Attaque réseaux BHP

Jean-Baptiste AUJOGUE jean-baptiste.aujogue@etu.univ-lyon1.fr (mailto:jean-baptiste.aujogue@etu.univ-lyon1.fr), Xian YANG xian.yang@etu.univ-lyon1.fr (mailto:xian.yang@etu.univ-lyon1.fr)

20 déc. 2017

0. Introduction

Nous allons traiter un jeu de données concernant la sécurité du réseau. Il s'agit apparamment, lors d'un certain genre d'attaque du réseau, des états du réseau, de la transmission des paquets informatiques etc. Le jeu de données est téléchargeable sous ce lien (https://archive.ics.uci.edu/ml/machine-learning-databases/00404/OBS-Network-DataSet_2_Aug27.arff) et la description des variables se trouve ici (https://archive.ics.uci.edu/ml/datasets/Burst+Header+Packet+%28BHP%29+flooding+attack+on+Optical+Burst+Switching+%28OBS%29+Network
Notre but principal dans cet article est de faire une analyse en composantes principales sur le jeu de données. Néanmoins, on verra que les données sont assez brutes qui nécessiteront un pré-traitement avant de faire l'ACP.

L'article est organisé come suit :
Dans les parties I et II on importe les données et traite les valeurs manquantes. Dans la partie III on étudie la colinéarité deux à deux des variables et discute la garde ou non des couples de variables extrèmement corrélés. On argumente pourquoi dans notre analyse nous optons pour l'enlève des variables redondantes. La partie IV est une analyse complète des données en composantes principales, qui étudie et illustre entre autres la variance expliquée par les dimensions principales, les relations entres les variables numériques, la qualité de représentation et la contribution des variables par rapport aux dimensions principales. Cette partie finit par une analyse factorielle des données mixtes sans détails mathématiques pour révéler les relations entre les variables quantitatives et les variables qualitatives.

I. Importation du jeu de données

Le package `foreign` nous permet de lire le jeu de données au format `.arff`.

```
library(foreign)
```

```
setwd("C:/Users/ellie/Documents/R/OpticalBurstSwitching")
OBS <- read.arff("OBS-Network-DataSet_2_Aug27.arff")
# head(OBS)
```

Il nous convient de connaître la taille du jeu de données et d'en faire une statistique descriptive.

```
dim(OBS)
```

```
## [1] 1075   22
```

```
summary(OBS)
```

```
##      Node       Utilised Bandwith Rate Packet Drop Rate  Full_Bandwidth
##  Min.   :3.000   Min.   :0.2356         Min.   :0.08613   Min.   : 100.0
##  1st Qu.:3.000   1st Qu.:0.4469         1st Qu.:0.24754   1st Qu.: 300.0
##  Median :9.000   Median :0.5772         Median :0.43799   Median : 500.0
##  Mean   :6.014   Mean   :0.5979         Mean   :0.41136   Mean   : 540.5
##  3rd Qu.:9.000   3rd Qu.:0.7645         3rd Qu.:0.55658   3rd Qu.: 800.0
##  Max.   :9.000   Max.   :0.9280         Max.   :0.76794   Max.   :1000.0
##
##  Average_Delay_Time_Per_Sec Percentage_Of_Lost_Pcaket_Rate
##  Min.   :0.0004060          Min.   : 8.61
##  1st Qu.:0.0004510          1st Qu.:24.75
##  Median :0.0006110          Median :43.80
##  Mean   :0.0009619          Mean   :41.16
##  3rd Qu.:0.0009530          3rd Qu.:56.67
##  Max.   :0.0052370          Max.   :76.79
##
##  Percentage_Of_Lost_Byte_Rate Packet Received  Rate of Used_Bandwidth
##  Min.   : 8.613               Min.   :0.2321        Min.   : 27.55
##  1st Qu.:24.754               1st Qu.:0.4333        1st Qu.:138.41
##  Median :43.799               Median :0.5620        Median :291.59
##  Mean   :41.192               Mean   :0.5881        Mean   :340.78
##  3rd Qu.:56.672               3rd Qu.:0.7525        3rd Qu.:515.18
##  Max.   :76.794               Max.   :0.9139        Max.   :867.04
##
##  Lost_Bandwidth   Packet Size_Byte Packet_Transmitted Packet_Received
##  Min.   : 34.16   Min.   :1440     Min.   : 9048      Min.   : 2451
##  1st Qu.: 81.20   1st Qu.:1440     1st Qu.:27092      1st Qu.:12491
##  Median :159.51   Median :1440     Median :45188      Median :26847
##  Mean   :199.68   Mean   :1440     Mean   :48826      Mean   :30593
##  3rd Qu.:279.27   3rd Qu.:1440     3rd Qu.:72228      3rd Qu.:46588
##  Max.   :687.93   Max.   :1440     Max.   :90324      Max.   :77131
##
##   Packet_lost    Transmitted_Byte    Received_Byte
##  Min.   : 3913   Min.   : 13029120   Min.   :  3529440
##  1st Qu.: 7984   1st Qu.: 39012480   1st Qu.: 17736480
##  Median :14944   Median : 65070720   Median : 37357920
##  Mean   :18590   Mean   : 70308837   Mean   : 49873429
##  3rd Qu.:25962   3rd Qu.:104008320   3rd Qu.: 67086720
##  Max.   :62415   Max.   :130066560   Max.   :980066560
##  NA's   :15
##  10-Run-AVG-Drop-Rate 10-Run-AVG-Bandwith-Use  10-Run-Delay
##  Min.   :0.05875      Min.   :0.2074          Min.   :0.0004050
##  1st Qu.:0.18993      1st Qu.:0.3799          1st Qu.:0.0006560
##  Median :0.30702      Median :0.5089          Median :0.0007650
##  Mean   :0.30760      Mean   :0.5532          Mean   :0.0009336
##  3rd Qu.:0.40600      3rd Qu.:0.7341          3rd Qu.:0.0009840
##  Max.   :0.63371      Max.   :0.8909          Max.   :0.0049040
##
##  Node Status  Flood Status             Class
##  B   :475    Min.   :0.00000   Block       :120
##  NB  :285    1st Qu.:0.02305   NB-No Block:500
##  P NB:315    Median :0.07933   NB-Wait     :300
##              Mean   :0.13194   No Block    :155
##              3rd Qu.:0.23054
##              Max.   :0.56674
##
```

On voit que la variable `Packet Size_Byte` est complètement inutile car elle ne change jamais (=1440). On va la virer.

```
OBS <- OBS[names(OBS) != "Packet Size_Byte"]
```

Pour rendre les noms de variables faciles à traiter par `R`, on va remplacer des espaces par des points.

```
names(OBS) <- make.names(names(OBS), unique = TRUE)
```

Maintenant le jeu de données est prêt pour `R`.

II. Traitement des valeurs manquantes

On demande d'abord un aperçu des valeurs manquantes dans le jeu de données pour décider quel traitement prendre par la suite.

```
valMan <- which(is.na(OBS), arr.ind = TRUE, useNames = TRUE)
dim(valMan)
```

```
## [1] 15  2
```

Il s'avère que dans ce jeu de données de taille 1075 * 21, il n'y a que 15 valeurs manquantes. On n'a donc pas besoin d'en faire une visualisation et le traitement sera relativement libre et simple. La fonction suivante peut illustrer à l'aide d'un tableau la structure des valeurs manquantes.

```
library(mice)
mdp <- md.pattern(OBS)
mdp
```

```
##      Node Utilised.Bandwith.Rate Packet.Drop.Rate Full_Bandwidth
## 1060    1                      1                1              1
##   15    1                      1                1              1
##         0                      0                0              0
##      Average_Delay_Time_Per_Sec Percentage_Of_Lost_Pcaket_Rate
## 1060                          1                              1
##   15                          1                              1
##                               0                              0
##      Percentage_Of_Lost_Byte_Rate Packet.Received..Rate of.Used_Bandwidth
## 1060                            1                     1                  1
##   15                            1                     1                  1
##                                 0                     0                  0
##      Lost_Bandwidth Packet_Transmitted Packet_Received Transmitted_Byte
## 1060              1                  1               1                1
##   15              1                  1               1                1
##                   0                  0               0                0
##      Received_Byte X10.Run.AVG.Drop.Rate X10.Run.AVG.Bandwith.Use
## 1060             1                     1                        1
##   15             1                     1                        1
##                  0                     0                        0
##      X10.Run.Delay Node.Status Flood.Status Class Packet_lost
## 1060             1           1            1     1           1  0
##   15             1           1            1     1           0  1
##                  0           0            0     0          15 15
```

On voit que toutes les 15 valeurs manquantes appartiennent à la variable `Packet_lost`. Comme elles sont peu par rapport au nombre d'observation, on pourrait bien entendu les ignorer toutes. Mais ici, on applique une technique d'imputation qui porte le nom k plus proches voisins, remplaçant la valeur manquante par la moyenne des observations voisinée. Le nombre des voisins sera fixé avec le reste des données.

```
library(DMwR)
```

```
## Loading required package: lattice
```

```
## Loading required package: grid
```

```
OBScomplet <- knnImputation(OBS[! names(OBS) %in% c("Node.Status", "Class")])
# Quelles sont les valeurs imputées ?
OBScomplet[valMan]
```

```
##  [1] 20480.07 12191.96 11870.21 20480.07 12191.96 11870.21 20480.07
##  [8] 12191.96 11870.21 20480.07 12191.96 11870.21 20480.07 12191.96
## [15] 11870.21
```

III. Analyse des corrélations deux à deux

Dans une analyse de données brutes il peut être intéressant de regarder d'abord s'il y a des corrélations élevées deux à deux parmi les variables numériques, afin de détecter celles qui sont susceptibles d'être redondantes. Une carte thermique (heatmmap) peut faciliter la visualisation.

```
library(plotly)
```

```
## Loading required package: ggplot2
```

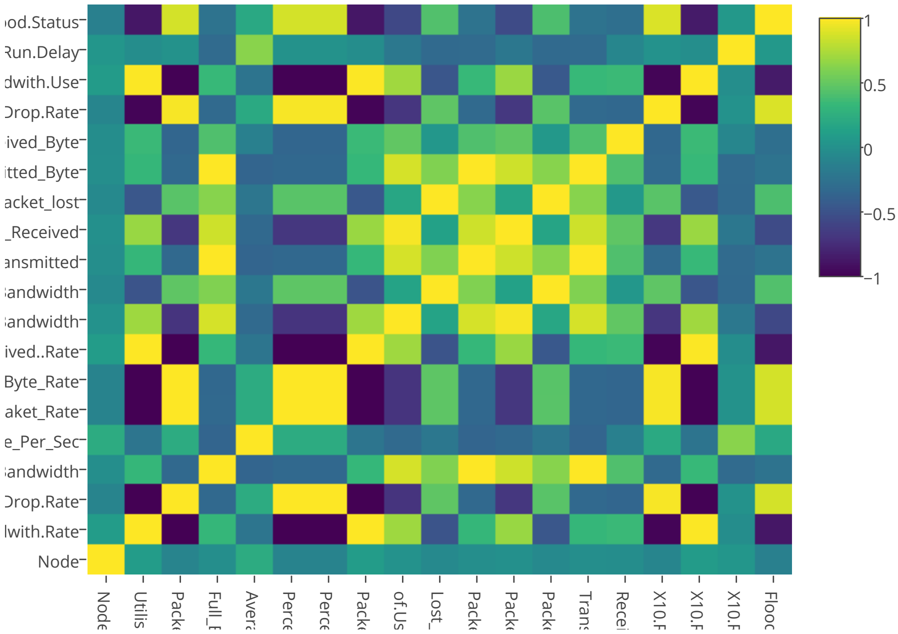
```
##
## Attaching package: 'plotly'
```

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

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

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

```
corOBS <- cor(OBScomplet[! names(OBScomplet) %in% c("Node.Status", "Class")])
corHmp <- plot_ly(x = names(OBScomplet[! names(OBScomplet) %in% c("Node.Status", "Class")]),
             y = names(OBScomplet[! names(OBScomplet) %in% c("Node.Status", "Class")]), z = corOBS, type = "heatmap")
corHmp
```



In the correlation heatmap we can repeatly spot extremely high pairwise correlation close or even equal to 1 or -1.

Highly to perfectly correlated variables could bias the PCA result in a way that PCA will overemphasize the common contribution of the (nearly) redundant variables. Therefore, it might make sense to find out and remove such variables before doing a PCA, **Especially when they describe actually (nearly) the same aspect of an issue.** For instance, `Tansmitted_Byte` and `Packet_Transmitted` have a correlation of 1, because they reflect the same quantity up to a ratio `Packet Size_Byte` , which, as being mentioned here above, never changes. Thus, we will remove one of both. Which one to remove is of our free choice. Here, we consider that quantitative variables in packets may be more reader friendly than those in bytes in terms of unit, so that we keep the variable `Packet_Transmitted` and drop the other one. But what about the high correlation between other variables (e.g. higher than 97% but not perfectly equal to 1) ? Shall we remove the nearly redundant variables too before doing our PCA?

The statistical community doesn't have a straightforward anwer to it. As a matter of fact, it hugely depends on the nature of data and the purpose to do the PCA. On one hand, like we said, highly correlated variables would be possible to strongly influence the result of the PCA and, as a result, the real contributions to the principal components of the underlying variables that are truely meaningful. If our PCA is meant to give such information, then high correlation should better be avoided prior to the PCA. On the other hand, however, a PCA with redundant variables can still faithfully reveal the high correlation between them, though principal components would be probably established otherwise. In that sense, if it is an exploratory PCA that we are doing, which only aims to find a broad outline of the relationships between variables disregarding how principal components are built, then including some redundant ones may be fine.

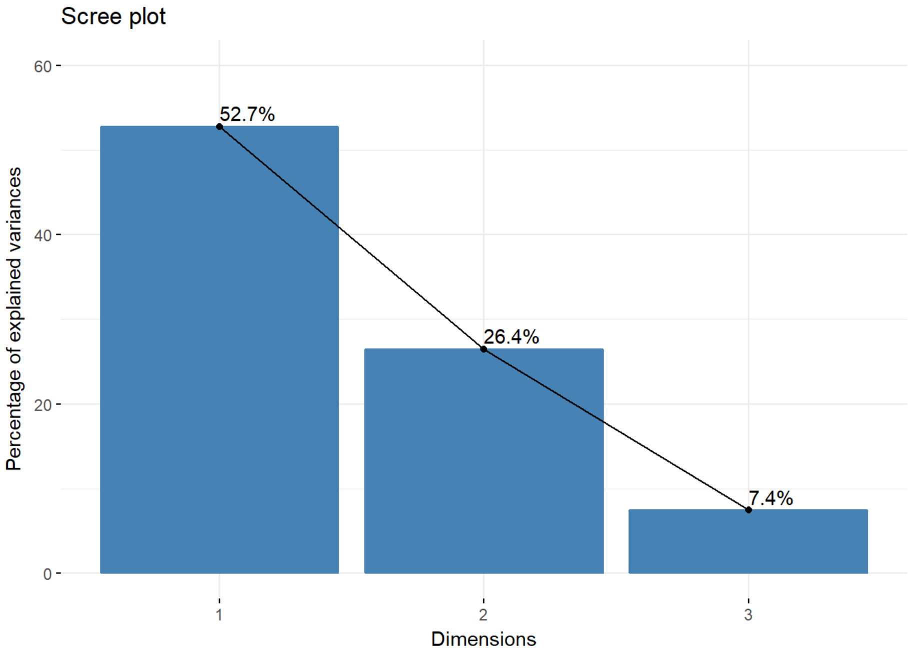
In the light of this, in the next section we decide to go an onerous but careful way, in which we do firstly a PCA with almost all the variables. With both the correlation circle of PCA and the correlation heatmap, we then kick out the redundant variables and do a second PCA with only variables that we consider to be meaningful. Further analysis (variable and individual relationship, quantitative and qualitative variable relationship) shall also be based on the latter PCA.

More technically, in a rough and rapid preprocessing procedure, we could limit our focus on high pairwise correlations (the reader should know that in a more rigorous treatment, high correlations within a multuple tuple of variables should also be considered) and refer to the heatmap of variable correlations. We may set a threshold, e.g. 0.97, and find out all couples whose absolute value of correlation exceeds this threshold, before we remove one of both variables by verifying that they are indeed telling (nearly) the same story.

IV. Analyse en composantes principales

1. Analyse des variables numériques

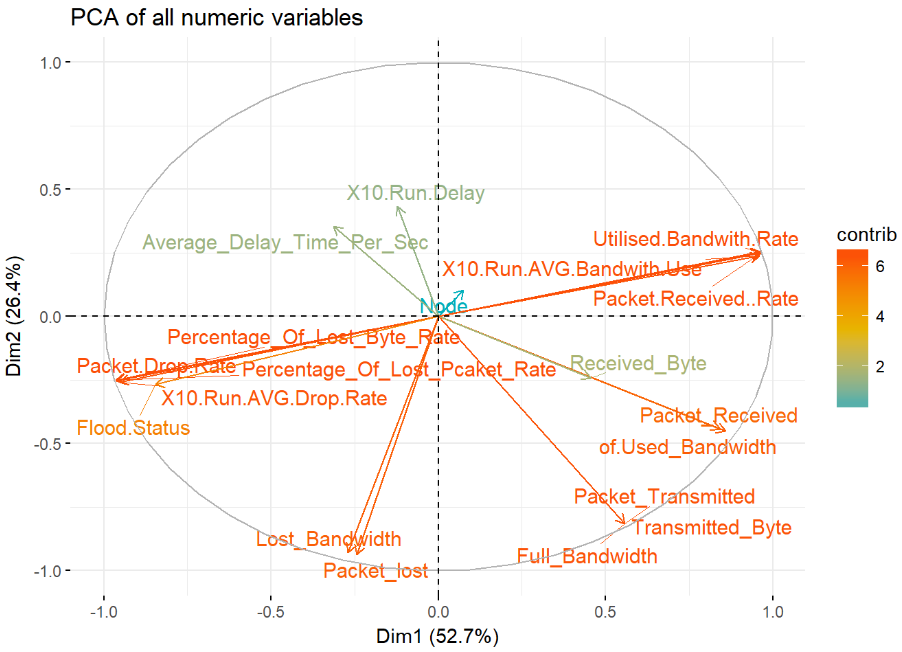
```
library("factoextra")
pcaOBS <- prcomp(OBScomplet[! names(OBScomplet) %in% c("Node.Status", "Class")], scale. = TRUE, rank. = 3)
fviz_eig(pcaOBS, ncp = 3, addlabels = TRUE, ylim = c(0, 60))
```




```
summary(pcaOBS)$importance[3, c(1, 2)]
```

```
##     PC1     PC2
## 0.52732 0.79172
```

```
fviz_pca_var(pcaOBS, col.var = "contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE, title =
             "PCA of all numeric variables")
```



Les trois composantes principales expliquent 87% de la variance totale alors que les deux premières 79%, ce qui est aussi acceptable.

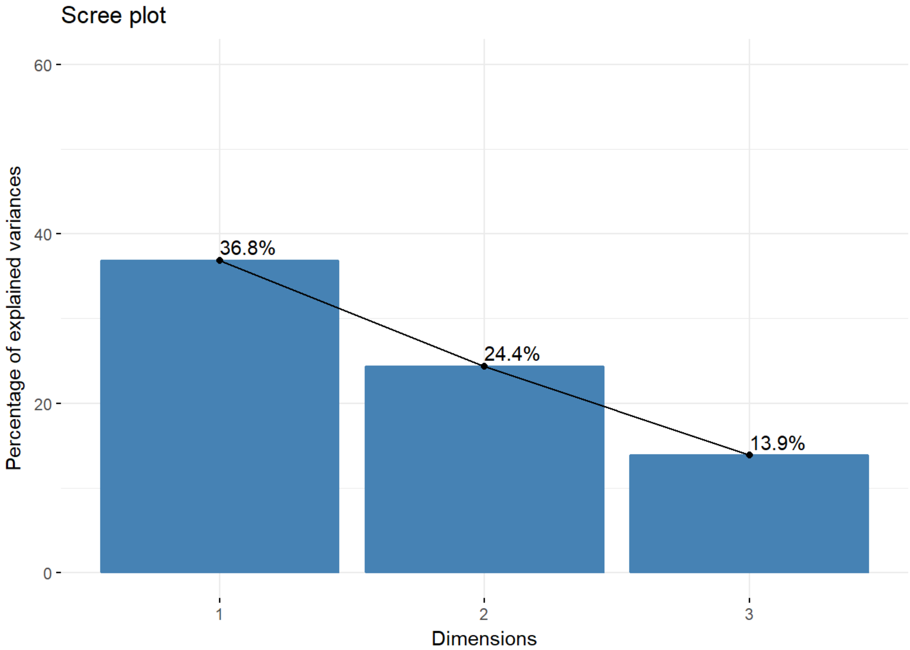
We see that in the correlation circle there are four subgroups of very closely situated variables which are potentially redundant variables (visibly there are five, but the one at the very left is highly negatively correlated with the one at the very right). For example, `Tansmitted_Byte`, `Packet_Transmitted` and `Full_Bandwidth` are even perfectly correlated, which can be also confirmed by the previous correlation heatmap. No doubt those subgroups consist of the *most contributing variables (the longest arrows that touch almost the circle)*, because they are redundant! This may be due to the fact that raw data are collected by non-statisticians and a primary treatment of unusual variables is missing. Including them in the PCA might not be of interest.

We now decide to remove extremely high correlations among variables (greater than 97%) for our next PCA so that only one variable of each of the four subgroups should stay in the game. Another reason for doing this, in a point of view of field knowledge, is that all the variables within the same subgroup mean in fact the same thing just in some different way. At the end, we choose to keep `Packet.Received..Rate`, `Packet_Received`, `Packet_Transmitted` and `Packet_Lost` as representatives of their belonging subgroup along with other well defined variables. In the more balanced coming PCA with uniquely the non redundant variables, we could spot a change in variance contribution of the variables, as well as a diminuation of explained variance by the principal components because data are now distributed in a more balanced way.

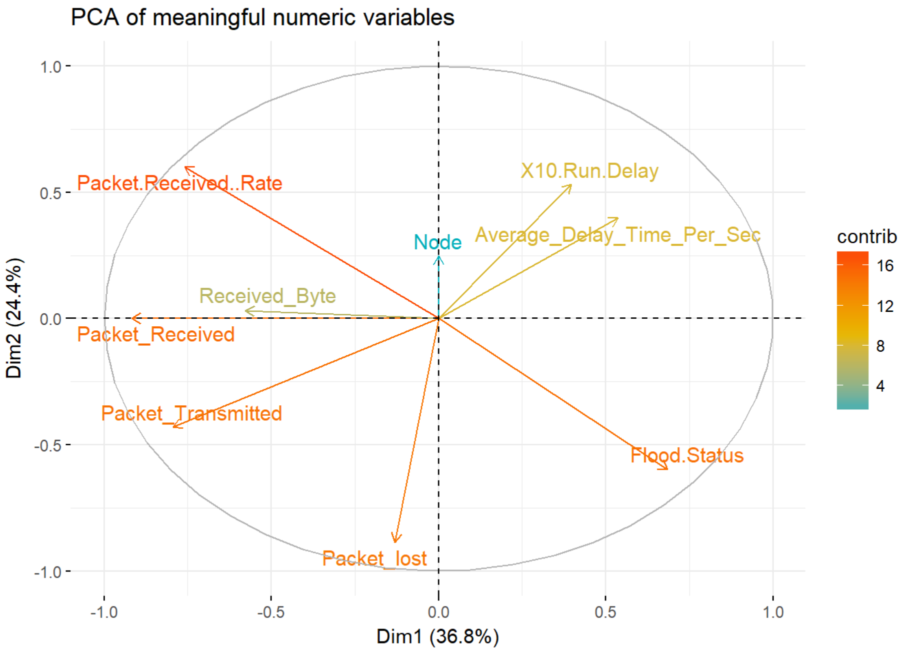
```
pca0BS2 <- prcomp(OBScomplet[c("Packet.Received..Rate", "Packet_Received", "Packet_Transmitted", "Packet_lost",
                              "Average_Delay_Time_Per_Sec", "X10.Run.Delay", "Node", "Received_Byte", "Flood.Status")], scale. = TRUE, ra
nk. = 3)
summary(pca0BS2)$importance[3, c(1, 2, 3)]
```

```
##     PC1     PC2     PC3
## 0.36840 0.61203 0.75056
```

```
fviz_eig(pca0BS2, ncp = 3, addlabels = TRUE, ylim = c(0, 60))
```




```
fviz_pca_var(pca0BS2, col.var = "contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE, title =
             "PCA of meaningful numeric variables")
```



Les trois composantes principales expliquent 75% de la variance totale alors que les deux premières seulement 61%. Un graphe d'ACP avec les 2 premères dimensions est donc pas très représentatif, qui peut révéler néanmoins quand-même des pistes.

Following information can be extracted from the correlation circle of the PCA graph above:

`Flood.Status`, `Packet_lost`, `Packet_Transmitted`, `Packet_Received` and `Packet.Received..Rate` are the five most well represented or in other words most contributing variables to the first both principal dimensions. `Packet_Received` has no contribution to the first dimension while `Node` has no contribution to the second one. `Received_Byte` has little contribution to the first dimension while `Packe_lost` has litte contribution to the second one. `X10.Run.Delay` and `Average_Delay_Time_Per_Sec` seem to be correlated in the projected dataset onto the first two dimensions. `Packet.Received..Rate` and `Flood.Status` are highly negatively correlated.

To extract all information about the remaining variables, we can do

```
variables <- get_pca_var(pcaOBS2)
```

What are the scores of them?

```
variables$coord[, 1:3]
```

```
##                                 Dim.1         Dim.2      Dim.3
## Packet.Received..Rate      -0.7598709053  0.600722370 -0.1033570
## Packet_Received            -0.9172582980  0.001244397  0.2268809
## Packet_Transmitted         -0.7944946606 -0.430210811  0.3630639
## Packet_lost                -0.1315211178 -0.886093952  0.3516166
## Average_Delay_Time_Per_Sec  0.5366453995  0.398986327  0.6148888
## X10.Run.Delay               0.3971744367  0.532528682  0.5054290
## Node                        0.0003522424  0.248589468  0.4157637
## Received_Byte              -0.5772036435  0.030542401  0.3098019
## Flood.Status                0.6851099789 -0.596702315  0.1637975
```

We may want to illustrate them in a `corrplot` :

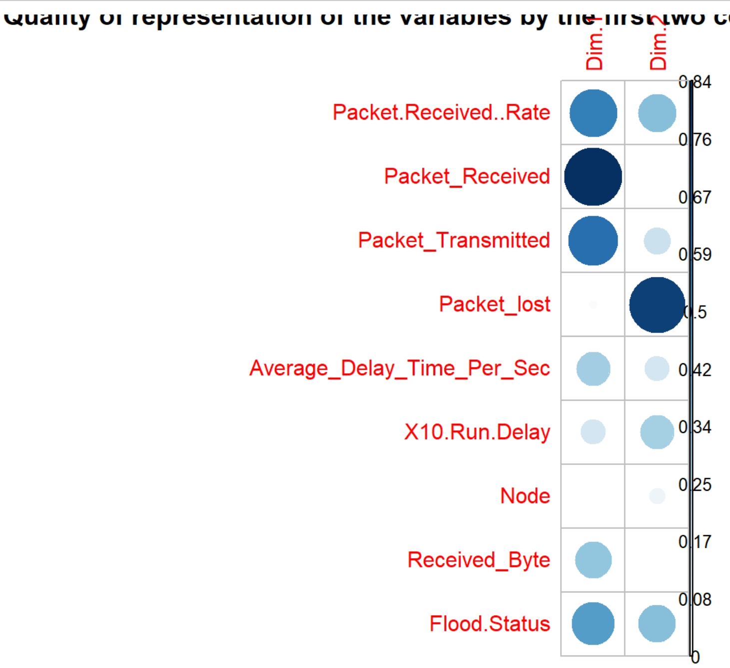
```
variables$cos2[, 1:2]
```

```
##                                Dim.1        Dim.2
## Packet.Received..Rate      5.774038e-01 3.608674e-01
## Packet_Received            8.413628e-01 1.548523e-06
## Packet_Transmitted         6.312218e-01 1.850813e-01
## Packet_lost                1.729780e-02 7.851625e-01
## Average_Delay_Time_Per_Sec 2.879883e-01 1.591901e-01
## X10.Run.Delay              1.577475e-01 2.835868e-01
## Node                       1.240747e-07 6.179672e-02
## Received_Byte              3.331640e-01 9.328383e-04
## Flood.Status               4.693757e-01 3.560537e-01
```

```
library("corrplot")
```

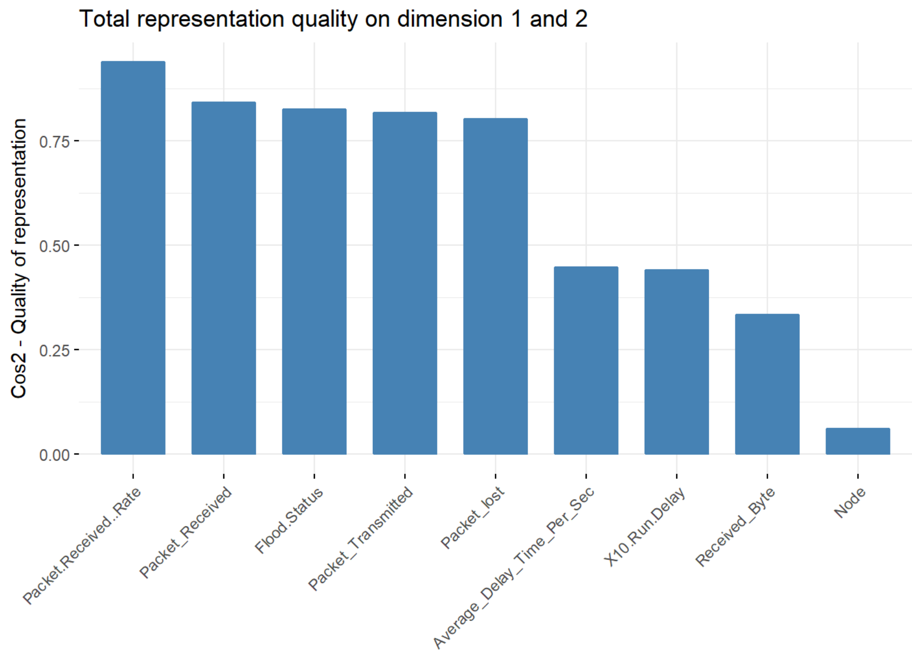
```
## corrplot 0.84 loaded
```

```
corrplot(variables$cos2[, 1:2], is.corr=FALSE, title = "Quality of representation of the variables by the first two components")
```



Or sum it up:

```
fviz_cos2(pcaOBS2, choice = "var", axes = 1:2, title = "Total representation quality on dimension 1 and 2")
```

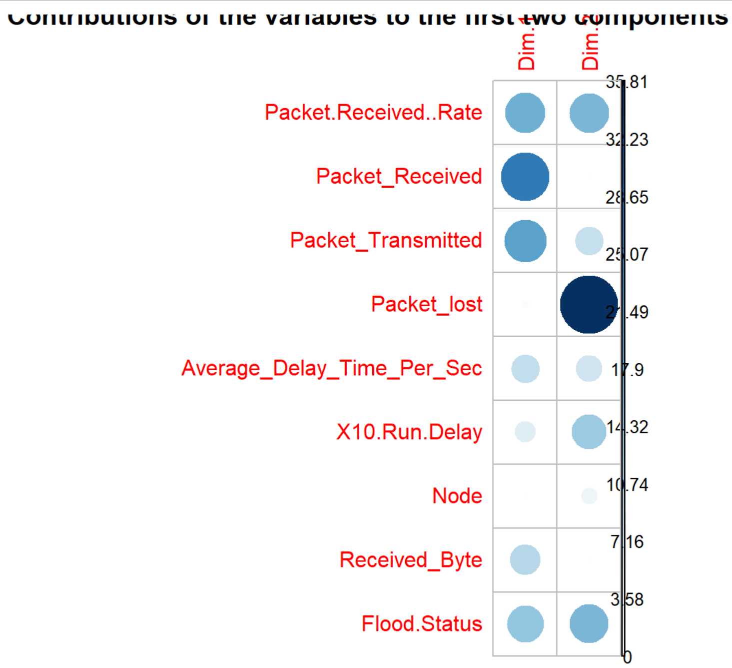


How are the contributions of the variables to the first two components?

```
variables$contrib[, 1:2]
```

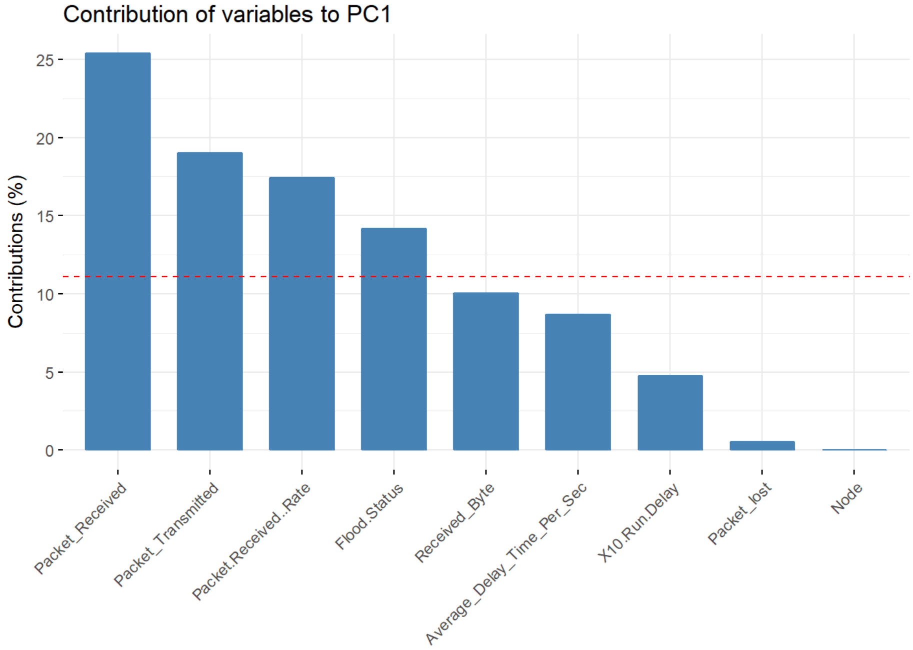
```
##                                Dim.1        Dim.2
## Packet.Received..Rate      1.741496e+01 1.645788e+01
## Packet_Received            2.537618e+01 7.062263e-05
## Packet_Transmitted         1.903815e+01 8.440901e+00
## Packet_lost                5.217156e-01 3.580847e+01
## Average_Delay_Time_Per_Sec 8.685957e+00 7.260093e+00
## X10.Run.Delay              4.757792e+00 1.293338e+01
## Node                       3.742192e-06 2.818328e+00
## Received_Byte              1.004849e+01 4.254343e-02
## Flood.Status               1.415675e+01 1.623834e+01
```

```
corrplot(variables$contrib[, 1:2], is.corr=FALSE, title = "Contributions of the variables to the first two components")
```

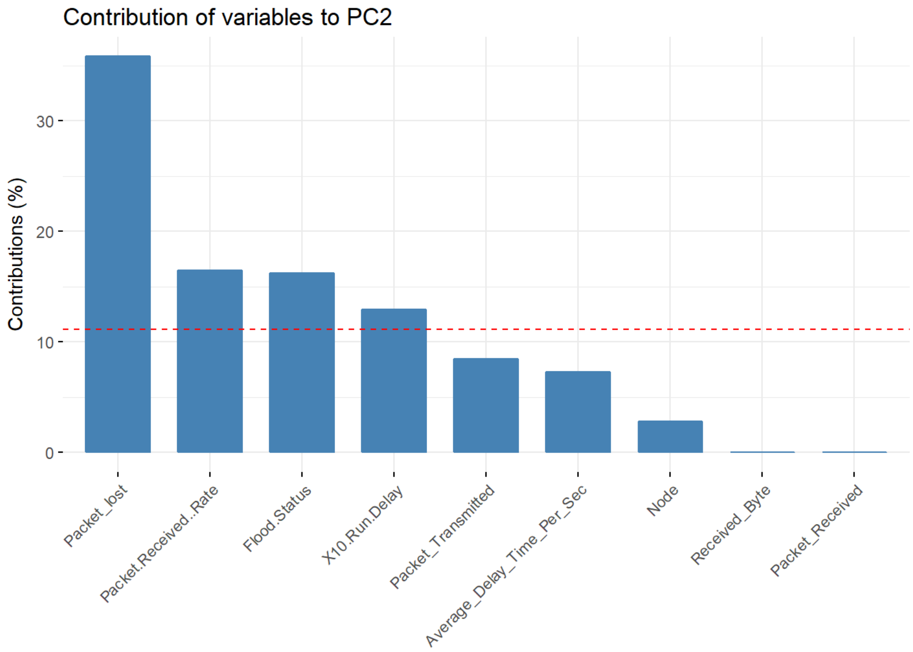


Or let's present them ranked:

```
fviz_contrib(pcaOBS2, choice = "var", axes = 1, title = "Contribution of variables to PC1")
```




```
fviz_contrib(pcaOBS2, choice = "var", axes = 2, title = "Contribution of variables to PC2")
```

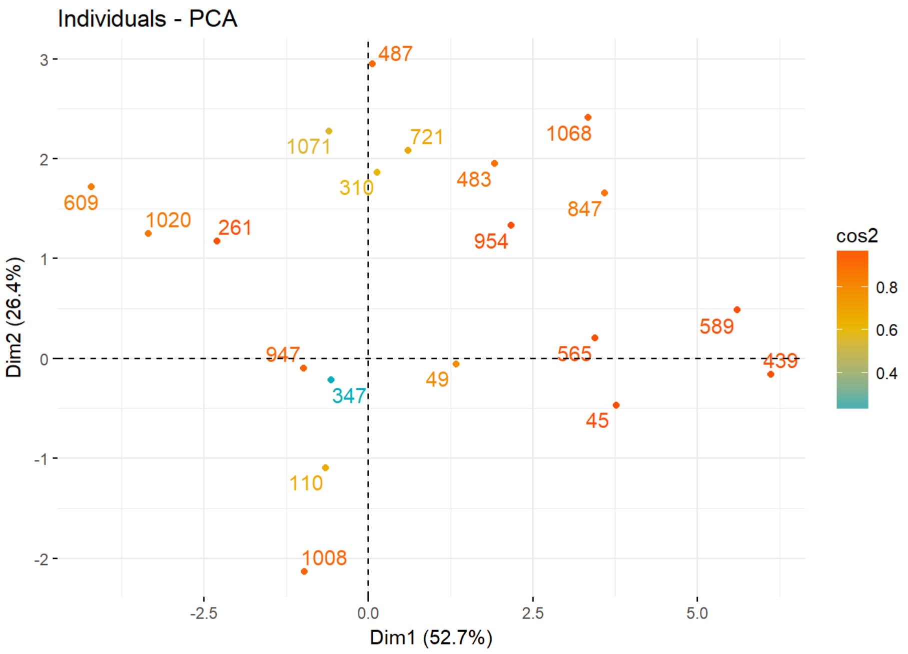


2. Analyse des individus

Now we would like to plot some individuals. We can see that here, using redundant variables in the PCA or not will lead to a considerable difference.

Firstly let's see a graph of 20 randomly selected individuals drawn in the PCA including redundant variables, coloured by their quality of representation.

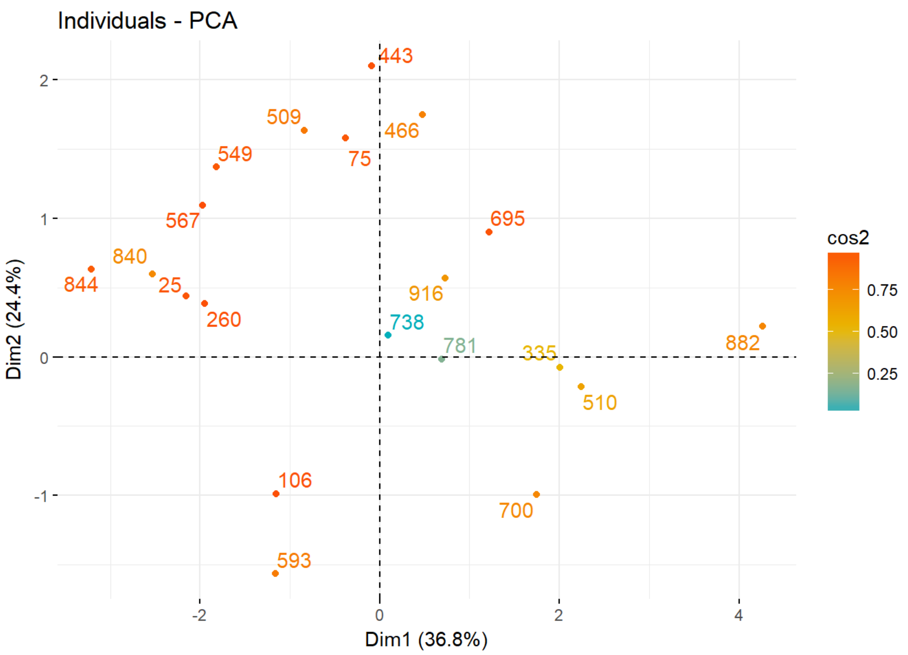
```
fviz_pca_ind(pcaOBS, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE,
             select.ind = list(name = sample.int(n = nrow(OBS), size = 20)))
```



Their quality of representation is better than the graph below. This is due to the fact that the first dimension is biased on the redundant information, so that an individual can be easily well represented by only these variables because they repeatly occur. Furthermore, we can see that the farer away the induvidual is from the origin, the better it is represented.

Then we show a graph of 20 randomly selected individuals drawn in the PCA *without* redundant variables, coloured by their quality of representation

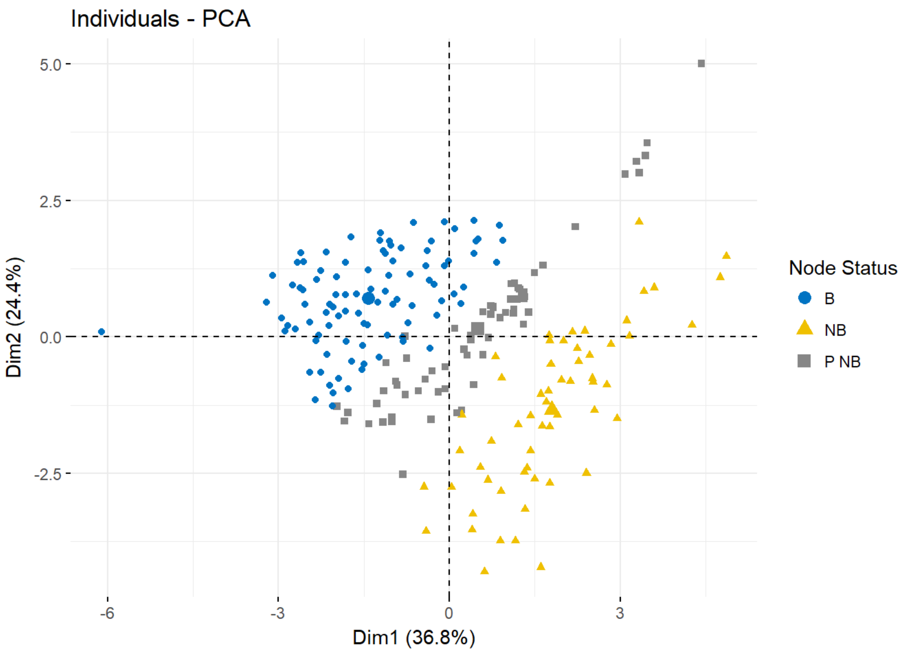
```
fviz_pca_ind(pcaOBS2, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE,
             select.ind = list(name = sample.int(n = nrow(OBS), size = 20)))
```



All kinds of quality can be observed this time, which is more meaningful than in the previous case. Hence, this PCA pragh is what we are going to keep using in the rest of our analysis.

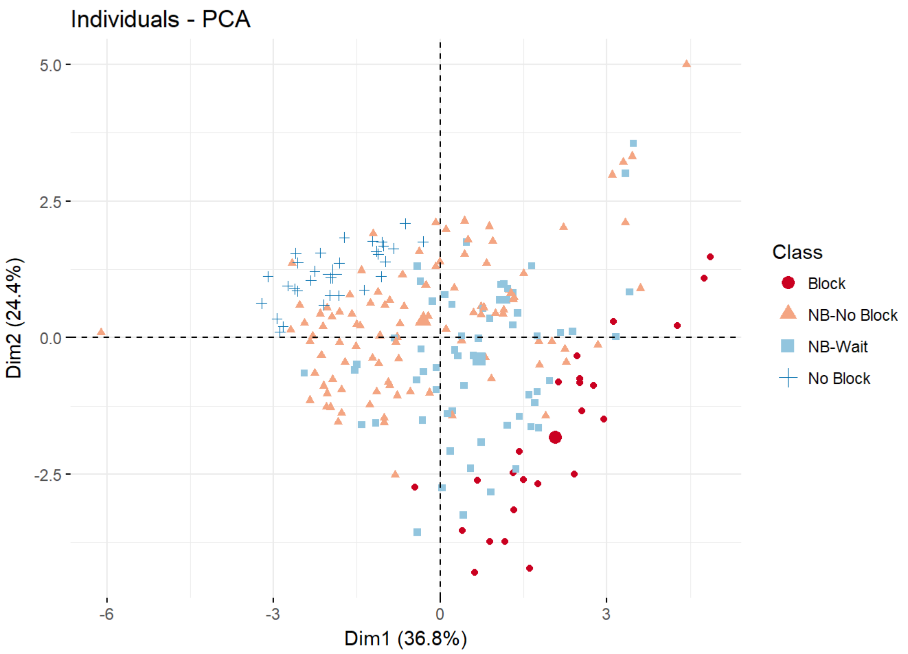
We now try to include the categorical information: `Node.Status` and `Class` in our visualisation. Firstly a graph of some randomly selected individuals coloured by subgroups of `Node.Status`

```
fviz_pca_ind(pcaOBS2, geom = "point", habillage = OBS$Node.Status,  palette = "jco",
             addEllipses = FALSE, legend.title = "Node Status")
```



Then a graph of randomly selected individuals coloured by subgroups of `Node.Status`

```
fviz_pca_ind(pcaOBS2, geom = "point", habillage = OBS$Class,  palette = "RdBu",
           select.ind = list(name = 1:1075), legend.title = "Class")
```



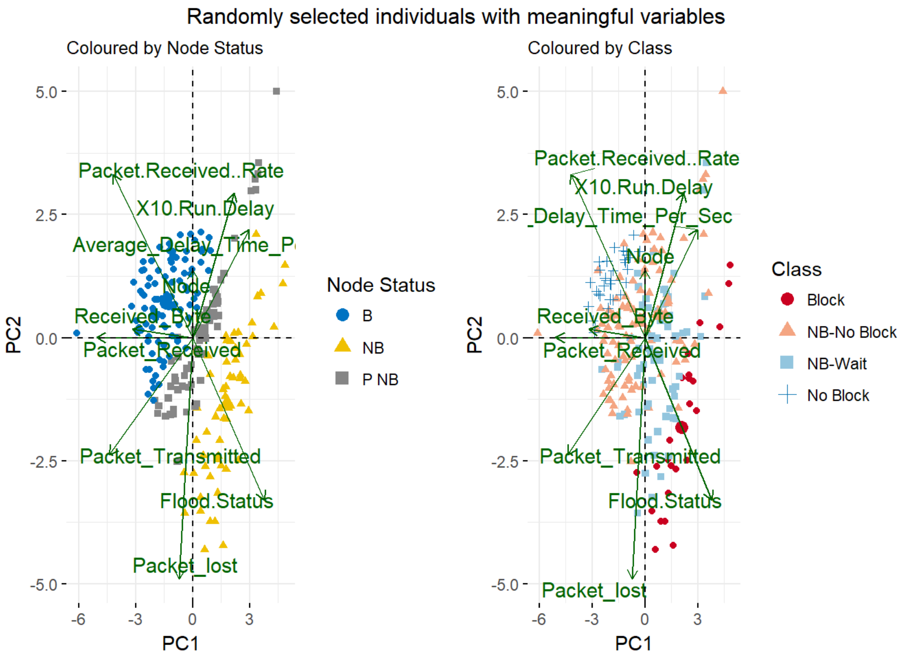
On peut voir que les centres des sous-groupes dans les deux cas s'allongent dans la même direction, digonalement dans le plan engendré par les premières 2 dimensions. Ce qui signifie que peut-être les deux classifications sont liées, ou ont une cause commune etc.

3. Présentation graphique intégrée des variables et des individus

Now we draw some biplots with randomly selected individuals and variables

```
library(gridExtra)
plot1 <- fviz_pca_biplot(pcaOBS2,
                habillage = OBS$Node.Status, palette = "jco",
                addEllipses = FALSE, label = "var",
                col.var = "darkgreen", repel = TRUE, title = NULL,
                legend.title = "Node Status",
                subtitle = "Coloured by Node Status", xlab = "PC1", ylab = "PC2")

plot2 <- fviz_pca_biplot(pcaOBS2,
                         habillage = OBS$Class, palette = "RdBu",
                         addEllipses = FALSE, label = "var",
                         col.var = "darkgreen", repel = TRUE,
                         legend.title = "Class", title = NULL,
                         subtitle = "Coloured by Class", xlab = "PC1", ylab = "PC2")
grid.arrange(plot1, plot2, ncol=2, top = "Randomly selected individuals with meaningful variables")
```



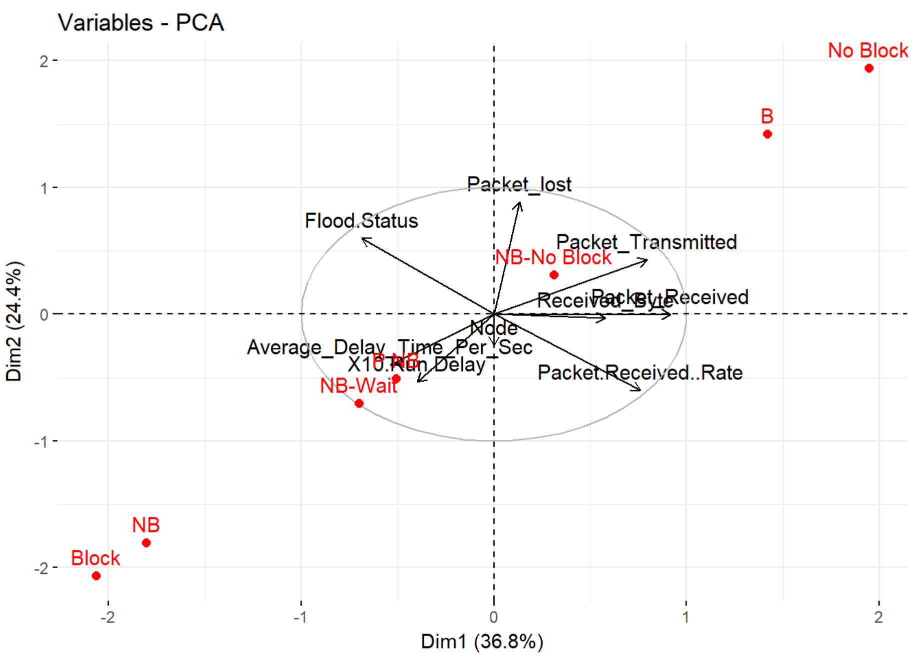
Ici on voit plus clairement : les deux classifications sont liées avec les variables `package received rate` et `flood status` , ainsi que quelques autres de leur groupe qui viennent d'être annulées à cause de la forte corrélation entre elles. En revanche, `package transmitted` ou `time per second` n'ont pas de lien, au moins dans ces deux dimensions, avec les variables catégoriques.

4. Analyse Factorielle des Données Mixtes

Finalement, on veut faire l'AFDM pour résumer les ralations entre les variables quantitatives et les variables qualitatives, dans un seul graphe de façon plus intuitive. On vient de visualiser ces relations dans le paragraphe précédent mais à travers la coloration des individus.

```
library(FactoMineR)
```

```
pcaOBS3 <- PCA(cbind(OBScomplet[c("Packet.Received..Rate", "Packet_Received", "Packet_Transmitted", "Packet_lost",
                          "Average_Delay_Time_Per_Sec", "X10.Run.Delay", "Node", "Received_Byte", "Flood.Status")],
                          OBS[c("Node.Status", "Class")]), quali.sup = 10:11, graph = FALSE)
p <- fviz_pca_var(pcaOBS3)
fviz_add(p, pcaOBS3$quali.sup$coord, color = "red")
```

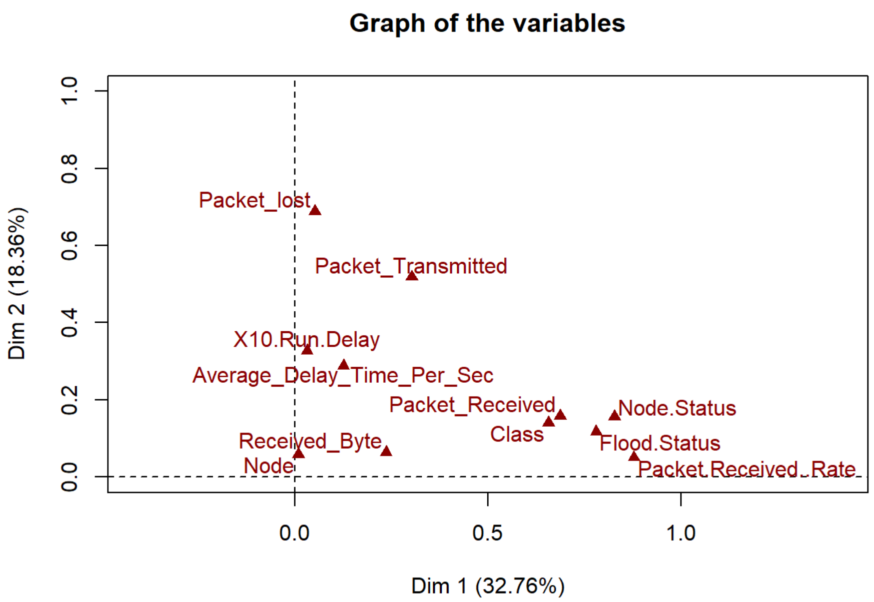



```
famdOBS <- FAMD(cbind(OBScomplet[c("Packet.Received..Rate", "Packet_Received", "Packet_Transmitted", "Packet_lost",
                          "Average_Delay_Time_Per_Sec", "X10.Run.Delay", "Node", "Received_Byte", "Flood.Status")],
                 OBS[c("Node.Status", "Class")]), graph = FALSE)

cumsum(famdOBS$eig[, 2])
```

```
## [1] 32.76325 51.12368 62.28384 71.62478 78.49738
```

```
plot(famdOBS, choix = "var")
```



Dans ce dernier graphe, il est clair que dans le plan engendré par les deux dimensions principales, `package received` et `flood status` sont bien liés avec les variables qualitatives `node status` et `class` .