Attaque réseaux BHP

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

20 déc. 2017

Table des matières


```
set.seed(11715490)
```

Pour lire fichiers en format .arff

```
## install.packages("foreign")
library(foreign)

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

```
##   Node Utilised Bandwith Rate Packet Drop Rate Full_Bandwidth
## 1    3               0.822038         0.190381           1000
## 2    9               0.275513         0.729111            100
## 3    3               0.923707         0.090383            900
## 4    9               0.368775         0.637710            100
## 5    3               0.905217         0.108670            800
## 6    9               0.514687         0.494142            100
##   Average_Delay_Time_Per_Sec Percentage_Of_Lost_Pcaket_Rate
## 1                   0.004815                      19.031487
## 2                   0.004815                      72.889036
## 3                   0.000633                       9.035834
## 4                   0.000552                      63.737843
## 5                   0.000497                      10.864208
## 6                   0.003098                      49.392131
##   Percentage_Of_Lost_Byte_Rate Packet Received  Rate of Used_Bandwidth
## 1                    19.038129             0.809619         822.03750
## 2                    72.911141             0.270889          27.55125
## 3                     9.038339             0.909617         831.33600
## 4                    63.770999             0.362290          36.87750
## 5                    10.866977             0.891330         724.17375
## 6                    49.414235             0.505858          51.46875
##   Lost_Bandwidth Packet Size_Byte Packet_Transmitted Packet_Received
## 1      177.96250             1440              90324           73128
## 2       72.44875             1440               9048            2451
## 3       68.66400             1440              81276           73930
## 4       63.12250             1440               9048            3278
## 5       75.82625             1440              72228           64379
```

```
## 6       48.53125             1440               9048            4577
##   Packet_lost Transmitted_Byte Received_Byte 10-Run-AVG-Drop-Rate
## 1       17196        130066560     105304320             0.146594
## 2        6598         13029120       3529440             0.517669
## 3        7346        117037440     106459200             0.058749
## 4        5770         13029120       4720320             0.522922
## 5        7849        104008320      92705760             0.076069
## 6        4471         13029120       6590880             0.405197
##   10-Run-AVG-Bandwith-Use 10-Run-Delay Node Status Flood Status
## 1                0.780936     0.001838           B     0.023455
## 2                0.242451     0.002236          NB     0.460725
## 3                0.886758     0.001751           B     0.000000
## 4                0.324522     0.001776          NB     0.439255
## 5                0.869009     0.001767           B     0.000000
## 6                0.442631     0.002250          NB     0.291742
##         Class
## 1 NB-No Block
## 2       Block
## 3    No Block
## 4       Block
## 5    No Block
## 6 NB-No Block
```

```
dim(OBS)
```

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

```
summary(OBS) # On voit que la variable `Packet Size_Byte` est complètement inutile car elle ne change ja
```

```
##       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
##
```

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

```
##  [1] "Node"                           "Utilised Bandwith Rate"
##  [3] "Packet Drop Rate"               "Full_Bandwidth"
##  [5] "Average_Delay_Time_Per_Sec"     "Percentage_Of_Lost_Pcaket_Rate"
##  [7] "Percentage_Of_Lost_Byte_Rate"   "Packet Received  Rate"
##  [9] "of Used_Bandwidth"              "Lost_Bandwidth"
## [11] "Packet_Transmitted"             "Packet_Received"
## [13] "Packet_lost"                    "Transmitted_Byte"
## [15] "Received_Byte"                  "10-Run-AVG-Drop-Rate"
## [17] "10-Run-AVG-Bandwith-Use"        "10-Run-Delay"
## [19] "Node Status"                    "Flood Status"
## [21] "Class"
```

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

Traitement des valeurs manquantes

```
valMan <- which(is.na(OBS), arr.ind = TRUE, useNames = TRUE)
dim(valMan) # Only 15 * 2 so we don't have to draw it.
```

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

```
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
```

```
library(ipred)
# preproc <- preProcess(OBS, method = "bagImpute")

# mice(OBS, m=5, maxit=50, meth='pmm', seed=500)
# OBScomplet <- mice(OBS)
# aggr(mdp, prop = FALSE, numbers = TRUE)

# install.packages("DMwR")
library(DMwR)
```

```
## Warning: package 'DMwR' was built under R version 3.4.3
```

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

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

```
OBScomplet <- knnImputation(OBS[! names(OBS) %in% c("Node.Status", "Class")])
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
```

```
head(OBScomplet)
```

```
##   Node Utilised.Bandwith.Rate Packet.Drop.Rate Full_Bandwidth
## 1    3               0.822038         0.190381           1000
## 2    9               0.275513         0.729111            100
## 3    3               0.923707         0.090383            900
## 4    9               0.368775         0.637710            100
## 5    3               0.905217         0.108670            800
## 6    9               0.514687         0.494142            100
##   Average_Delay_Time_Per_Sec Percentage_Of_Lost_Pcaket_Rate
## 1                   0.004815                       19.031487
## 2                   0.004815                       72.889036
## 3                   0.000633                        9.035834
## 4                   0.000552                       63.737843
## 5                   0.000497                       10.864208
## 6                   0.003098                       49.392131
##   Percentage_Of_Lost_Byte_Rate Packet.Received..Rate of.Used_Bandwidth
## 1                    19.038129              0.809619         822.03750
## 2                    72.911141              0.270889          27.55125
## 3                     9.038339              0.909617         831.33600
## 4                    63.770999              0.362290          36.87750
## 5                    10.866977              0.891330         724.17375
## 6                    49.414235              0.505858          51.46875
##   Lost_Bandwidth Packet_Transmitted Packet_Received Packet_lost
## 1      177.96250              90324           73128       17196
## 2       72.44875               9048            2451        6598
## 3       68.66400              81276           73930        7346
## 4       63.12250               9048            3278        5770
## 5       75.82625              72228           64379        7849
## 6       48.53125               9048            4577        4471
##   Transmitted_Byte Received_Byte X10.Run.AVG.Drop.Rate
## 1        130066560     105304320              0.146594
## 2         13029120       3529440              0.517669
## 3        117037440     106459200              0.058749
## 4         13029120       4720320              0.522922
## 5        104008320      92705760              0.076069
## 6         13029120       6590880              0.405197
##   X10.Run.AVG.Bandwith.Use X10.Run.Delay Flood.Status
## 1                 0.780936      0.001838     0.023455
## 2                 0.242451      0.002236     0.460725
## 3                 0.886758      0.001751     0.000000
## 4                 0.324522      0.001776     0.439255
## 5                 0.869009      0.001767     0.000000
## 6                 0.442631      0.002250     0.291742
```

```
# install.packages("Amelia")
# library(Amelia)
# missmap(OBS)
```

Méthode ACP

```
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
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%)? 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, 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. 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.

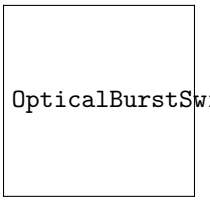
```
library("factoextra")
pcaOBS <- prcomp(OBScomplet[! names(OBScomplet) %in% c("Node.Status", "Class")], scale. = TRUE, rank. =
# fviz_eig(pcaOBS)
summary(pcaOBS)$importance[3, c(1, 2)]
```

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

```
fviz_pca_var(pcaOBS, col.var = "contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TR
```

```
## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:

## Warning in var.loadings * comp.sdev:
```

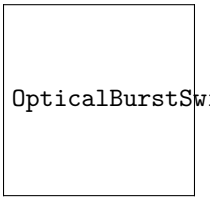
```
## Warning in var.loadings * comp.sdev:
```



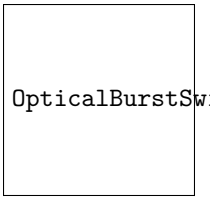

```
## We see that in the correlation circle there are four subgroups of very closely situated variables whi
## redundant variables (visibly there are five, but the one at the very left is highly negatively correl
## very right). For example, Tansmitted_Byte`, `Packet_Transmitted` and `Full_Bandwidth` are even perfe
## which can be also confirmed by the previous correlation heatmap. No doubt those subgroups consist of
## variables, because they are redundant! We now decide to remove extremely high correlation among varia
## 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, i
## 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 choos
## `Packet.Received..Rate`, `Packet_Received`, `Packet_Transmitted` and `Packet_Lost` as representatives
## other well defined variables. In the more balanced coming PCA with uniquely the non redundqnt variabl
## a change in variance contributions.

pcaOBS2 <- prcomp(OBScomplet[c("Packet.Received..Rate", "Packet_Received", "Packet_Transmitted", "Packe
                             "Average_Delay_Time_Per_Sec", "X10.Run.Delay", "Node", "Received_Byte", "F
summary(pcaOBS2)$importance[3, c(1, 2, 3)]
```

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

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




```
fviz_pca_var(pcaOBS2, col.var = "contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = 
```




```
## Following information can be extracted from the correlation circle of PCA above:
##   `Flood.Status`, `Packet_lost`, `Packet_Transmitted`, `Packet_Received` and `Packet.Received..Rate`
## 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 
##    `Received_Byte` has little contribution to the first dimension while `Packe_lost` has litte contri
##    `X10.Run.Delay` and Àverage_Delay_Time_Per_Sec` seem to be correlated in the projected dataset on
##    `Packet.Received..Rate` and `Flood.Status` are highly negatively correlated.
```

```
## To extract all information about the variables, we can do
variables <- get_pca_var(pcaOBS2)
## What are the coordinates of the variables selected?
# dim(variables$coord) # 9, 9
# head(variables$coord)
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
```

```
## What is the quality of representation of the variables by the first two components?
variables$cos2[, 1:2]
```

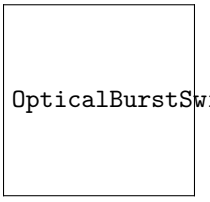
```
##                                 Dim.1        Dim.2
## Packet.Received..Rate      5.774038e-01 3.608674e-01
## Packet_Received            8.413628e-01 1.548523e-06
## Packet_Transmitted         6.312218e-01 1.850813e-01
## Packet_lost                1.729780e-02 7.851625e-01
## Average_Delay_Time_Per_Sec 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")
```

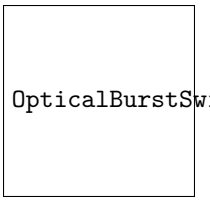
```
## Warning: package 'corrplot' was built under R version 3.4.3
```

```
## corrplot 0.84 loaded
```

```
corrplot(variables$cos2[, 1:2], is.corr=FALSE)
```




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

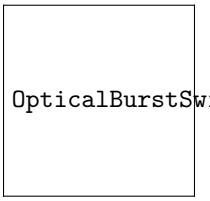


OpticalBurstSwitching_files/figure-latex/unnamed-chunk-5-5.pdf

```
## How the the contribution of the variables to the first two components?
variables$contrib[, 1:2]
```

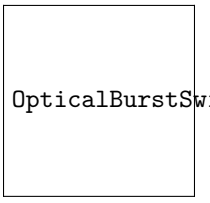
```
##                                   Dim.1        Dim.2
## Packet.Received..Rate      1.741496e+01 1.645788e+01
## Packet_Received            2.537618e+01 7.062263e-05
## Packet_Transmitted         1.903815e+01 8.440901e+00
## Packet_lost                5.217156e-01 3.580847e+01
## Average_Delay_Time_Per_Sec 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)
```



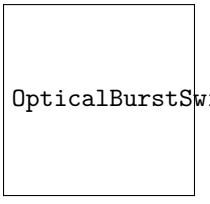
OpticalBurstSwitching_files/figure-latex/unnamed-chunk-5-6.pdf

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



OpticalBurstSwitching_files/figure-latex/unnamed-chunk-5-7.pdf

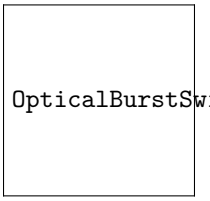
```
## Contribution of variables to PC2
fviz_contrib(pcaOBS2, choice = "var", axes = 2)
```



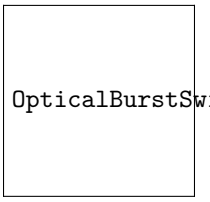
OpticalBurstSwitching_files/figure-latex/unnamed-chunk-5-8.pdf

```
### Now we would like to plot some individuals. We can see that here, using redundant variables in the l
### lead to a considerable difference.

## Firstly a graph of 20 randomly selected individuals drawn in the PCA with redundant variables, colou
## representation
```

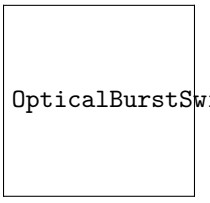
```
fviz_pca_ind(pcaOBS, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE
             select.ind = list(name = sample.int(n = nrow(OBS), size = 20)))
```



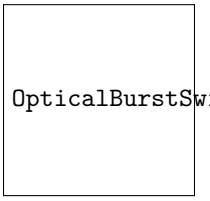

```
## Their quality of representation is always the best, equal to 1! This is due to the fact that the firs
## consist of a huge amount of redundant information, so that an individual can be easily well represen
## only these variables of him.

## Then we show a graph of 20 randomly selected individuals drawn in the PCA _without_ redundant variabl
## representation
fviz_pca_ind(pcaOBS2, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRU
             select.ind = list(name = sample.int(n = nrow(OBS), size = 20)))
```



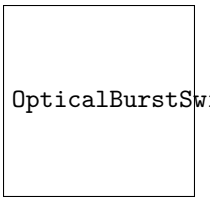

```
## All kinds of quality can be observed this time, which is more meaningful than in the previous case. I
## we are going to keep using in the rest of our analysis.
fviz_pca_ind(pcaOBS2, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRU
             select.ind = list(name = sample.int(n = nrow(OBS), size = 20)))
```




```
## We now try to involve the categorical information: `Node.Status` and `Class`.
## 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")
```



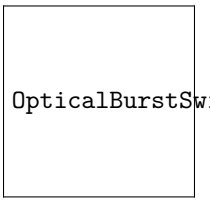

```
## Then a graph of randomly selected individuals coloured by subgroups of `Node.Status`
fviz_pca_ind(pcaOBS2, geom = "point", habillage = OBS$Class,  palette = "RdBu",
            select.ind = list(name = 1:1075), legend.title = "Class")
```


OpticalBurstSwitching_files/figure-latex/unnamed-chunk-5-13.pdf

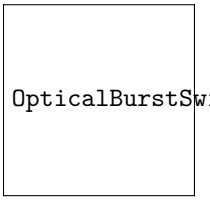
```
## Now we draw some biplots with randomly selected individuals and variables
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.4.3
```

```
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")
```


OpticalBurstSwitching_files/figure-latex/unnamed-chunk-5-14.pdf

```
library(FactoMineR)
#### Attention: On ne peut pas faire le graphe de quanti et quali ensemble! Car elles ne partagent pas
#### Par contre quali et individus ensemble c'est possible mais ca revient a sous-groupes avec ellipse.
#### Pour tracer quali et quanti ensemble, il faut absolument passer par FAMD.

pcaOBS3 <- PCA(cbind(OBScomplet[c("Packet.Received..Rate", "Packet_Received", "Packet_Transmitted", "Pa
                         "Average_Delay_Time_Per_Sec", "X10.Run.Delay", "Node", "Received_Byte", "Flood
                         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")
```

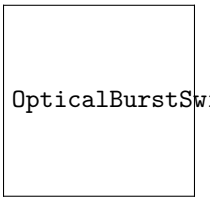

OpticalBurstSwitching_files/figure-latex/unnamed-chunk-5-15.pdf

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

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

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

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



OpticalBurstSwitching_files/figure-latex/unnamed-chunk-5-16.pdf