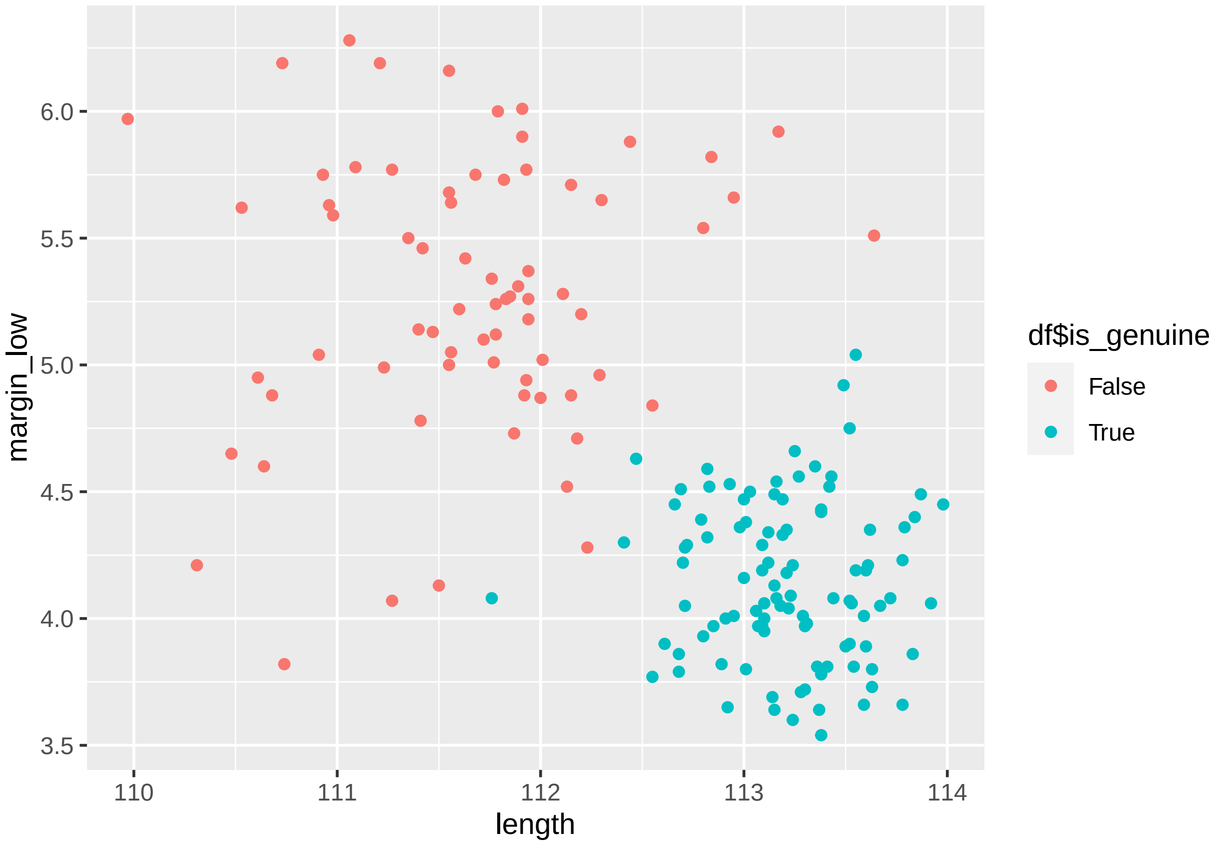



```
###contribution des individus pour chacun des axes 1,2 et 3
contrib<-res.pca$ind$contrib%>%as.data.frame()
```

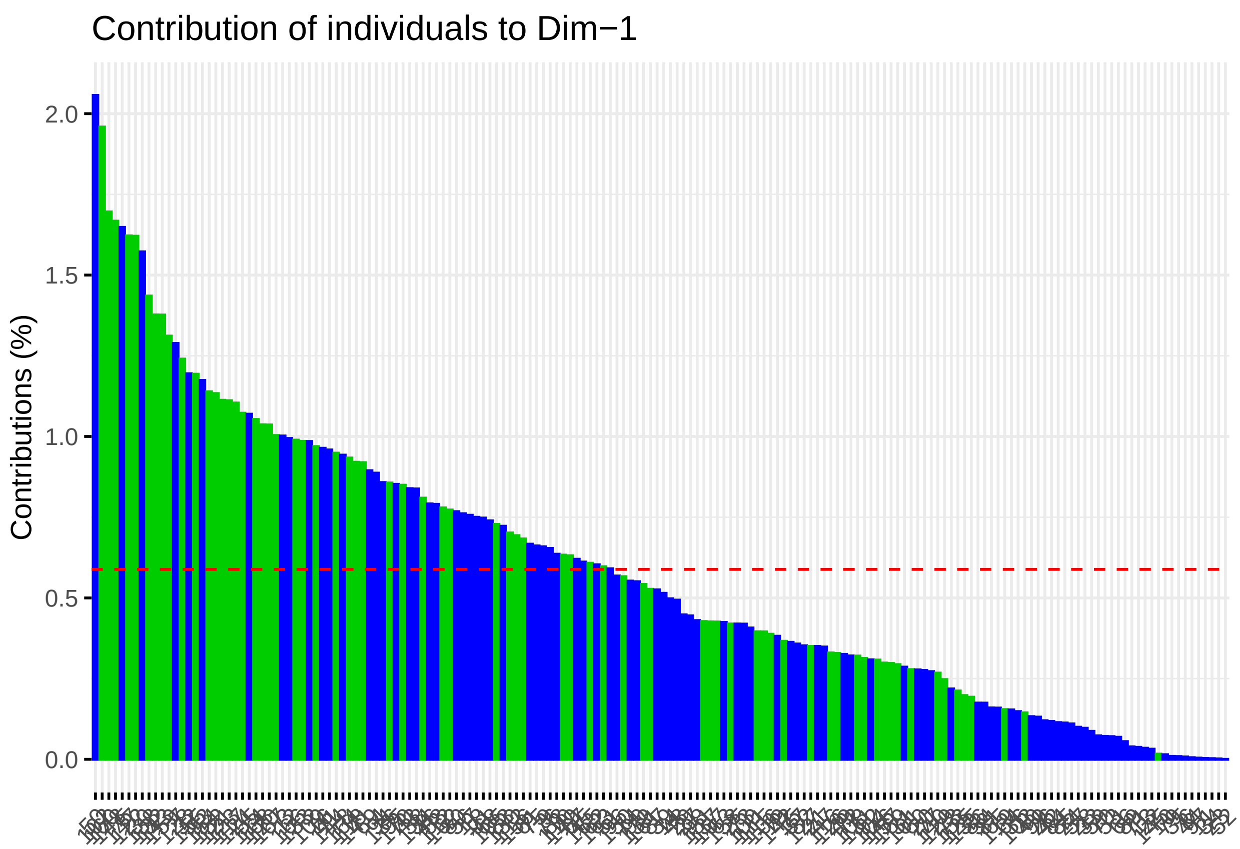
```
###contribution des variables pour chacun des axes 1,2 et 3
contrib_var<-res.pca$var$contrib%>%as.data.frame()
```

```
###J'observe mes billets dans le plan length margin_low
###Les différentes pca me font penser que ce plan devrait
###séparer de façon efficace les vraies et les faux billets
ggplot(df, aes(length, margin_low))+geom_point(aes(colour = df$is_genuine))
```

```
## Warning: Use of `df$is_genuine` is discouraged. Use `is_genuine` instead.
```

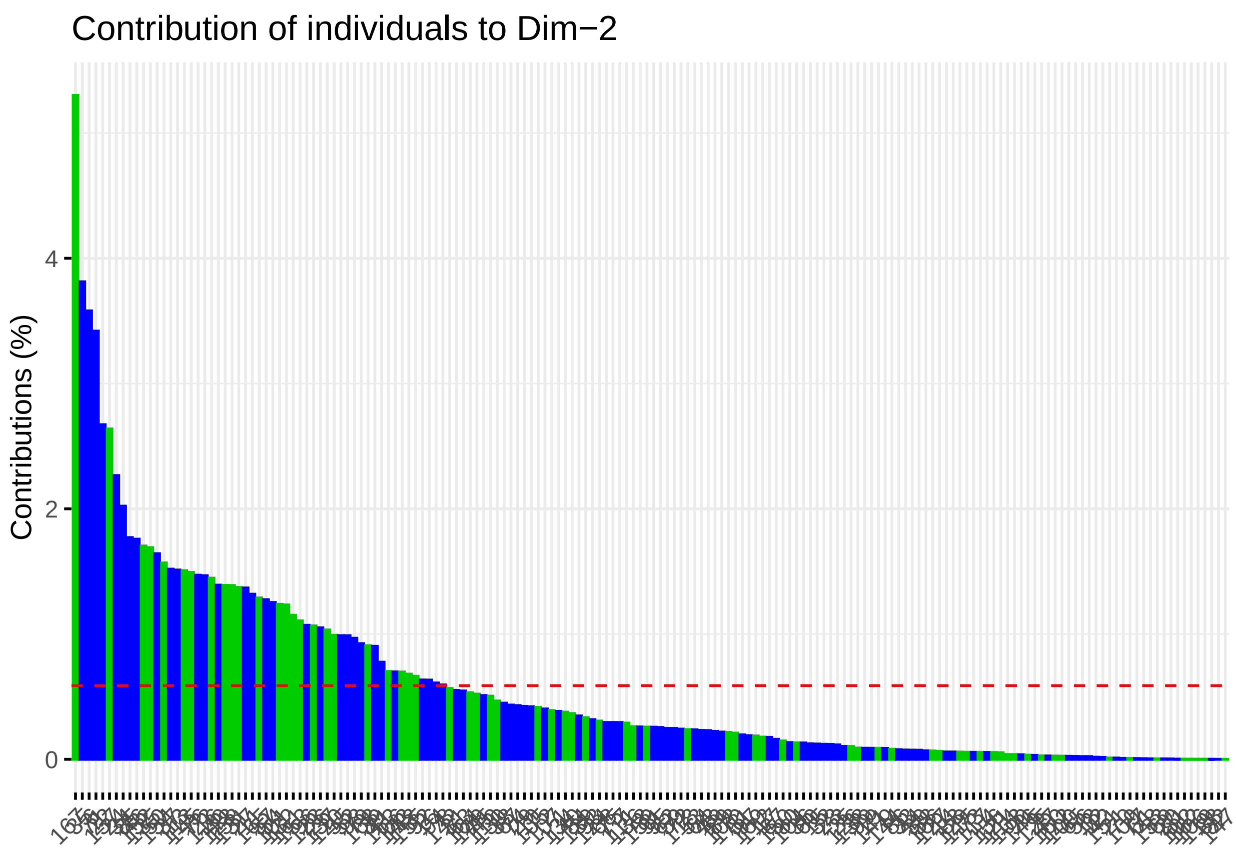



```
clr<-cbind(df,contrib)%>%select(is_genuine,Dim.1)%>%
  arrange(desc(Dim.1))

clr$is_genuine<-as.numeric(clr$is_genuine)+2

clr2<-cbind(df,contrib)%>%select(is_genuine,Dim.2)%>%
  arrange(desc(Dim.2))

clr2$is_genuine<-as.numeric(clr2$is_genuine)+2

###Je calcul la contribution des billets pour mes axe 1 et 2
### de ma pca
fviz_contrib(res.pca, choice="ind", axes = 1, top = Inf,
             fill = clr$is_genuine, color = clr$is_genuine)
```

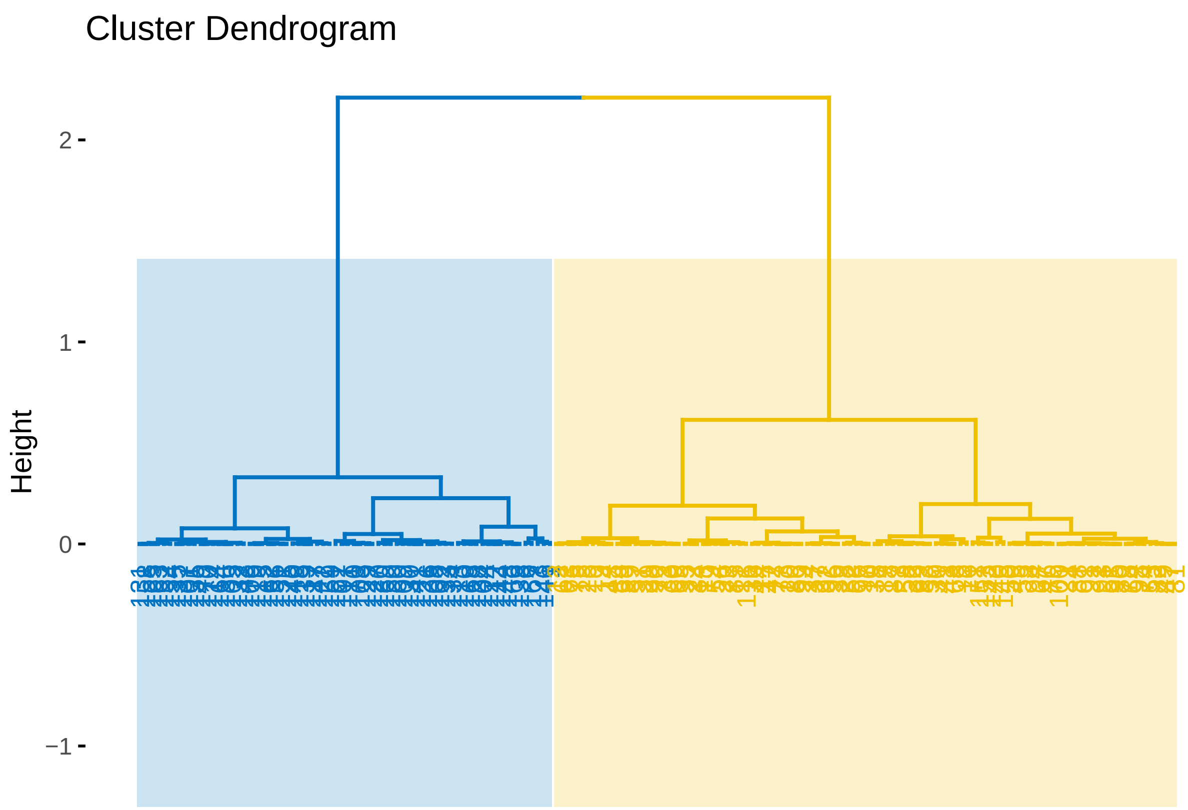



```
###False en vert, True en bleu
```

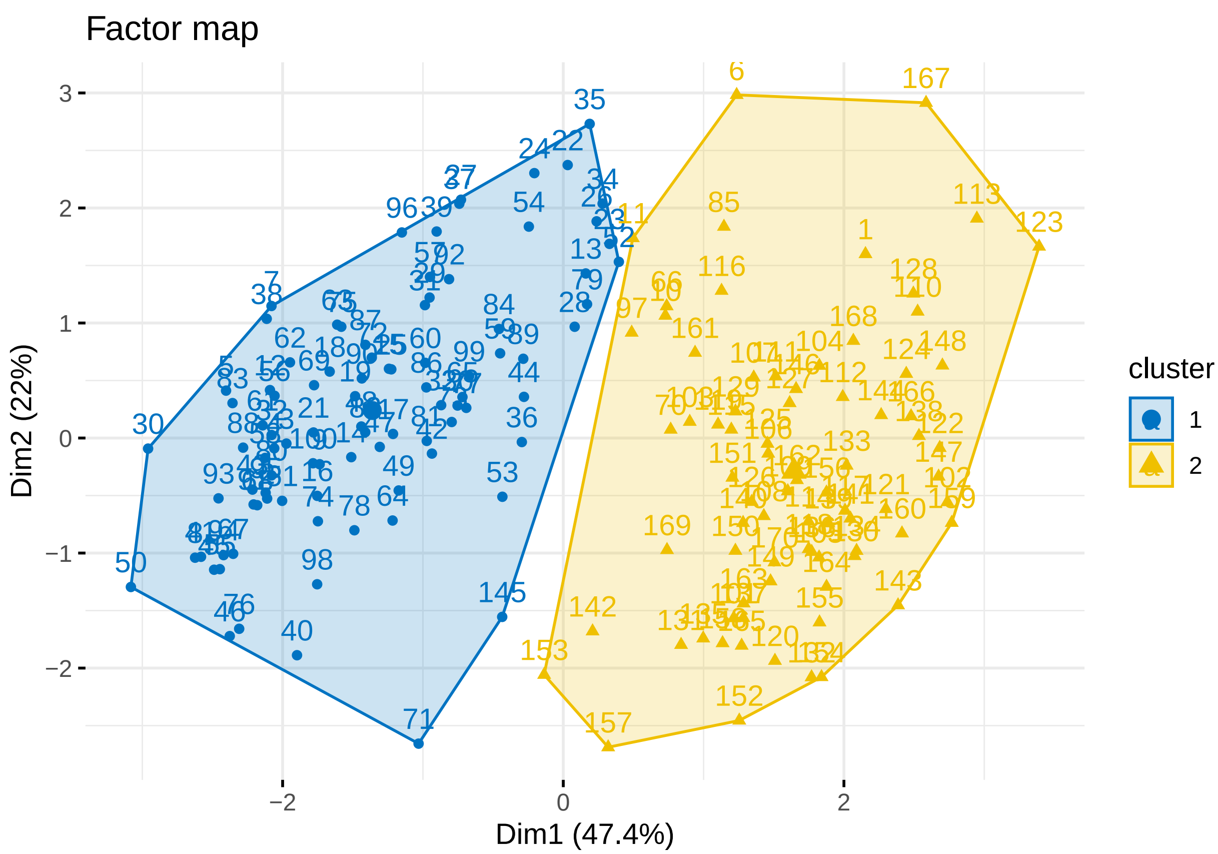
```
fviz_contrib(res.pca, choice="ind", axes = 2, top = Inf,
             fill = clr2$is_genuine, color = clr2$is_genuine)
```




```r
###J'effectue une classification non supervisé à l'aide de la fonction
###HCPC ( cah + kmeans)
res.pca <- PCA(df_cor,
               graph = FALSE, scale.unit = TRUE, ncp = 3)
res.hcpc <- HCPC(res.pca, graph = FALSE, nb.par = 200, nb.clust = 2)


fviz_dend(res.hcpc,
          k = 2,
          cex = 0.7,                     # Taille du text
          palette = "jco",               # Palette de couleur ?ggpubr::ggpar
          rect = TRUE, rect_fill = TRUE, # Rectangle autour des groupes
          rect_border = "jco",           # Couleur du rectangle
          labels_track_height = 0.8      # Augment l'espace pour le texte
          )
```



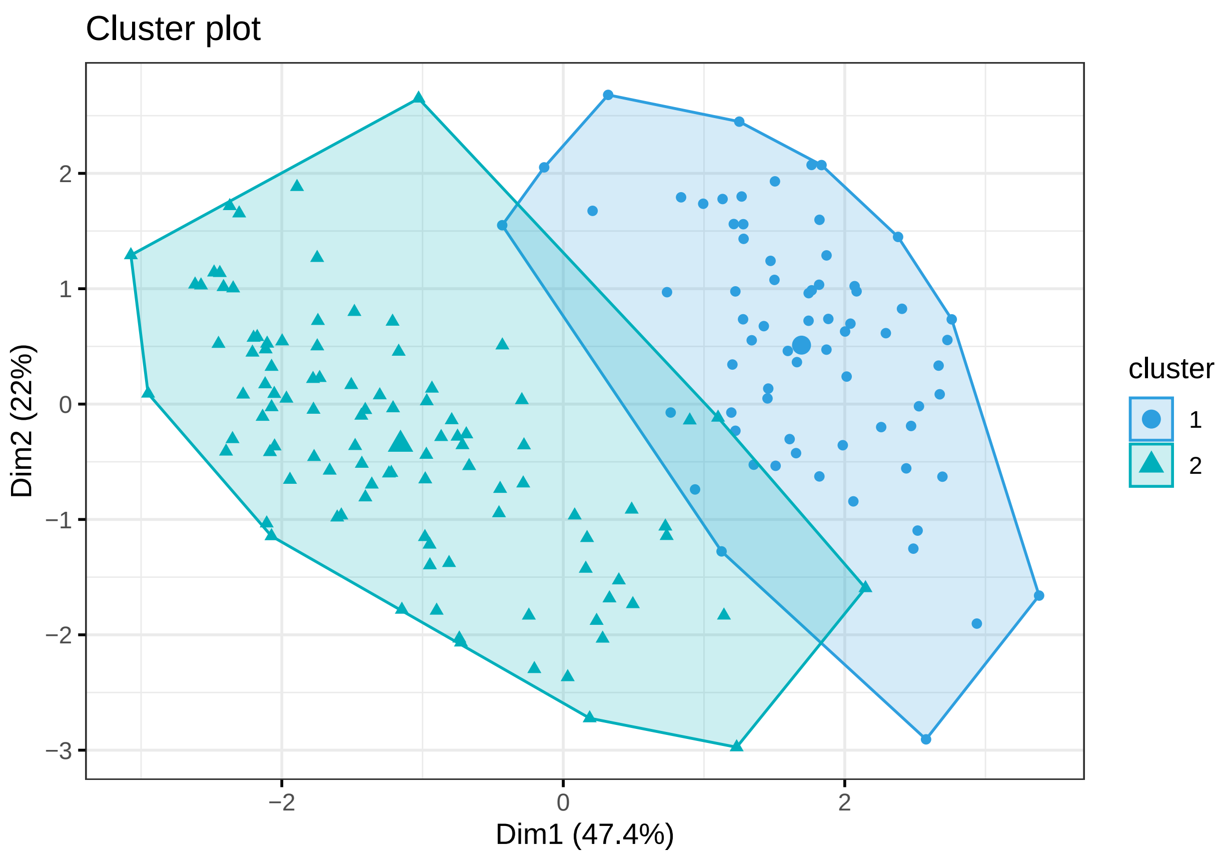

```
fviz_cluster(res.hcpc,
show.clust.cent = TRUE, # Show cluster centers
palette = "jco",          # Color palette see ?ggpubr::ggpar
ggtheme = theme_minimal(),
main = "Factor map"
)
```




```
res.hcpc$data.clust$clust<-
  res.hcpc$data.clust$clust%>%factor(labels = c("2","1"))

###Je calcul la réussite de cette classification
mean(res.hcpc$data.clust$clust == df$is_genuine%>%as.numeric())
```

```
## [1] 0.9470588
```

```
res.km<-kmeans(df_cor, 2, nstart = 25)

fviz_cluster(res.km, data = df_cor,
             palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
             )
```




```r
###Je calcul la réussite de cette classification
mean(res.km$cluster == df$is_genuine%>%as.numeric())
```

```
## [1] 0.9823529
```

```r
a<-table(res.km$cluster-1, df$is_genuine%>%as.numeric()-1)

pred<-(res.km$cluster-1)%>%as.logical()%>%as.factor()
actual<-df$is_genuine%>%as.logical()%>%as.factor()

actual[actual == TRUE]%>%length()
```

```
## [1] 100
```

```r
confusionMatrix(pred, actual, positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    68    1
##      TRUE      2   99
##
##                Accuracy : 0.9824
```

```
##                  95% CI : (0.9493, 0.9963)
##     No Information Rate : 0.5882
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9635
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9900
##             Specificity : 0.9714
##          Pos Pred Value : 0.9802
##          Neg Pred Value : 0.9855
##              Prevalence : 0.5882
##          Detection Rate : 0.5824
##    Detection Prevalence : 0.5941
##       Balanced Accuracy : 0.9807
##
##        'Positive' Class : TRUE
##
```

```
###Je souhaite faire une classification supervisé
###Je commence par utilisé un modèle de regression logistique
model <- glm(is_genuine~
               margin_low + margin_up + diagonal + height_left + height_right + length,
             family="binomial",data= df)


###Je supprime les variable qui nuisent à mon modèle
a<-MASS::stepAIC(model)
```

```
## Start:  AIC=14
## is_genuine ~ margin_low + margin_up + diagonal + height_left +
##     height_right + length
##
##                Df Deviance    AIC
## - diagonal      1    0.000 12.000
## - height_right  1    0.000 12.000
## - height_left   1    0.000 12.000
## <none>               0.000 14.000
## - margin_up     1    8.265 20.265
## - length        1   11.198 23.198
## - margin_low    1   42.342 54.342
##
## Step:  AIC=12
## is_genuine ~ margin_low + margin_up + height_left + height_right +
##     length
##
##                Df Deviance    AIC
## - height_right  1    0.000 10.000
## - height_left   1    0.000 10.000
## <none>               0.000 12.000
## - margin_up     1    8.568 18.568
## - length        1   12.462 22.462
## - margin_low    1   47.782 57.782
```

```
##
## Step:  AIC=10
## is_genuine ~ margin_low + margin_up + height_left + length
##
##                Df Deviance    AIC
## - height_left  1    0.000  8.000
## <none>              0.000 10.000
## - margin_up    1    8.585 16.585
## - length       1   12.716 20.716
## - margin_low   1   53.624 61.624
##
## Step:  AIC=8
## is_genuine ~ margin_low + margin_up + length
##
##               Df Deviance    AIC
## <none>             0.000  8.000
## - margin_up   1    8.586 14.586
## - length      1   12.721 18.721
## - margin_low  1   57.812 63.812
```

```r
###je conserve margin up et low et length
```

```r
###Je vais calculer le taux de réussite de mon modèle glm à 3 variables
###et le comparer à un modèle lda
###Je vais utiliser la validation croisé pour tester mes modèles

pred_a<-c()
pred_b<-c()
pred_c<-c()
actual<-c()

i = 1
r = data.frame(mean_a = 0, mean_b = 0, mean_c = 0)

for (i in 1 : 100) {

training.samples <- df$is_genuine %>%
  createDataPartition(p = 0.7, list = FALSE)

train.data <- df[training.samples, ]
test.data <- df[-training.samples, ]

# Estimate preprocessing parameters
preproc.param <- train.data %>%
  preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train.transformed <- preproc.param %>% predict(train.data)
test.transformed <- preproc.param %>% predict(test.data)

###reference pour la matrice de confusion
actual<-c(actual,test.transformed$is_genuine)


# Fit the model
```

```r
model1 <- lda(is_genuine~., data = train.transformed)
# Make predictions
predictions <- model1 %>% predict(test.transformed)
# Model accuracy
a<-mean(predictions$class==test.transformed$is_genuine)

# prediction du model1 pour la matrice de confusion
pred_a<-(c(pred_a, predictions$class)-1)%>%as.logical()%>%as.factor()


# Fit the model
model2 <- glm(is_genuine~ length + margin_low + margin_up,
            family="binomial",data= train.transformed)

probabilities <- model2 %>% predict(test.transformed, type = "response")

predicted.classes <- ifelse(probabilities > 0.5, "True", "False")

b<-mean(predicted.classes == test.transformed$is_genuine)

# prediction du model1 pour la matrice de confusion
pred_b<-c(pred_b%>%as.logical(),
          predicted.classes%>%as.logical())%>%as.logical()%>%as.factor()


# Fit the model
model3 <- glm(is_genuine~ .,family="binomial",data= train.transformed)

probabilities <- model3 %>% predict(test.transformed, type = "response")

predicted.classes <- ifelse(probabilities > 0.5, "True", "False")

c<-mean(predicted.classes == test.transformed$is_genuine)

# prediction du model1 pour la matrice de confusion
pred_c<-c(pred_c%>%as.logical(),
          predicted.classes%>%as.logical())%>%as.logical()%>%as.factor()


r$mean_a<-r$mean_a + a
r$mean_b<-r$mean_b + b
r$mean_c<-r$mean_c + c

i = i+1
}

r$mean_a<-r$mean_a/(i-1)
r$mean_b<-r$mean_b/(i-1)
r$mean_c<-r$mean_c/(i-1)

actual<-(actual-1)%>%as.logical()%>%as.factor()

conf1<-confusionMatrix(pred_a, actual, positive = "TRUE")
```

```r
conf2<-confusionMatrix(pred_b, actual, positive = "TRUE")
conf3<-confusionMatrix(pred_c, actual, positive = "TRUE")

conf1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2089   20
##      TRUE     11 2980
##
##                Accuracy : 0.9939
##                  95% CI : (0.9914, 0.9959)
##     No Information Rate : 0.5882
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9875
##
##  Mcnemar's Test P-Value : 0.1508
##
##             Sensitivity : 0.9933
##             Specificity : 0.9948
##          Pos Pred Value : 0.9963
##          Neg Pred Value : 0.9905
##              Prevalence : 0.5882
##          Detection Rate : 0.5843
##    Detection Prevalence : 0.5865
##       Balanced Accuracy : 0.9940
##
##        'Positive' Class : TRUE
##
```

```r
conf2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2046   28
##      TRUE     54 2972
##
##                Accuracy : 0.9839
##                  95% CI : (0.9801, 0.9872)
##     No Information Rate : 0.5882
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9667
##
##  Mcnemar's Test P-Value : 0.005766
##
##             Sensitivity : 0.9907
##             Specificity : 0.9743
```

```
##          Pos Pred Value : 0.9822
##          Neg Pred Value : 0.9865
##              Prevalence : 0.5882
##          Detection Rate : 0.5827
##    Detection Prevalence : 0.5933
##       Balanced Accuracy : 0.9825
##
##        'Positive' Class : TRUE
##
```

```
conf3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2035   76
##      TRUE     65 2924
##
##                Accuracy : 0.9724
##                  95% CI : (0.9675, 0.9767)
##     No Information Rate : 0.5882
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.943
##
##  Mcnemar's Test P-Value : 0.3997
##
##             Sensitivity : 0.9747
##             Specificity : 0.9690
##          Pos Pred Value : 0.9783
##          Neg Pred Value : 0.9640
##              Prevalence : 0.5882
##          Detection Rate : 0.5733
##    Detection Prevalence : 0.5861
##       Balanced Accuracy : 0.9719
##
##        'Positive' Class : TRUE
##
```

```
###J'affiche les résultats
###LDA : Nbre de fois ou lda est meilleur
###length_margin : nbre de fois ou glm est meilleur
###egal: égalité // mean_a réussite LDA // mean_b réussite glm
r
```

```
##      mean_a    mean_b    mean_c
## 1 0.9939216 0.9839216 0.9723529
```

```
model1
```

```
## Call:
## lda(is_genuine ~ ., data = train.transformed)
```

```
##
## Prior probabilities of groups:
##     False      True
## 0.4117647 0.5882353
##
## Group means:
##          diagonal height_left height_right margin_low  margin_up     length
## False -0.05869044   0.6270678    0.6495289  0.9285925  0.7243752 -0.9761150
## True   0.04108331  -0.4389475   -0.4546702 -0.6500148 -0.5070626  0.6832805
##
## Coefficients of linear discriminants:
##                      LD1
## diagonal     -0.07883636
## height_left   0.06650273
## height_right -0.13381362
## margin_low   -1.65636378
## margin_up    -1.07126350
## length        0.98662629
```

```
model2
```

```
##
## Call:  glm(formula = is_genuine ~ length + margin_low + margin_up, family = "binomial",
##     data = train.transformed)
##
## Coefficients:
## (Intercept)       length   margin_low    margin_up
##       72.42        59.98       -85.49       -91.86
##
## Degrees of Freedom: 118 Total (i.e. Null);  115 Residual
## Null Deviance:       161.2
## Residual Deviance: 2.281e-08     AIC: 8
```

```
model3
```

```
##
## Call:  glm(formula = is_genuine ~ ., family = "binomial", data = train.transformed)
##
## Coefficients:
##  (Intercept)      diagonal   height_left  height_right    margin_low
##        32.63        -10.93        -15.28         18.81        -56.78
##    margin_up        length
##       -59.78         27.43
##
## Degrees of Freedom: 118 Total (i.e. Null);  112 Residual
## Null Deviance:       161.2
## Residual Deviance: 8.767e-09     AIC: 14
```

```
df_t<-filter(df, is_genuine == "True")
df_f<-filter(df, is_genuine == "False")


###Je test la normalité de mes données
shapiro.test(df_t$diagonal)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_t$diagonal
## W = 0.98977, p-value = 0.6461
```

```r
shapiro.test(df_t$height_left)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_t$height_left
## W = 0.97153, p-value = 0.02896
```

```r
shapiro.test(df_t$height_right)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_t$height_right
## W = 0.97212, p-value = 0.03217
```

```r
shapiro.test(df_t$margin_low)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_t$margin_low
## W = 0.98048, p-value = 0.1448
```

```r
shapiro.test(df_t$margin_up)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_t$margin_up
## W = 0.9742, p-value = 0.04671
```

```r
shapiro.test(df_t$length)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_t$length
## W = 0.97805, p-value = 0.09361
```

```r
###Je test la normalité de mes données
shapiro.test(df_f$diagonal)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_f$diagonal
## W = 0.95618, p-value = 0.01547
```

```r
shapiro.test(df_f$height_left)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_f$height_left
## W = 0.9895, p-value = 0.8285
```

```r
shapiro.test(df_f$height_right)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_f$height_right
## W = 0.96716, p-value = 0.06299
```

```r
shapiro.test(df_f$margin_low)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_f$margin_low
## W = 0.97762, p-value = 0.2431
```

```r
shapiro.test(df_f$margin_up)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_f$margin_up
## W = 0.97558, p-value = 0.1877
```

```r
shapiro.test(df_f$length)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_f$length
## W = 0.98173, p-value = 0.3993
```

```r
###Je sauvegarde mes objets
save(model1, model2, preproc.param, file = "model")
```

```r
summary(model)
```

```
##
## Call:
## glm(formula = is_genuine ~ margin_low + margin_up + diagonal +
##     height_left + height_right + length, family = "binomial",
##     data = df)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -7.465e-05  -2.100e-08   2.100e-08   2.100e-08   7.553e-05
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4746.47 8556000.81  -0.001    1.000
## margin_low      -131.68   69995.63  -0.002    0.998
## margin_up       -217.08   77520.99  -0.003    0.998
## diagonal          15.04   69484.85   0.000    1.000
## height_left      -59.09  133557.62   0.000    1.000
## height_right      43.04   72860.71   0.001    1.000
## length            45.75   22283.83   0.002    0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.3035e+02  on 169  degrees of freedom
## Residual deviance: 2.0247e-08  on 163  degrees of freedom
## AIC: 14
##
## Number of Fisher Scoring iterations: 25
```