



Practical Work I

Data Mining 21-22 Q2

Calisalvo Veciana, Xènia
Domingo Navarro, Joan

Del Pozo Iglesias, Claudia
Hervás Roldán, Joan

Collado . . . , Victor
Singh, Jobanjot

March 25, 2022

Contents

1	Introduction	3
1.1	Motivation of the work and description of the problem	3
1.2	Data source presentation	3
1.3	Load Required Packages	3
1.4	Some useful functions	4
1.5	Load data	5
2	Initial analysys	6
2.1	Data Description	6
2.2	Univariate descriptive statistics of raw variables	8
3	Data preprocessing	37
3.1	Binary variables	37
3.2	Numerical Variables	37
3.3	Categorical Variables	40
4	Basic statistical descriptive analysis	43

5 PCA Analysis	44
5.1 Scree plot	44
5.2 Factorial map visualisation	46
6 Hierarchical Clustering	63
6.1 Precise description of the data used	65
6.2 Clustering method used, metrics and aggregation criteria used	65
6.3 Resulting Dendrogram	65
6.4 Discuss about how to get the final number of clusters	65
6.5 Table with a description of the clusters size	65
7 Profiling of clusters	65
7.1 Numeric Variables	66
7.2 Binary Variables	74
7.3 Categorical Variables	82
7.4 Classes' interpretation	89
8 Global discussions and general conclusions	90
9 Working plan	90
9.1 Gantt's diagram	90
9.2 Task assignment grid	90
9.3 Deviances of final scheduling with respect to the original	90
9.4 Risk contingency plan	92
10 Annex	93
10.1 Individuals projections	93
10.2 Common projection of numerical variables	107
10.3 Categorical variables projection	122
10.4 Projection of illustrative qualitative variables on individuals' map	122
10.5 Profiling plots	296
11 Bibliography	326

1 Introduction

1.1 Motivation of the work and description of the problem

With the augment of technological devices around us and the daily use we give them, it makes sense to be informed, or at least curious, about which device is worth buying depending on its characteristics and, primarily, its price. Nowadays everybody has a mobile phone in their pocket, so you must be interested in which cellphone can make the best photos, or which mobile has the most battery, or biggest screen. Since the first mobile phones, every year thousands of new models with better or worst characteristics are created, and the popular opinion seems to believe that the price of the mobile is an indicator of its quality. Is this belief true? Can some mobiles be more expensive and have worst components than others with a lower price?

We will try to find the relations between the characteristics and the price of the mobiles in our dataset and group them depending on their components. With all this organized information, we could check if our mobiles are overpriced or too cheap, we could see the evolution of the mobile prices and its quality, give a fair price to the new models and discard mobiles or whole brands regarding on their components and prices.

1.2 Data source presentation

- Data source

<https://www.kaggle.com/iabhishekofficial/mobile-price-classification>

- How to get the data?

Through the link above, click the download button. We will only use the train.csv dataset, since it includes the categorical variable price range.

- What is data about?

All the technical features of mobile phones (screen size, resolution, camera, etc) and their corresponding selling prices. The objective is to study the relation between said features and the different price ranges.

We will create the qualitative variables screen_dimension and pixel_dimension from 4 numerical variables called sc_h (Screen Height of mobile in cm), sc_w (Screen Width of mobile in cm), px_height (Pixel Resolution Height) and px_width (Pixel Resolution Width).

We will also factorize the numerical variables pc (Primary Camera megapixels), fc (Front Camera megapixel) and ram (Random Access Memory in MegaBytes).

- Basic structure of data matrix:

- nr. of records: 2000
- nr. of variables: 19
- nr. of numerical variables: 7
- nr of binary variables: 6
- nr of qualitative variables: 6
- number and % of missing data per variable: 390 missing values in variable screen width (19%)
- total % missing: 0.93%

1.3 Load Required Packages

```
# Clean workspace
rm(list=ls())

options(contrasts=c("contr.treatment","contr.treatment"))
requiredPackages <- c("effects", "FactoMineR", "car", "factoextra", "RColorBrewer", "ggplot2",
                      "dplyr", "ggmap", "ggthemes", "knitr", "magrittr", "class", "readr", "purrr", "cluster")
package.check <- lapply(requiredPackages, FUN = function(x) {
  if (!require(x, character.only = TRUE)) {
    install.packages(x, dependencies = TRUE)
  }
})
```

```

    library(x, character.only = TRUE)
  }
})

search()

## [1] ".GlobalEnv"           "package:cluster"      "package:purrr"
## [4] "package:readr"         "package:class"        "package:magrittr"
## [7] "package:knitr"         "package:ggthemes"     "package:ggmap"
## [10] "package:dplyr"         "package:RColorBrewer" "package:factoextra"
## [13] "package:ggplot2"       "package:car"          "package:FactoMineR"
## [16] "package:effects"       "package:carData"      "package:stats"
## [19] "package:graphics"      "package:grDevices"    "package:utils"
## [22] "package:datasets"      "package:methods"      "Autoloads"
## [25] "package:base"

```

1.4 Some useful functions

```

# Some useful functions
calcQ <- function(x) {
  s.x <- summary(x)
  iqr<-s.x[5]-s.x[2]
  list(souti=s.x[2]-3*iqr, mouti=s.x[2]-1.5*iqr, min=s.x[1], q1=s.x[2], q2=s.x[3],
       q3=s.x[5], max=s.x[6], mouts=s.x[5]+1.5*iqr, soutu=s.x[5]+3*iqr ) }

#Calcula els valor test de la variable Xnum per totes les modalitats del factor P
ValorTestXnum <- function(Xnum,P){
  #freq dis of fac
  nk <- as.vector(table(P));
  n <- sum(nk);
  #mitjanes x grups
  xk <- tapply(Xnum,P,mean);
  #valors test
  txk <- (xk-mean(Xnum))/(sd(Xnum)*sqrt((n-nk)/(n*nk)));
  #p-values
  ppx <- pt(txk,n-1,lower.tail=F);
  for(c in 1:length(levels(as.factor(P)))){if (ppx[c]>0.5){ppx[c]<-1-ppx[c]}}
  return (ppx)
}

ValorTestXquali <- function(P,Xquali){
  taula <- table(P,Xquali);
  n <- sum(taula);
  pk <- apply(taula,1,sum)/n;
  pj <- apply(taula,2,sum)/n;
  pf <- taula/(n*pk);
  pjm <- matrix(data=pj,nrow=dim(pf)[1],ncol=dim(pf)[2], byrow=TRUE);
  dpf <- pf - pjm;
  dvt <- sqrt(((1-pk)/(n*pk))*%*%t(pjm*(1-pj)));
  #i hi ha divisions iguals a 0 dona NA i no funciona
  zkj <- dpf;
  zkj[dpf!=0]<-dpf[dpf!=0]/dvt[dpf!=0];
  pzkj <- pnorm(zkj,lower.tail=F);
  for(c in 1:length(levels(as.factor(P)))){for (s in 1:length(levels(Xquali))){if (pzkj[c,s]> 0.5){pzkj[c,s]<-1-pzkj[c,s]}}}
  return (list(rowpf=pf,vtest=zkj,pval=pzkj))
}

```

1.5 Load data

```
#set working directory
setwd("D:/Drive/UNI/Q8/MD/MD-practical_work1")
df <- read_csv("data_mobiles.csv")

## Rows: 2000 Columns: 21

## -- Column specification -----
## Delimiter: ","
## dbl (21): battery_power, blue, clock_speed, dual_sim, fc, four_g, int_memory...
## 
## i Use 'spec()' to retrieve the full column specification for this data.
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.

names(df)

## [1] "battery_power" "blue"          "clock_speed"   "dual_sim"
## [5] "fc"            "four_g"         "int_memory"    "m_dep"
## [9] "mobile_wt"      "n_cores"        "pc"           "px_height"
## [13] "px_width"       "ram"            "sc_h"          "sc_w"
## [17] "talk_time"      "three_g"        "touch_screen" "wifi"
## [21] "price_range"

var_bin <- names(df)[c(2,4,6,18:20)]
summary(df[var_bin])

##      blue      dual_sim      four_g      three_g
## Min. :0.000  Min. :0.0000  Min. :0.0000  Min. :0.0000
## 1st Qu.:0.000  1st Qu.:0.0000  1st Qu.:0.0000  1st Qu.:1.0000
## Median :0.000  Median :1.0000  Median :1.0000  Median :1.0000
## Mean   :0.495  Mean   :0.5095  Mean   :0.5215  Mean   :0.7615
## 3rd Qu.:1.000  3rd Qu.:1.0000  3rd Qu.:1.0000  3rd Qu.:1.0000
## Max.   :1.000  Max.   :1.0000  Max.   :1.0000  Max.   :1.0000
##      touch_screen      wifi
## Min. :0.000  Min. :0.000
## 1st Qu.:0.000  1st Qu.:0.000
## Median :1.000  Median :1.000
## Mean   :0.503  Mean   :0.507
## 3rd Qu.:1.000  3rd Qu.:1.000
## Max.   :1.000  Max.   :1.000

var_num <- names(df)[c(1,3,7:10,17)]
summary(df[var_num])

##      battery_power      clock_speed      int_memory      m_dep
## Min.   :501.0  Min.   :0.500  Min.   : 2.00  Min.   :0.1000
## 1st Qu.:851.8  1st Qu.:0.700  1st Qu.:16.00  1st Qu.:0.2000
## Median :1226.0  Median :1.500  Median :32.00  Median :0.5000
## Mean   :1238.5  Mean   :1.522  Mean   :32.05  Mean   :0.5018
## 3rd Qu.:1615.2  3rd Qu.:2.200  3rd Qu.:48.00  3rd Qu.:0.8000
## Max.   :1998.0  Max.   :3.000  Max.   :64.00  Max.   :1.0000
##      mobile_wt      n_cores      talk_time
## Min.   : 80.0  Min.   :1.000  Min.   : 2.00
## 1st Qu.:109.0  1st Qu.:3.000  1st Qu.: 6.00
## Median :141.0  Median :4.000  Median :11.00
## Mean   :140.2  Mean   :4.521  Mean   :11.01
## 3rd Qu.:170.0  3rd Qu.:7.000  3rd Qu.:16.00
## Max.   :200.0  Max.   :8.000  Max.   :20.00
```

```
var_cat <- names(df)[c(5,11:16,21)]
summary(df[var_cat])
```

	fc	pc	px_height	px_width
## Min.	0.000	Min. : 0.000	Min. : 0.0	Min. : 500.0
## 1st Qu.	1.000	1st Qu.: 5.000	1st Qu.: 282.8	1st Qu.: 874.8
## Median	3.000	Median :10.000	Median : 564.0	Median :1247.0
## Mean	4.309	Mean : 9.916	Mean : 645.1	Mean :1251.5
## 3rd Qu.	7.000	3rd Qu.:15.000	3rd Qu.: 947.2	3rd Qu.:1633.0
## Max.	19.000	Max. :20.000	Max. :1960.0	Max. :1998.0
	ram	sc_h	sc_w	price_range
## Min.	256	Min. : 5.00	Min. : 0.000	Min. :0.00
## 1st Qu.	1208	1st Qu.: 9.00	1st Qu.: 2.000	1st Qu.:0.75
## Median	2146	Median :12.00	Median : 5.000	Median :1.50
## Mean	2124	Mean :12.31	Mean : 5.767	Mean :1.50
## 3rd Qu.	3064	3rd Qu.:16.00	3rd Qu.: 9.000	3rd Qu.:2.25
## Max.	3998	Max. :19.00	Max. :18.000	Max. :3.00

2 Initial analisys

2.1 Data Description

2.1.1 Metadata description

Each row of the matrix contains different data about the specifications of a cell phone, the columns of these rows can be divided into 3 types; numerical, qualitative and boolean. To begin with, the numerical type variables are:

- “battery_power”, indicates the battery capacity.
- “clock_speed”, indicates the speed at which the CPU executes the instructions.
- “int_memory”, indicates the internal memory of the device.
- “m_dep”, indicates the depth of the device.
- “mobile_wt”, indicates the weight of the device.
- “n_cores”, indicates the number of cores of the device.
- “talk_time”, indicates the device battery life.

Then the boolean variables are: * “blue”, indicates whether the device has Bluetooth. * “dual_sim”, indicates if the device has support for carrying two sim cards. * “four_g”, indicates if the device has 4G. * “three_g”, indicates if the device has 3G. * “touch_screen”, indicates if the device has a touch screen. * “wifi”, indicates if the device can connect to a wifi network.

Finally the qualitative type variables are: * “fc”, indicates the quality of the front camera and is divided into 5 modalities depending on the number of megapixels. * “pc”, indicates the quality of the main camera and is divided into 5 modalities depending on the number of megapixels. * “px_height” and “px_width” are transformed into a new variable “pixel_dimension”, which indicates the pixels per inch of the device and is divided into 4 modalities depending on the PPI. * “ram”, which indicates the RAM and is divided into 3 modalities depending on the cost of the device. * “sc_h” and “sc_w” are transformed into a new variable “screen_dimension”, which indicates the size of the screen and is divided into 3 modalities depending on the size of the device. * “price range”, which is divided into 4 modalities depending on the price range of the device.

2.1.2 Data Mining process

At first, in the selection process, we used a website named kaggle to download our initial data matrix and loaded it in RStudio to visualize all the variables. Next we entered in the data preparation process, in this part we made a formal metadata table in order to have a clear representation of all variables. We searched for unnecessary outliers, in our case there weren't any, but we had missing data in the screen_width variable so we imputed it with K-Nearest-Neighbour (KNN) algorithm. /

After that we started the data transformation process, in our case we didn't have enough qualitative variables so we had to transform some of the numerical ones and also modify the metadata accordingly. /

With our new preprocessed matrix we started the data mining and interpretation process, at first we used a factorial method named Principal Components Analysis (PCA) with our numerical variables. We applied this

Variable	Modalities	Meaning	Type	Measuring unit	Missing code	Measuring procedure	Range	Role
battery_power		battery capacity	numerical	mAh			[501,1998]	Explanatory
blue		has bluetooth or not	boolean					Explanatory
clock_speed		speed at which the CPU executes instructions	numerical	GHz			[0,5,3]	Explanatory
dual_sim		has dual sim support or not	boolean					Explanatory
fc		front camera	qualitative	megapixels	0			Explanatory
	no camera						0	
	bad quality						[1,5]	
	regular quality						[5,10]	
	good quality						[10,15]	
	excellent quality						[15,19]	
four_g		has 4G or not	boolean					Explanatory
int_memory		internal Memory	numerical	GB			[2,64]	Explanatory
m_dep		mobile depth	numerical	cm			[0,1,1]	Explanatory

Figure 1: alt text

mobile_wt		mobile weight	numerical	grams			[80,200]	Explanatory
n_cores		number of cores	numerical	int			[1,8]	Explanatory
pc		primary Camera	qualitative	megapixels	0			Explanatory
	no camera						0	
	bad quality						[1,5]	
	regular quality						[5,10]	
	good quality						[10,15]	
	excellent quality						[15,20]	
pixel_dimension		resolution	qualitative	PPI	0	$\text{sqrt(px_height}^2 + \text{px_width}^2) / \text{sqrt(sc_h}^2 + \text{sc_w}^2)$		Explanatory
	bad resolution						[0,150]	
	regular resolution						[150,300]	
	good resolution						[300,450]	
	excellent resolution						>450	
ram		Random Access	qualitative	MB				Explanatory

Figure 2: alt text

		Memory						
	low cost						[256,2048]	
	mid cost						[2048,3072]	
	high cost						[3072,4096]	
screen_dimension		screen size	qualitative	cm ²	0	sc_h * sc_w		Explanatory
	small						[0,115]	
	medium						[115,230]	
	big						[230,345]	
talk_time		duration of battery	numerical	hours			[2,20]	Explanatory
three_g		has 3G or not	boolean					Explanatory
touch_screen		has touch screen or not	boolean					Explanatory
wifi		has wifi or not	boolean					Explanatory
price_range		the price range	qualitative	euros				Target
	low cost						<200	
	medium cost						[200,600]	
	high cost						[600,1000]	
	very high cost						>1000	

Figure 3: alt text

method to simplify the complexity of our sample spaces with many dimensions while preserving most of their information and to visualize these results we used the scree plot and the factorial map. Next we did the clustering to group the variables with similar characteristics or distance and represent it with a dendrogram. The following step was doing the profiling to generate descriptions of the clusters we calculated previously. To interpret and better understand this information we made different kinds of graphs, such as barplots, histograms, boxplots, etc.. In our workflow diagram, we have drawn return arrows from the interpretation and evaluation processes to their corresponding data mining process in case, after seeing them, something had to be modified. /

Finally, the last step of our work was to gain the information from the scree plot and factorial map that we generated with the PCA analysis, the dendrogram we extracted from the clustering process and the different plots we made during the profiling to make our final conclusions on the project.

2.2 Univariate descriptive statistics of raw variables

```
dataset<- "data_mobiles"
vars <- names(df)[c(1:21)]
colNames<- colnames(df[vars])
colors = c("#EF5350", "#EC407A", "#AB47BC", "#5C6BC0", "#42A5F5", "#26C6DA", "#26A69A", "#66BB6A", "#D4E157")
```

2.2.1 Numerical variables

```
var <- df$battery_power
cat("Minimum value: ", min(var), "\n")      # Get minimum value

## Minimum value: 501

cat("Maximum value: ", max(var), "\n")      # Get maximum value

## Maximum value: 1998
```

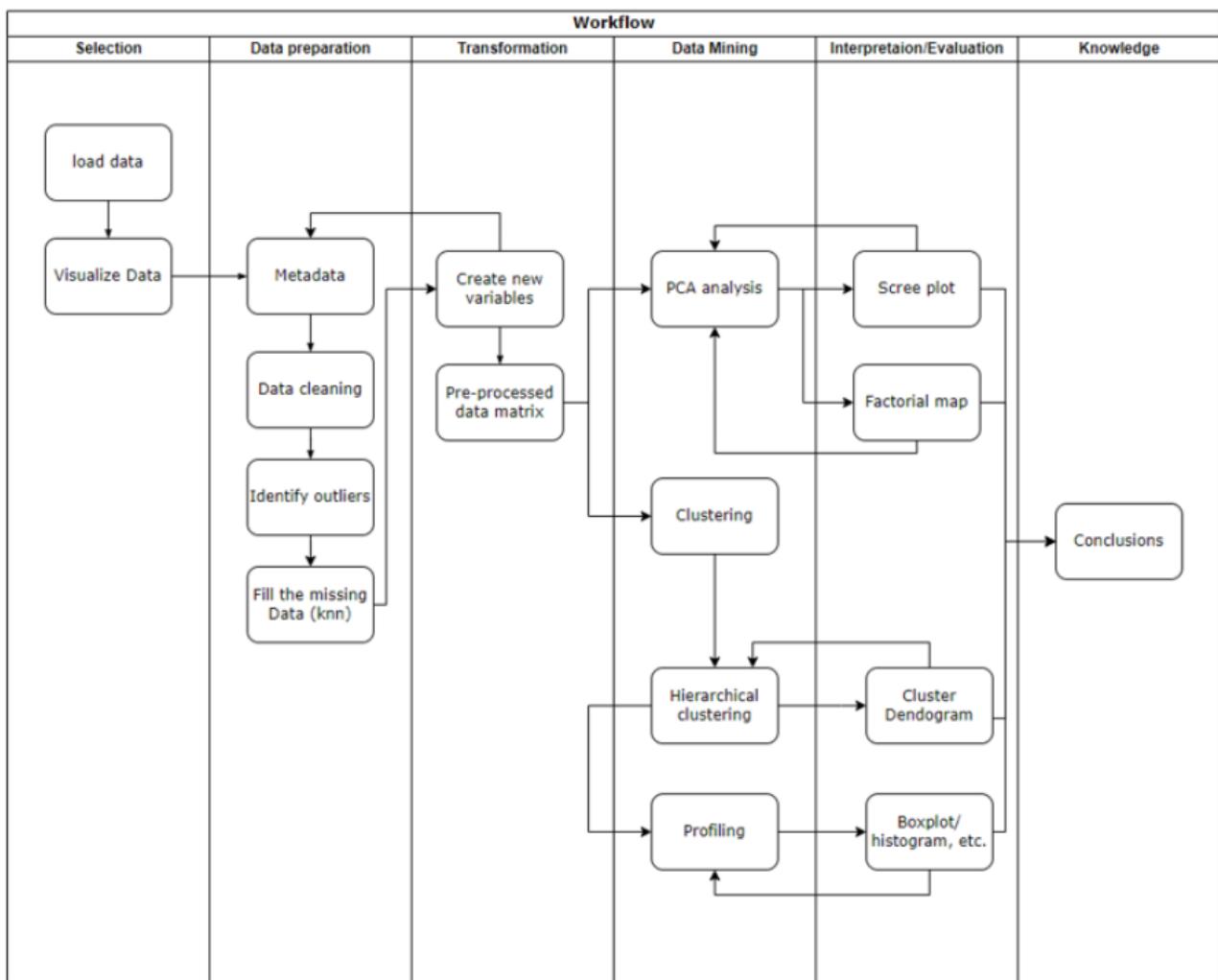


Figure 4: alt text

```

cat("Mean: ", mean(var))                                # Get mean

## Mean: 1238.518

cat("Median: ", median(var), "\n")                      # Get median

## Median: 1226

cat("Variance: ", var(var), "\n")                       # Get variance

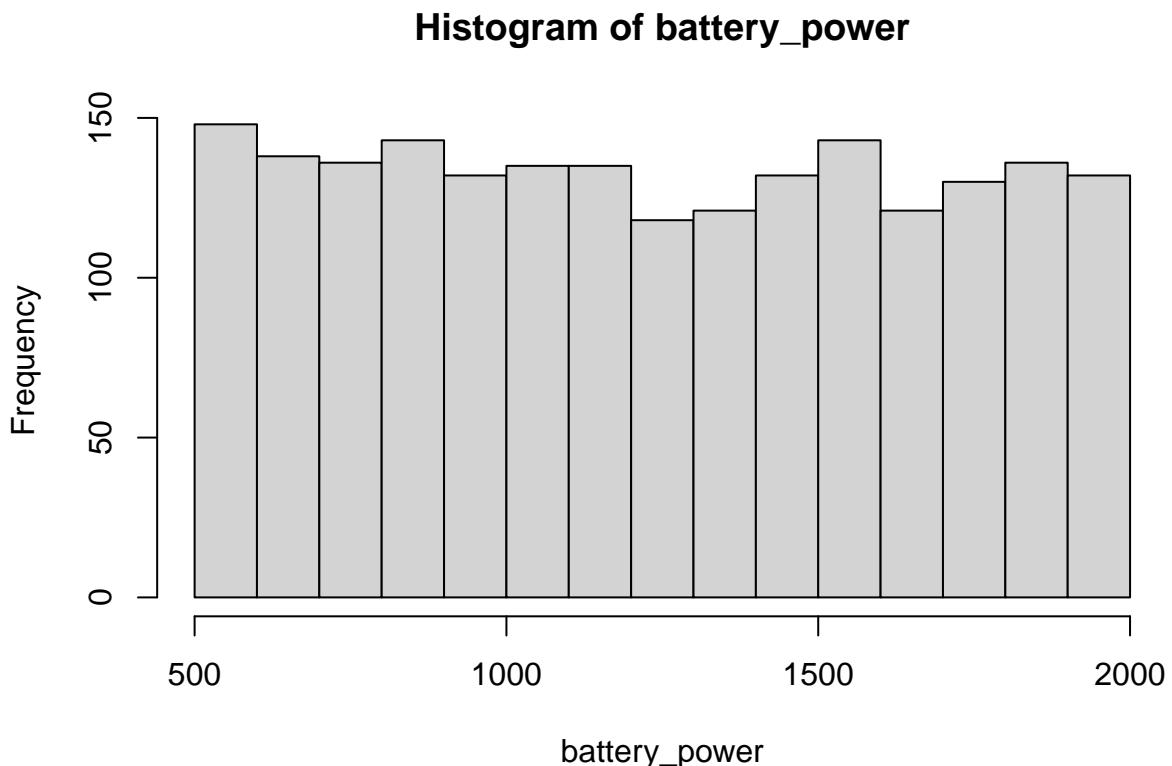
## Variance: 193088.4

cat("Standard deviation: ", sd(var), "\n")             # Get standard deviation

## Standard deviation: 439.4182

# Get histogram
hist(var, xlab = "battery_power", main = "Histogram of battery_power")

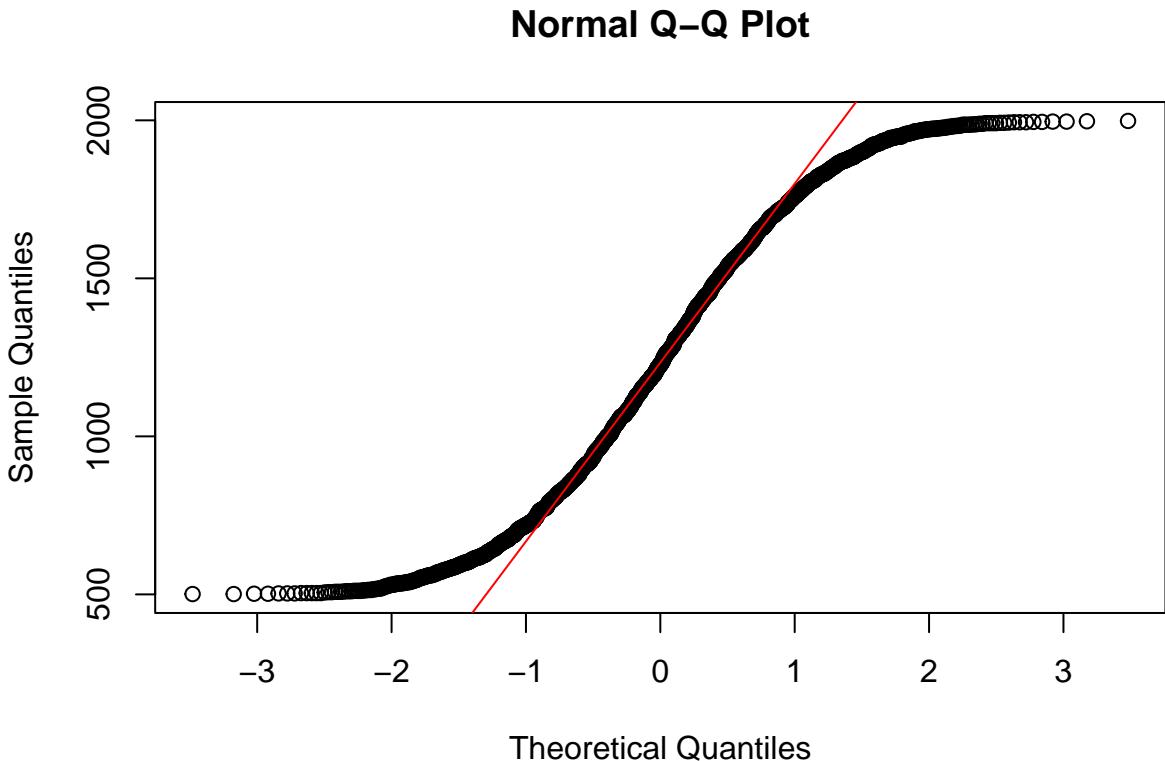
```



```

# Get a plot to see if follow a normal distribution
qqnorm(var); qqline(var, col = "red")

```



Here we can see that our data does not follow a normal distribution and that the frequency has its lower peaks in 1250 and 1700 and its highest at 600 and 1500 approximately.

```

var <- df$clock_speed
cat("Minimum value: ", min(var), "\n")      # Get minimum value

## Minimum value:  0.5

cat("Maximum value: ", max(var), "\n")      # Get maximum value

## Maximum value:  3

cat("Mean: ", mean(var), "\n")                # Get mean

## Mean:  1.52225

cat("Median: ", median(var), "\n")            # Get median

## Median:  1.5

cat("Variance: ", var(var), "\n")              # Get variance

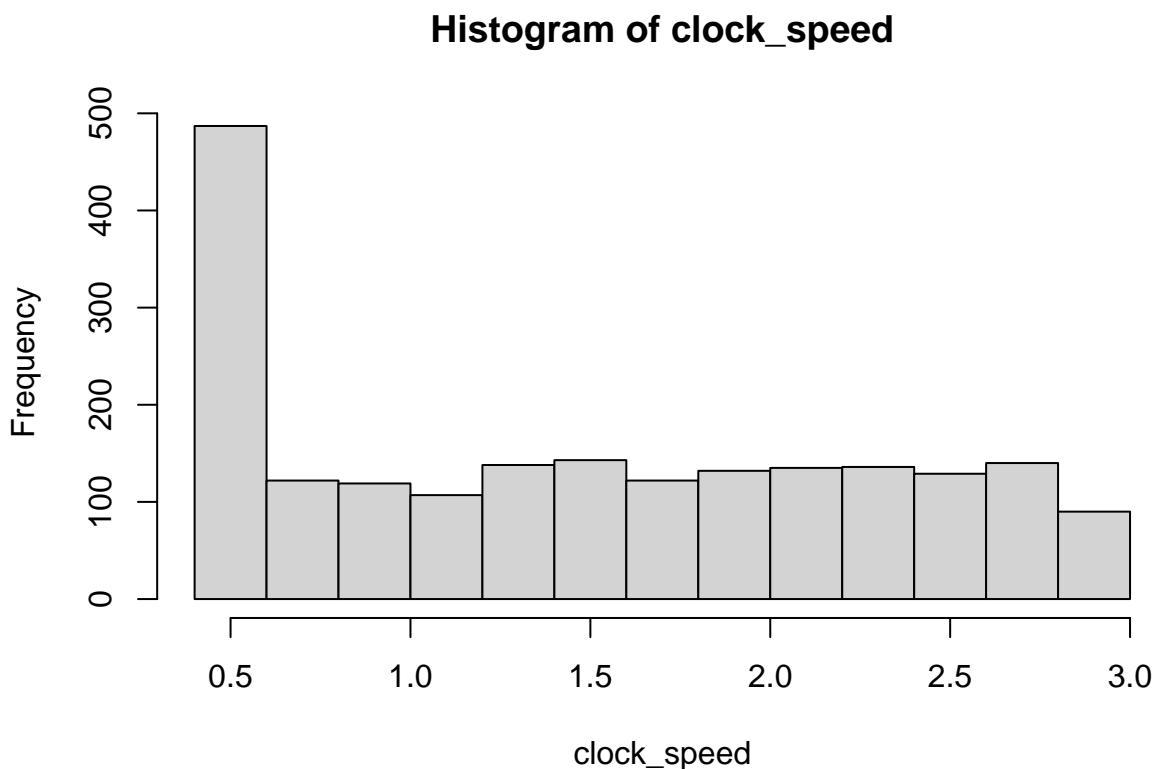
## Variance:  0.6658629

cat("Standard deviation: ", sd(var), "\n")    # Get standard deviation

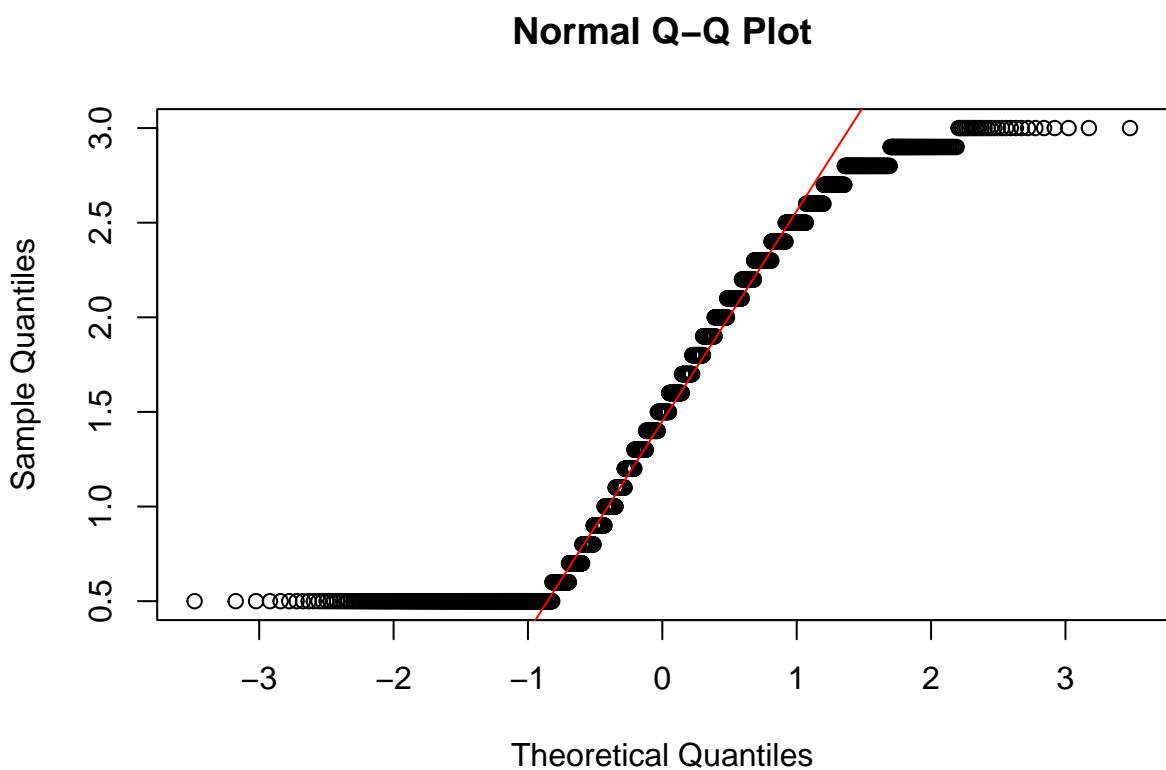
## Standard deviation:  0.8160042

# Get histogram
hist(var, xlab = "clock_speed", main = "Histogram of clock_speed")

```



```
# Get a plot to see if follow a normal distribution
qqnorm(var); qqline(var, col = "red")
```



We can observe that it does not follow a normal distribution and the frequency barely reaches the value 200.

```
var <- df$int_memory
cat("Minimum value: ", min(var), "\n")      # Get minimum value
```

```

## Minimum value: 2

cat("Maximum value: ", max(var), "\n")      # Get maximum value

## Maximum value: 64

cat("Mean: ", mean(var), "\n")            # Get mean

## Mean: 32.0465

cat("Median: ", median(var), "\n")        # Get median

## Median: 32

cat("Variance: ", var(var), "\n")         # Get variance

## Variance: 329.267

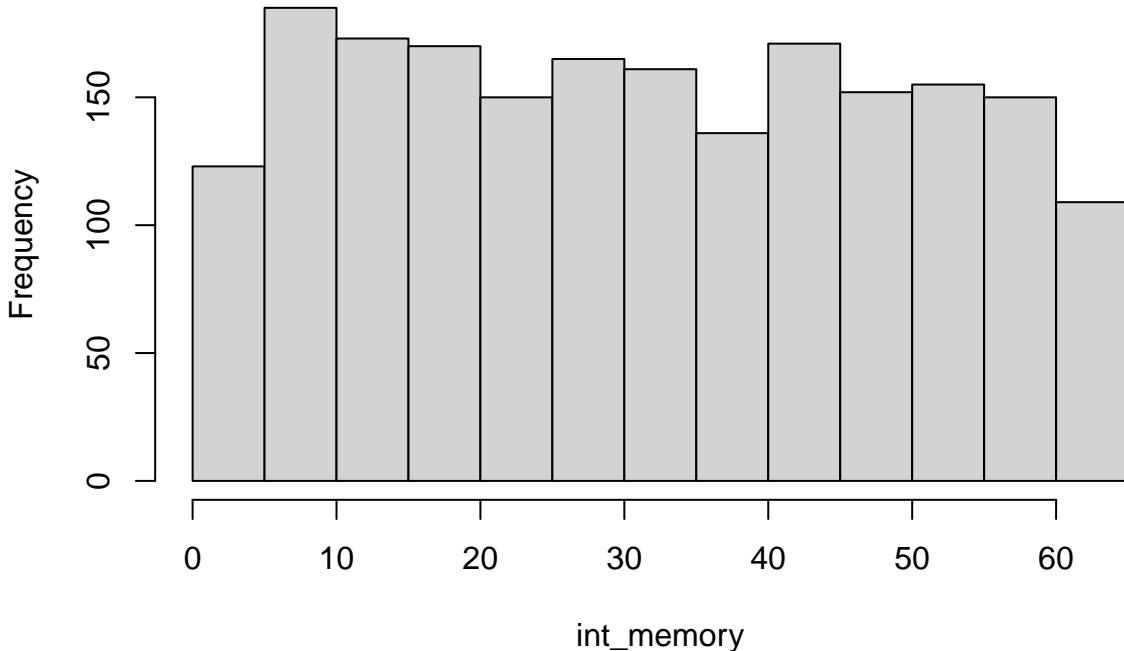
cat("Standard deviation: ", sd(var), "\n") # Get standard deviation

## Standard deviation: 18.14571

# Get histogram
hist(var, xlab = "int_memory", main = "Histogram of int_memory")

```

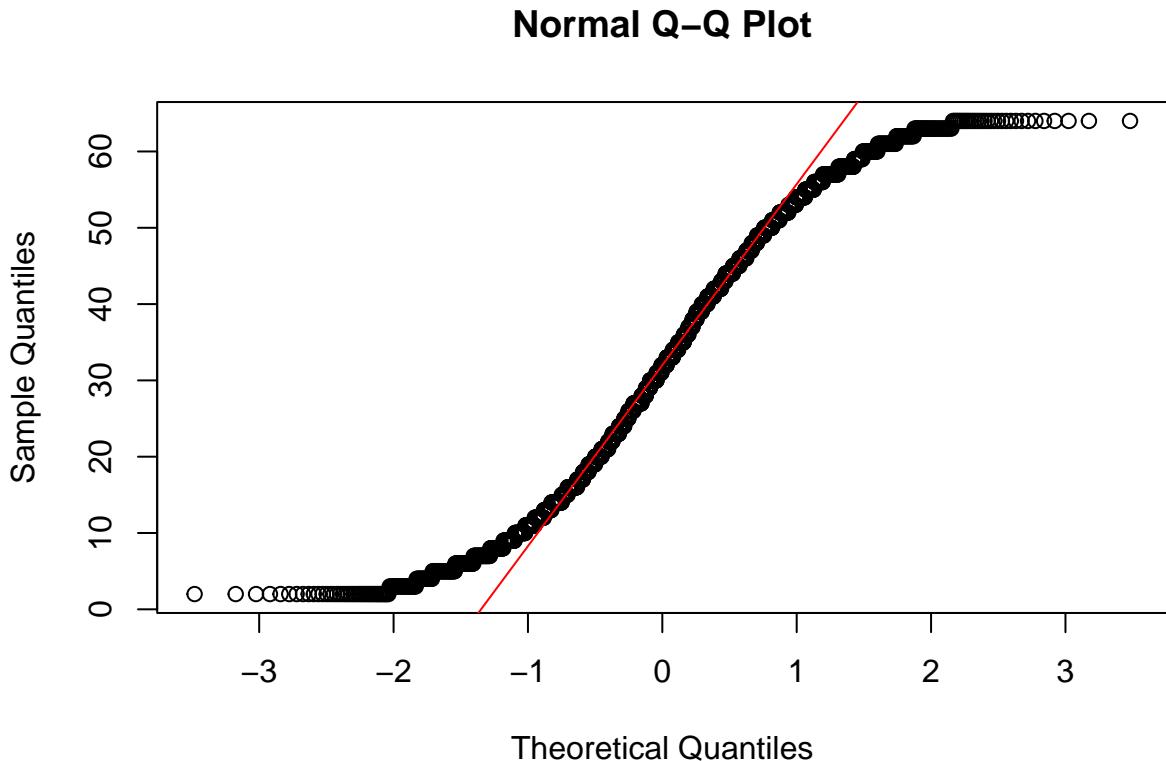
Histogram of int_memory



```

# Get a plot to see if follow a normal distribution
qqnorm(var); qqline(var, col = "red")

```



We can observe that the highest values are between 5 and 20 and on the other hand 40.

```
# Age
var <- df$m_dep
cat("Minimum value: ", min(var), "\n")      # Get minimum value

## Minimum value:  0.1

cat("Maximum value: ", max(var), "\n")      # Get maximum value

## Maximum value:  1

cat("Mean: ", mean(var), "\n")                # Get mean

## Mean:  0.50175

cat("Median: ", median(var), "\n")             # Get median

## Median:  0.5

cat("Variance: ", var(var), "\n")              # Get variance

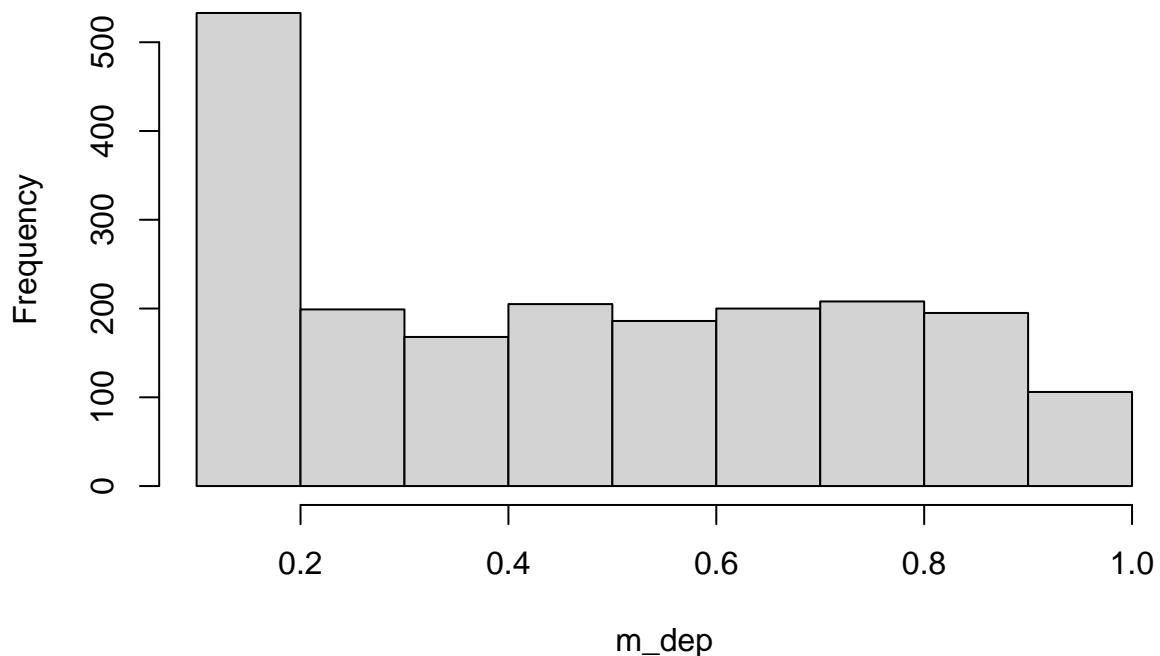
## Variance:  0.08318353

cat("Standard deviation: ", sd(var), "\n")    # Get standard deviation

## Standard deviation:  0.2884155

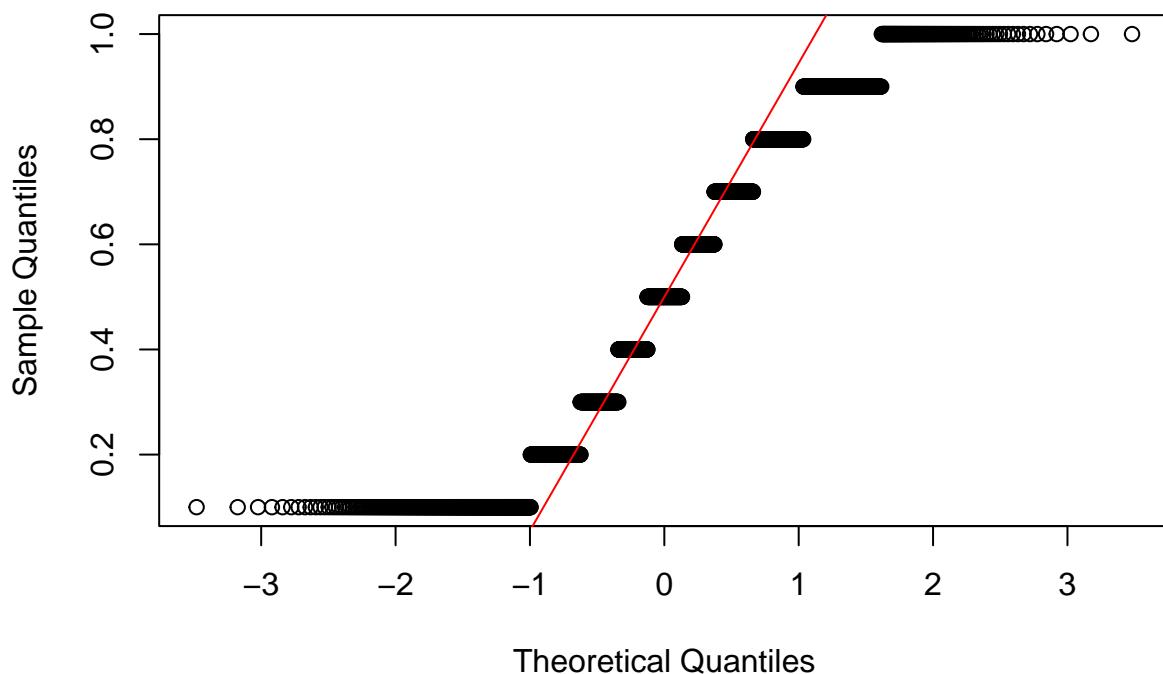
# Get histogram
hist(var, xlab = "m_dep", main = "Histogram of m_dep")
```

Histogram of m_dep



```
# Get a plot to see if follow a normal distribution
qqnorm(var); qqline(var, col = "red")
```

Normal Q-Q Plot



The values remain very low, the higher frequency is in the closest values to the axis.

```
var <- df$mobile_wt
cat("Minimum value: ", min(var), "\n")      # Get minimum value
```

```

## Minimum value: 80

cat("Maximum value: ", max(var), "\n")      # Get maximum value

## Maximum value: 200

cat("Mean: ", mean(var), "\n")            # Get mean

## Mean: 140.249

cat("Median: ", median(var), "\n")        # Get median

## Median: 141

cat("Variance: ", var(var), "\n")         # Get variance

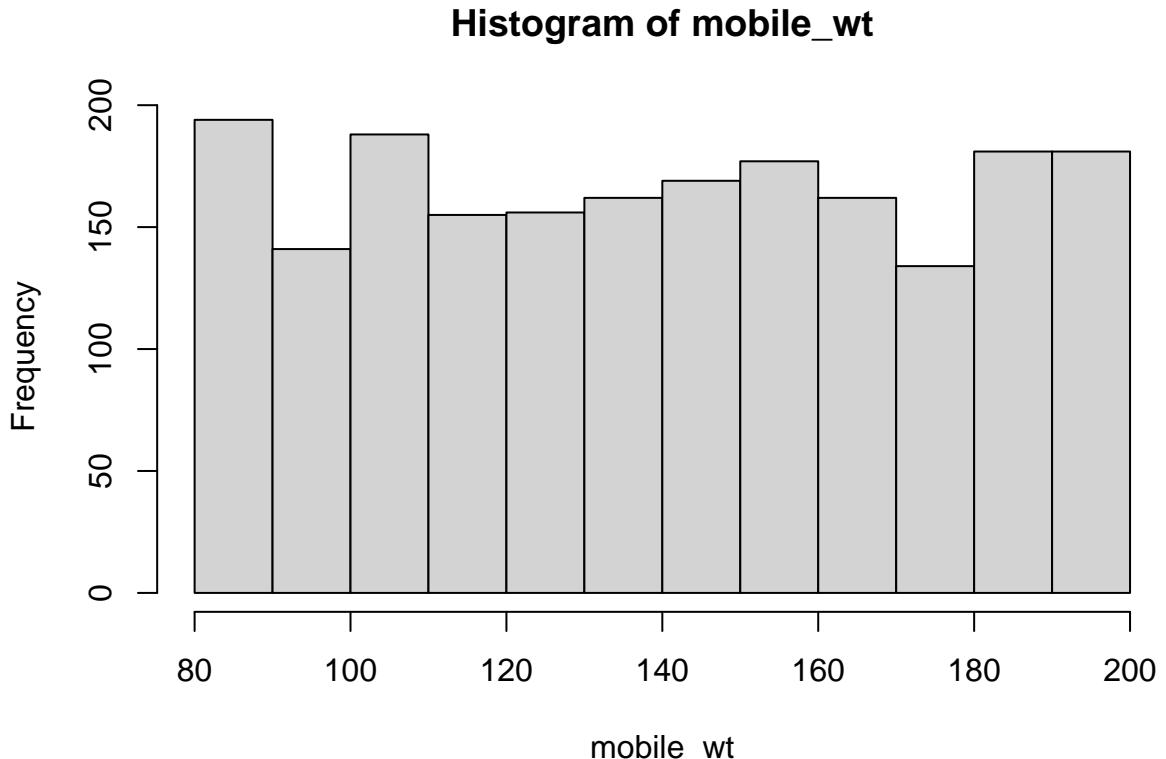
## Variance: 1253.136

cat("Standard deviation: ", sd(var), "\n") # Get standard deviation

## Standard deviation: 35.39965

# Get histogram
hist(var, xlab = "mobile_wt", main = "Histogram of mobile_wt")

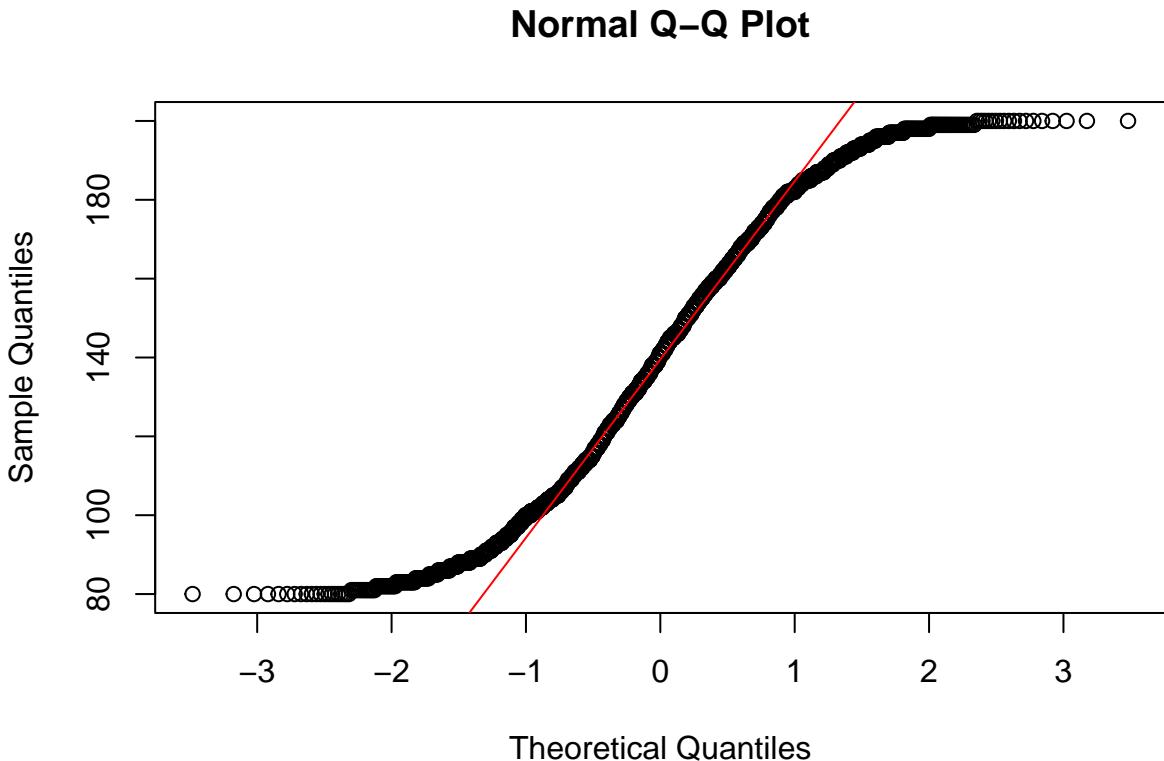
```



```

# Get a plot to see if follow a normal distribution
qqnorm(var); qqline(var, col = "red")

```



The lower frequency values are around 90 and 170 and the higher ones in 80, 100 to 110 and from 180 to 200.

```

var <- df$n_cores
cat("Minimum value: ", min(var), "\n")      # Get minimum value

## Minimum value:  1

cat("Maximum value: ", max(var), "\n")        # Get maximum value

## Maximum value:  8

cat("Mean: ", mean(var), "\n")                # Get mean

## Mean:  4.5205

cat("Median: ", median(var), "\n")             # Get median

## Median:  4

cat("Variance: ", var(var), "\n")              # Get variance

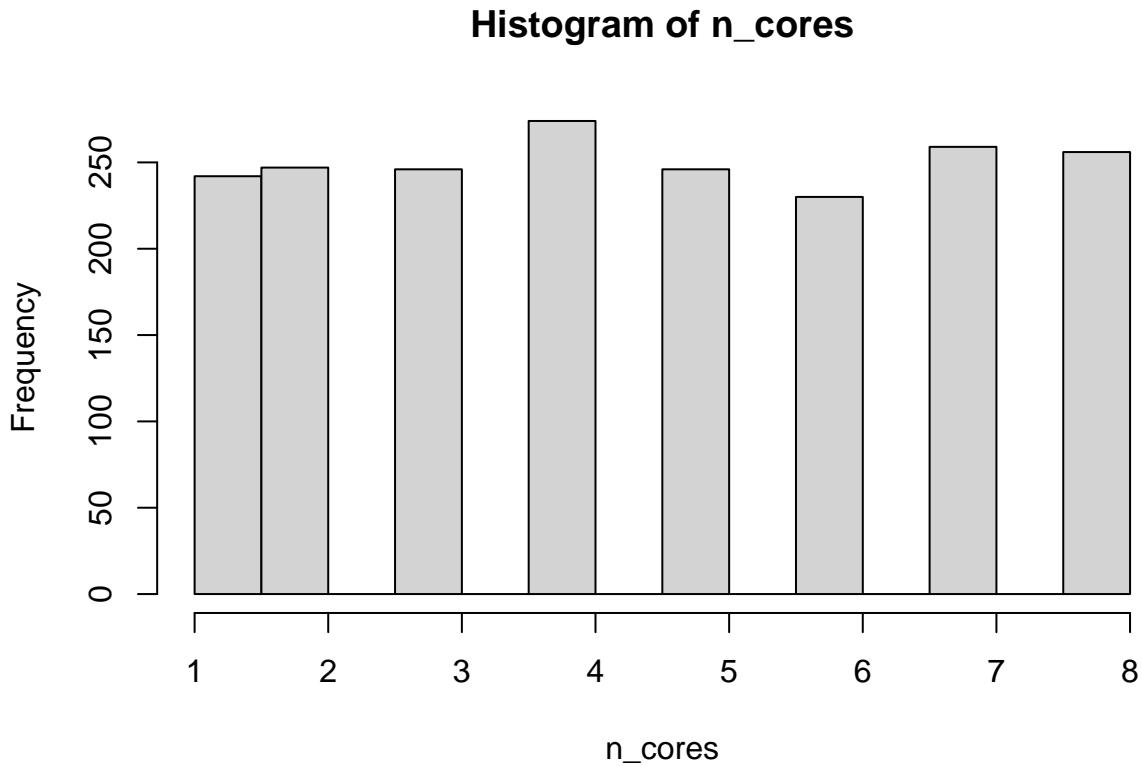
## Variance:  5.234197

cat("Standard deviation: ", sd(var), "\n")    # Get standard deviation

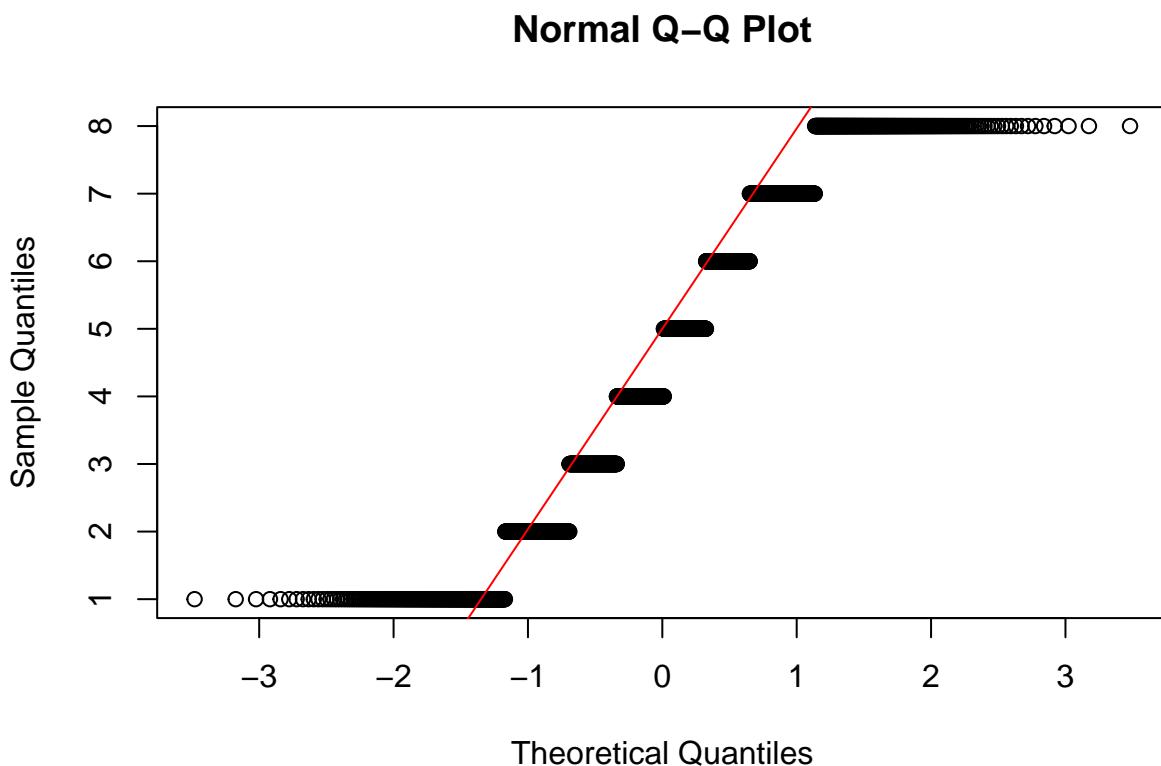
## Standard deviation:  2.287837

# Get histogram
hist(var, xlab = "n_cores", main = "Histogram of n_cores")

```



```
# Get a plot to see if follow a normal distribution
qqnorm(var); qqline(var, col = "red")
```



Values are very low, the maximum is 8, we can observe gaps between values.

```
var <- df$talk_time
cat("Minimum value: ", min(var), "\n")      # Get minimum value
```

```

## Minimum value: 2

cat("Maximum value: ", max(var), "\n")      # Get maximum value

## Maximum value: 20

cat("Mean: ", mean(var), "\n")            # Get mean

## Mean: 11.011

cat("Median: ", median(var), "\n")        # Get median

## Median: 11

cat("Variance: ", var(var), "\n")         # Get variance

## Variance: 29.85481

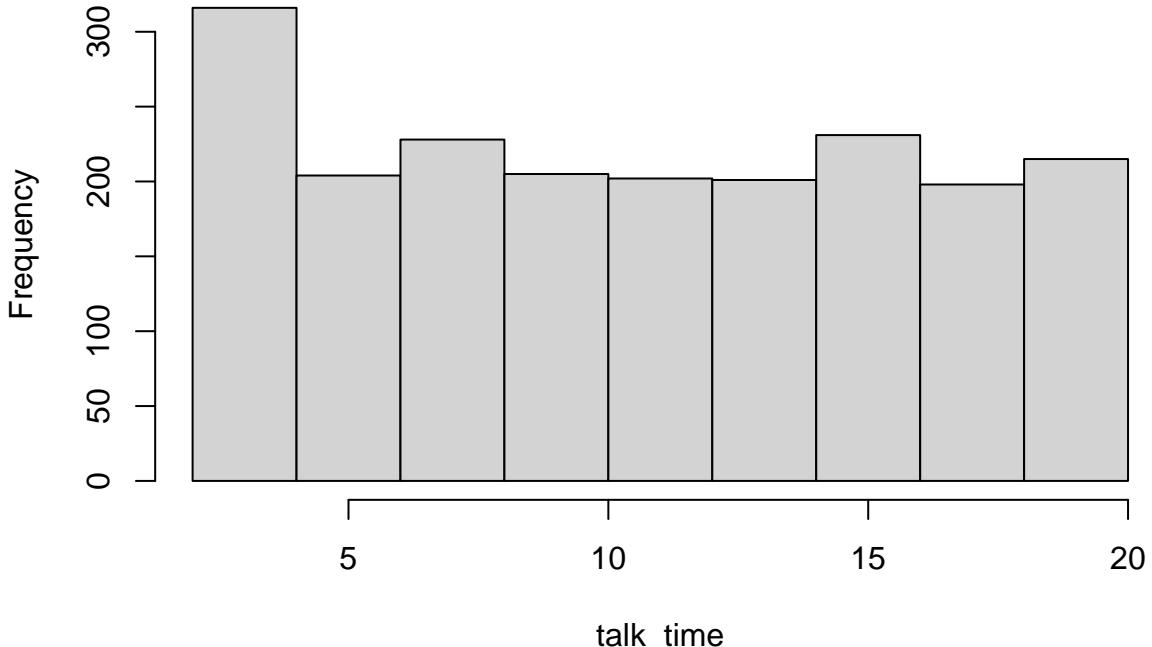
cat("Standard deviation: ", sd(var), "\n") # Get standard deviation

## Standard deviation: 5.463955

# Get histogram
hist(var, xlab = "talk_time", main = "Histogram of talk_time")

```

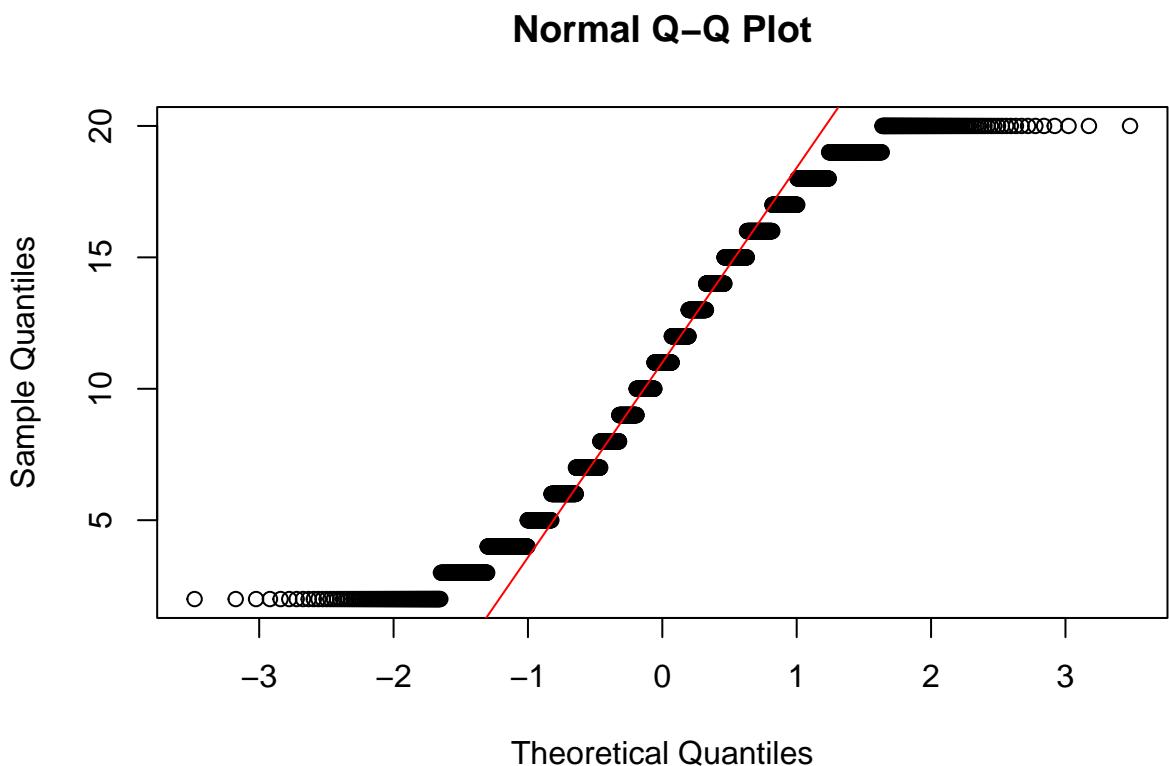
Histogram of talk_time



```

# Get a plot to see if follow a normal distribution
qqnorm(var); qqline(var, col = "red")

```

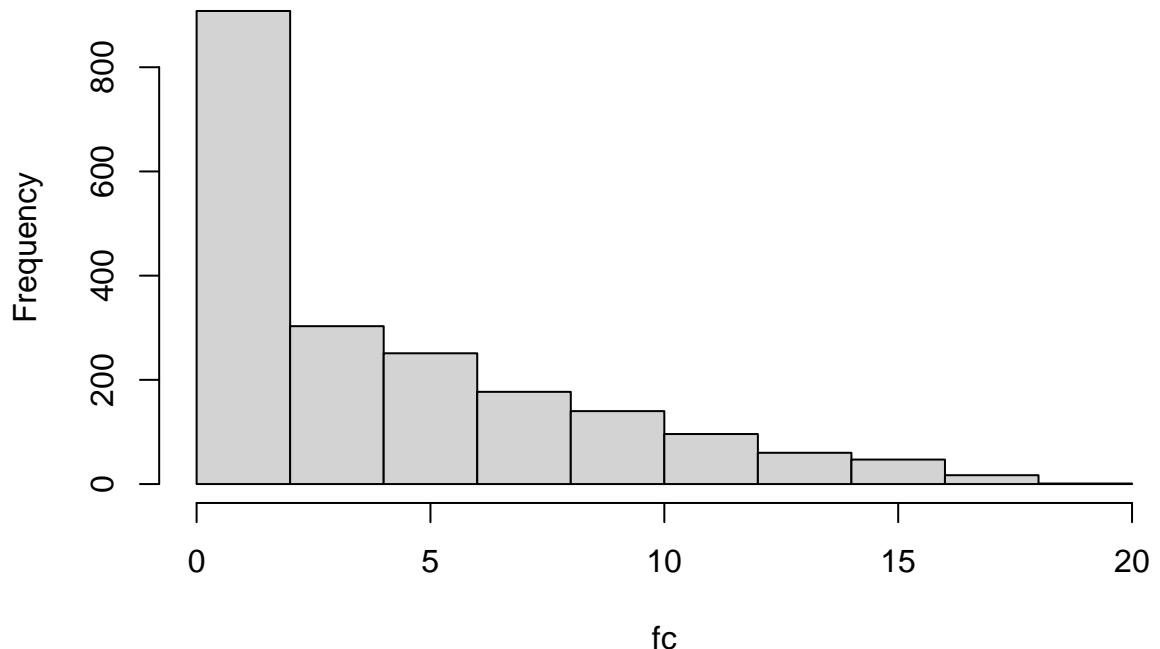


The frequency distribution is very equal, as we can see that the mean is 11 and the maximum value is 20. The higher frequency values are from 0 to 5.

2.2.2 Categorical variables

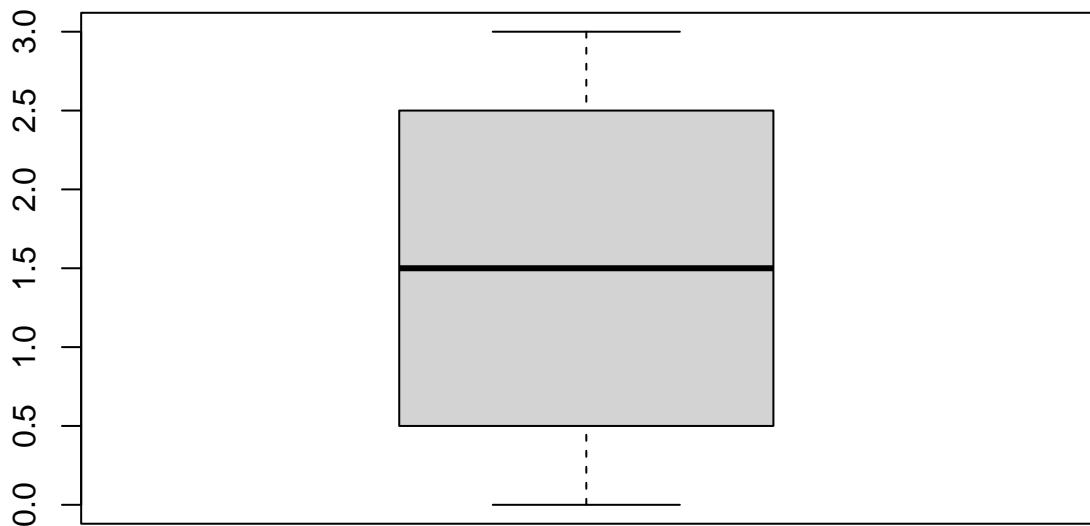
```
var <- df$fc
hist(var, xlab = "fc", main = "Histogram of fc")
```

Histogram of fc



```
boxplot(df$price_range, main="Boxplot of fc")
```

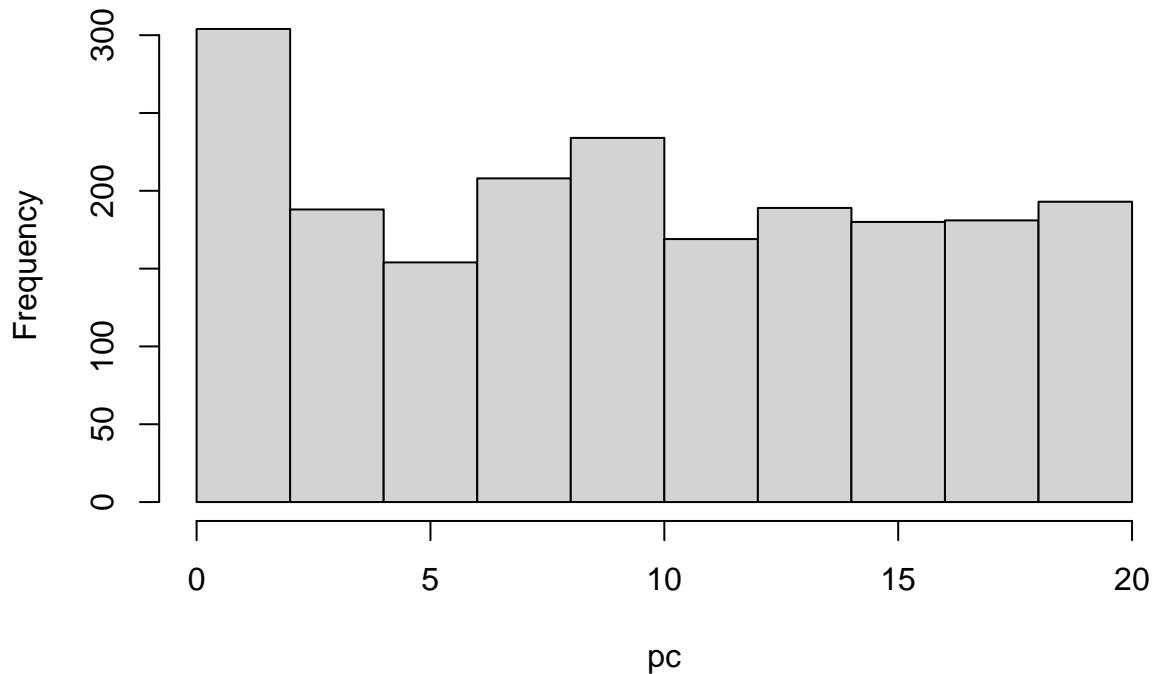
Boxplot of fc



The frequency decreases as the fc increases.

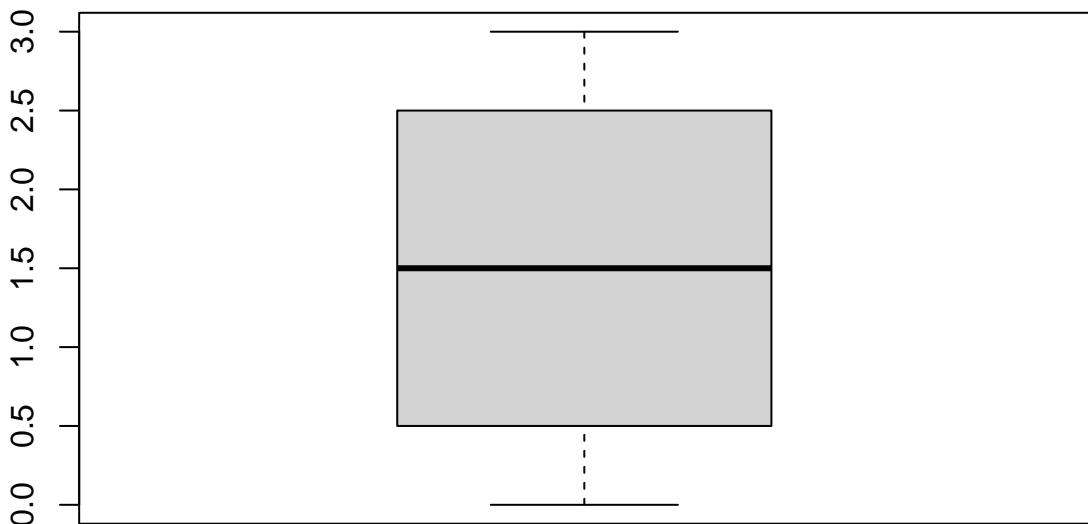
```
var <- df$pc  
hist(var, xlab = "pc", main = "Histogram of pc")
```

Histogram of pc



```
boxplot(df$price_range, main="Boxplot of pc")
```

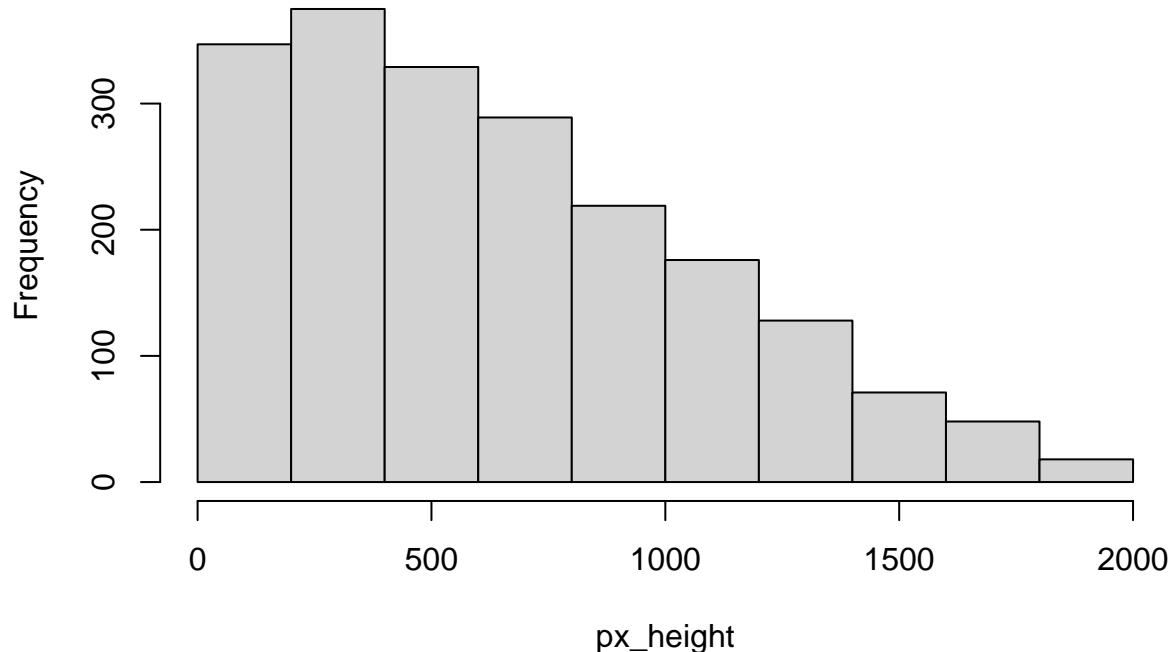
Boxplot of pc



The frequency distribution is balanced, excepting for the lower values near the axis.

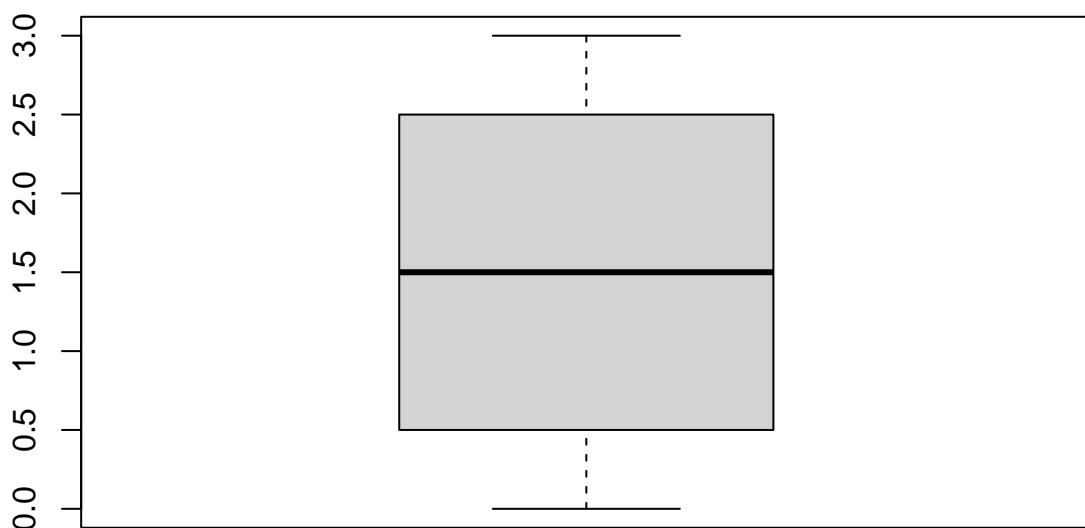
```
var <- df$px_height  
hist(var, xlab = "px_height", main = "Histogram of px_height")
```

Histogram of px_height



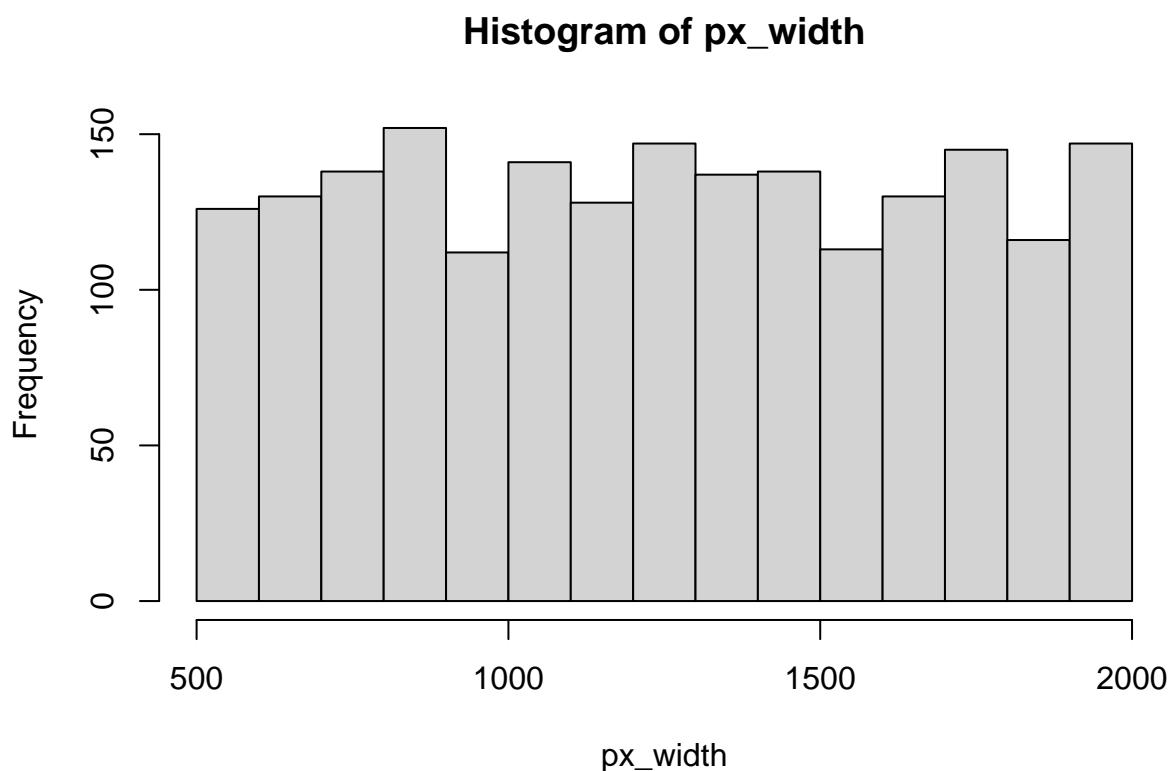
```
boxplot(df$price_range, main="Boxplot of px_height")
```

Boxplot of px_height

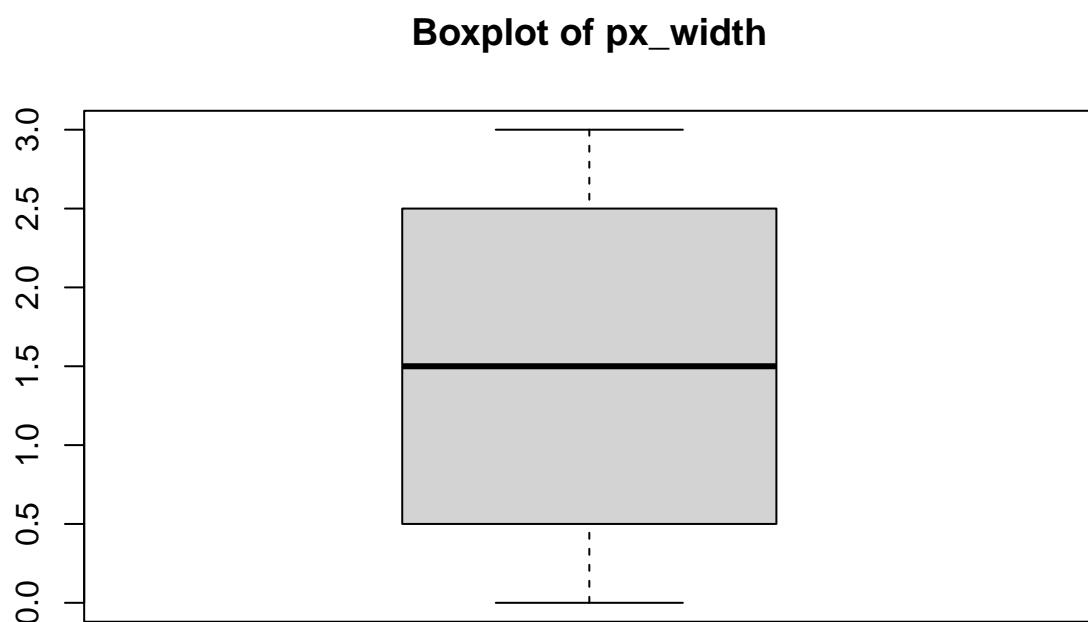


The frequency distribution decreases uniformly.

```
var <- df$px_width  
hist(var, xlab = "px_width", main = "Histogram of px_width")
```

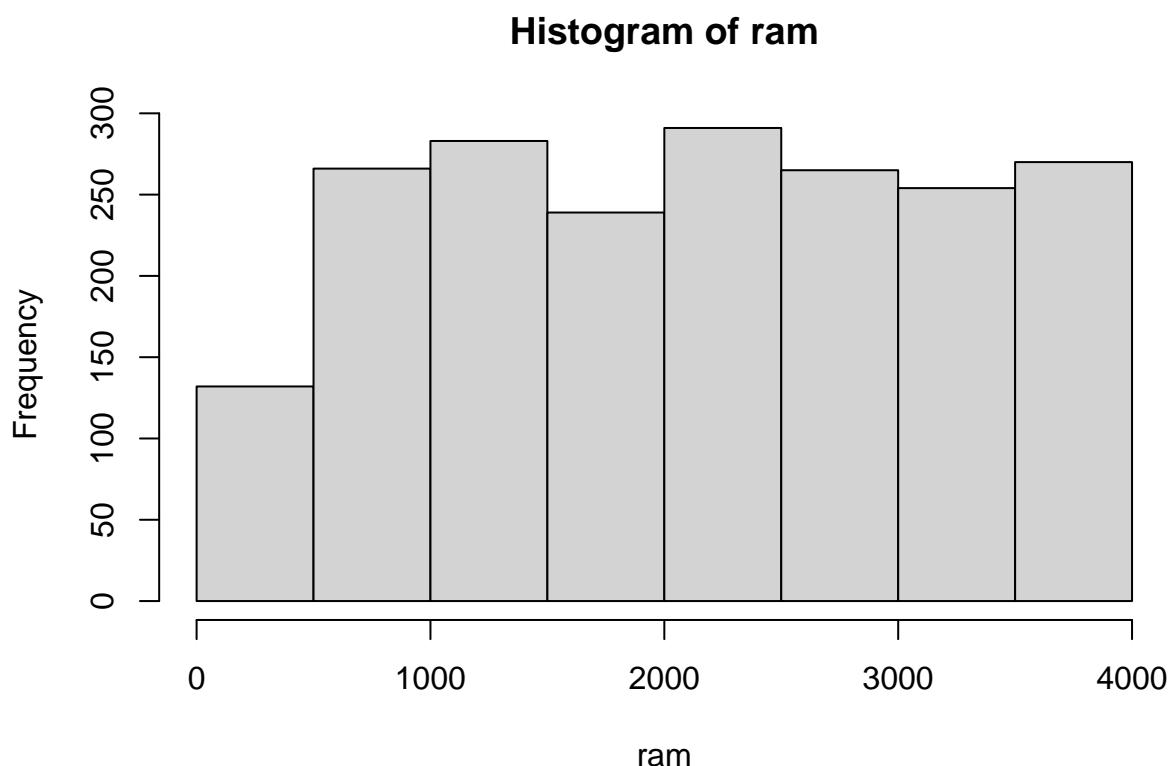


```
boxplot(df$price_range, main="Boxplot of px_width")
```

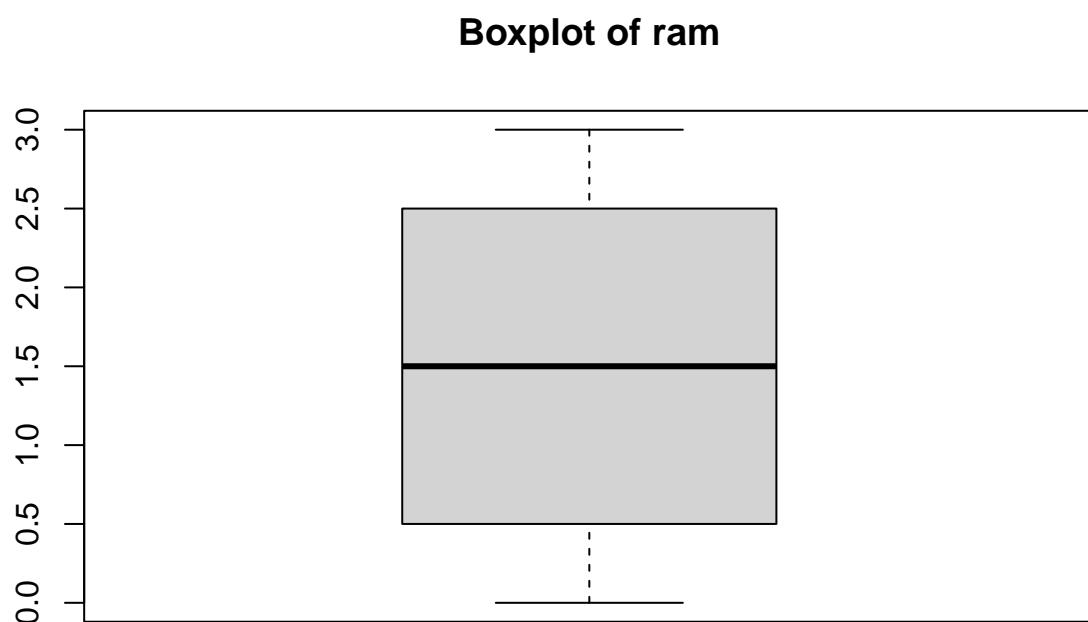


The distribution remains with high values along the data.

```
var <- df$ram  
hist(var, xlab = "ram", main = "Histogram of ram")
```

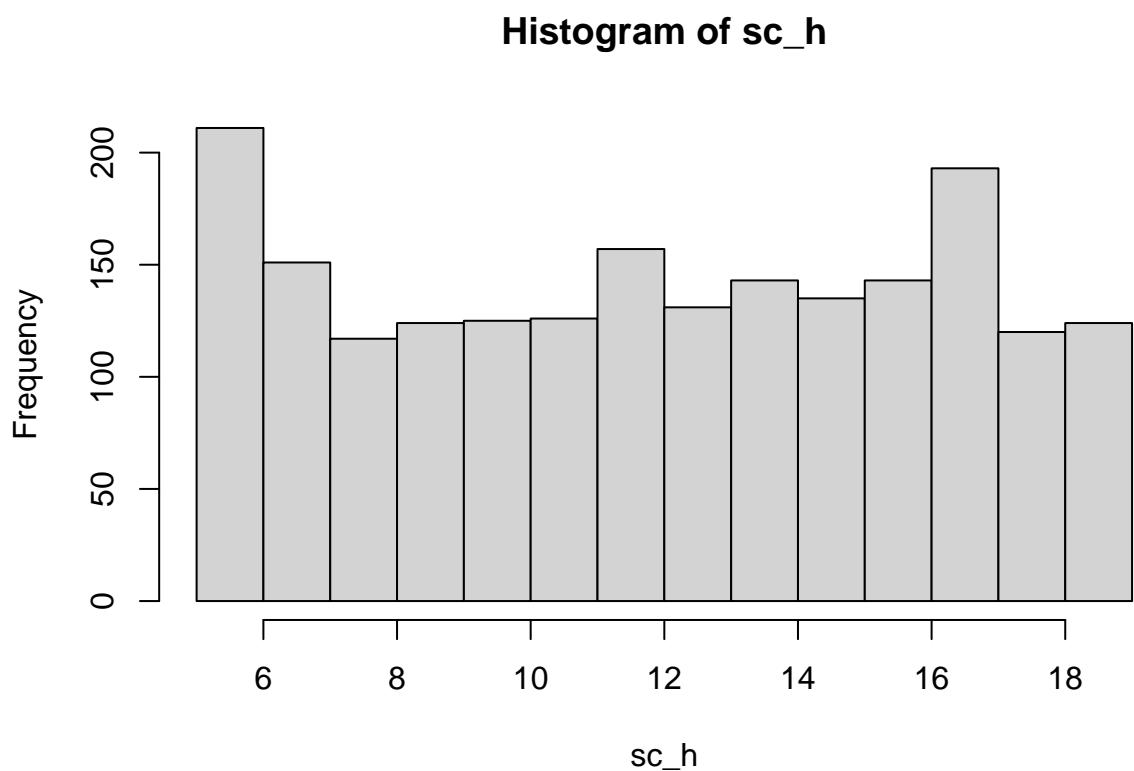


```
boxplot(df$price_range, main="Boxplot of ram")
```

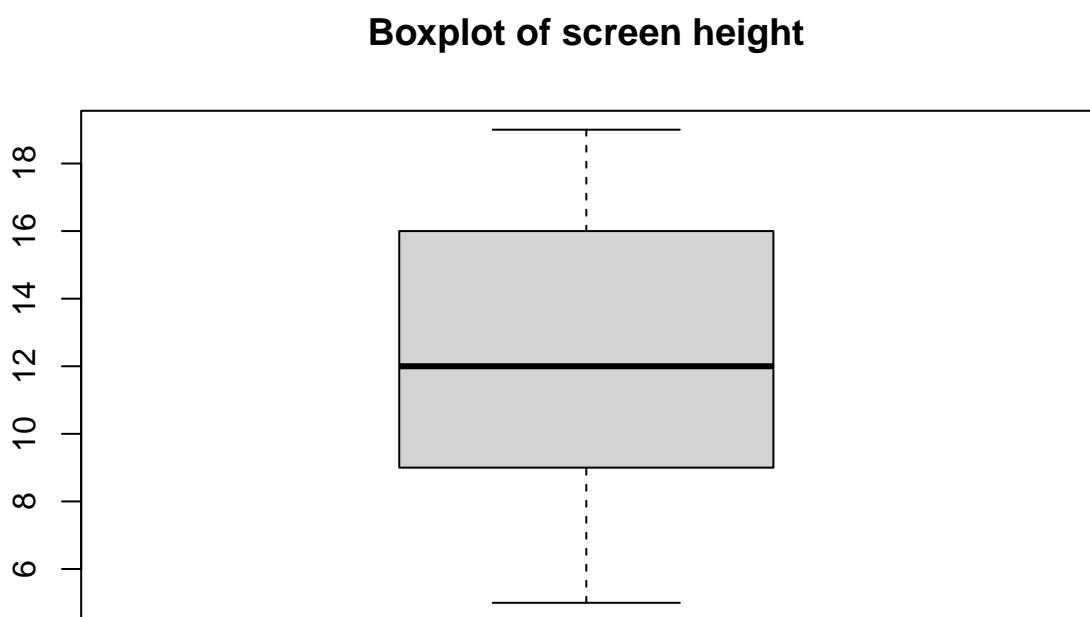


The frequency grows and from 500 remains balanced between 250 and 300.

```
var <- df$sc_h  
hist(var, xlab = "sc_h", main = "Histogram of sc_h")
```



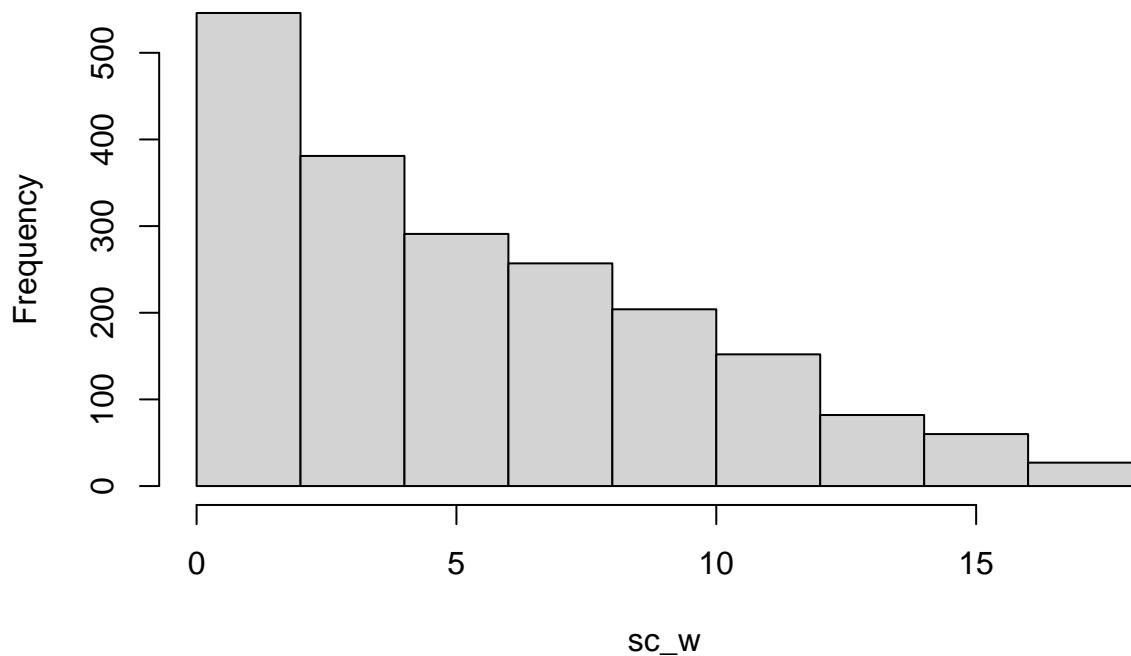
```
boxplot(df$sc_h, main="Boxplot of screen height")
```



The values remain between 8 and 16, with the highest values being 12 and 16.

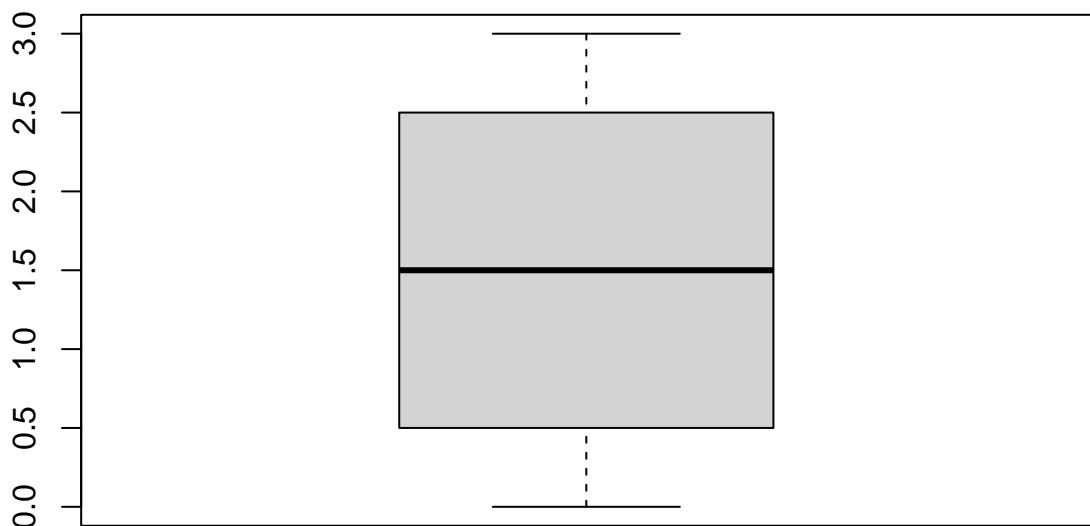
```
var <- df$sc_w  
hist(var, xlab = "sc_w", main = "Histogram of sc_w")
```

Histogram of sc_w



```
boxplot(df$price_range, main="Boxplot of sc_w")
```

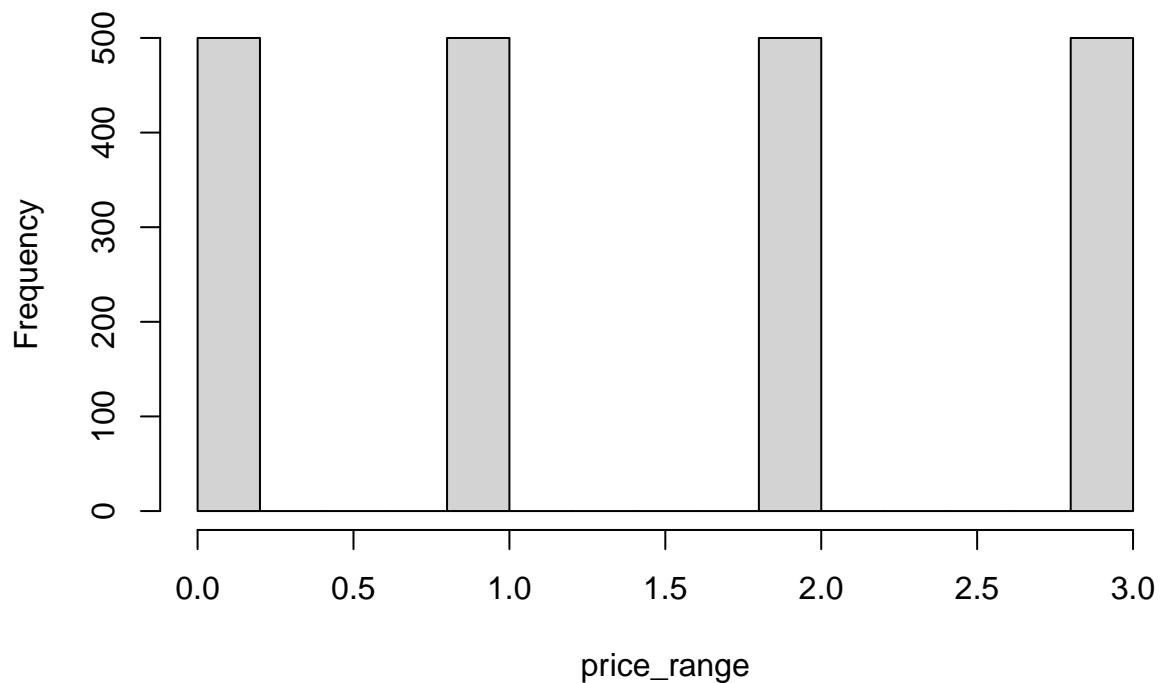
Boxplot of sc_w



The screen width frequency decreases and the higher values are between 0 and 5.

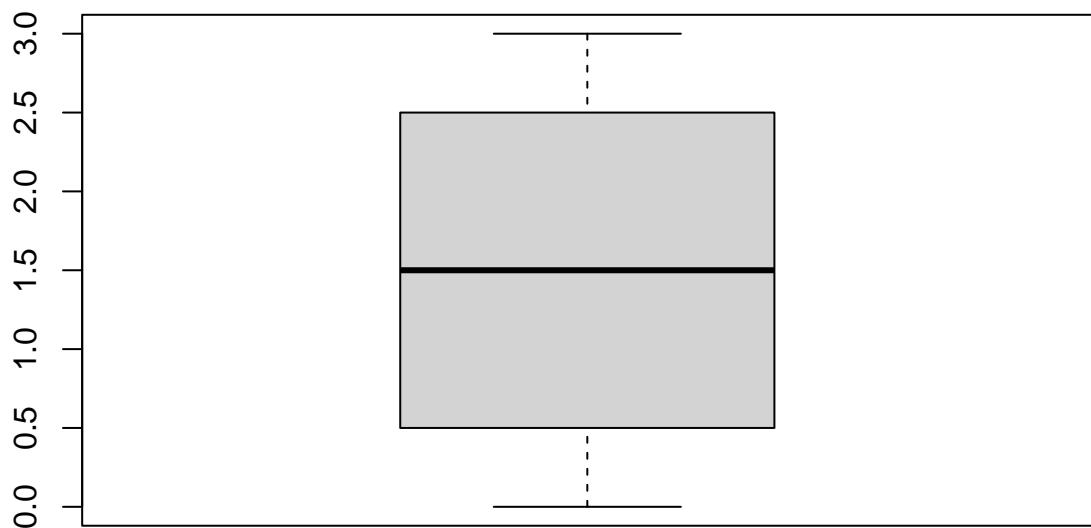
```
var <- df$price_range  
hist(var, xlab = "price_range", main = "Histogram of price_range")
```

Histogram of price_range



```
boxplot(df$price_range, main="Boxplot of price range")
```

Boxplot of price range



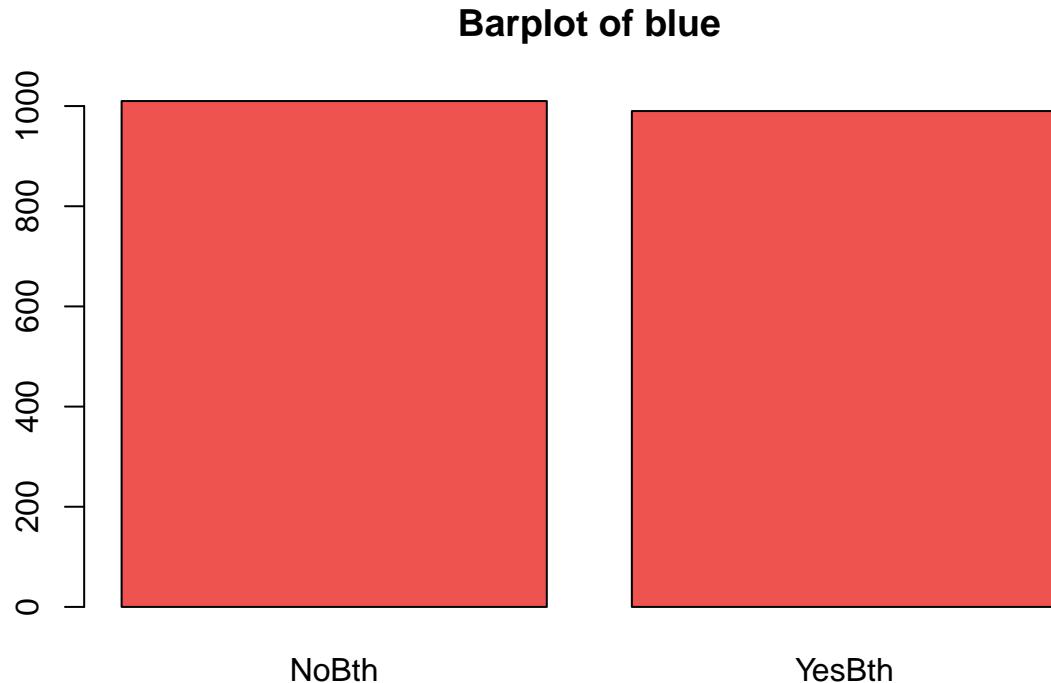
The price range distribution has very big gaps between the values.

2.2.3 Binary variables

```
df$blue <- factor(df$blue, labels = c("NoBth", "YesBth"))
summary(df$blue)
```

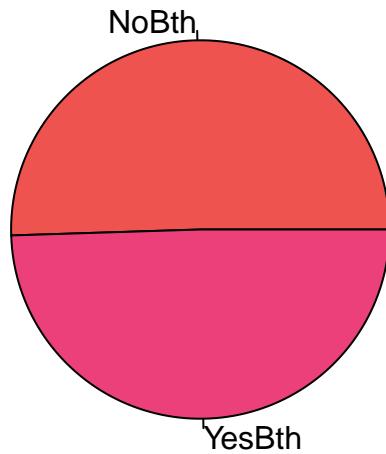
```
##  NoBth YesBth
## 1010    990
```

```
var <- df$blue
barplot(sort(table(var), decreasing = T), col = colors[1], main="Barplot of blue");
```



```
pie(table(var), col = colors, main="Pie of blue")
```

Pie of blue



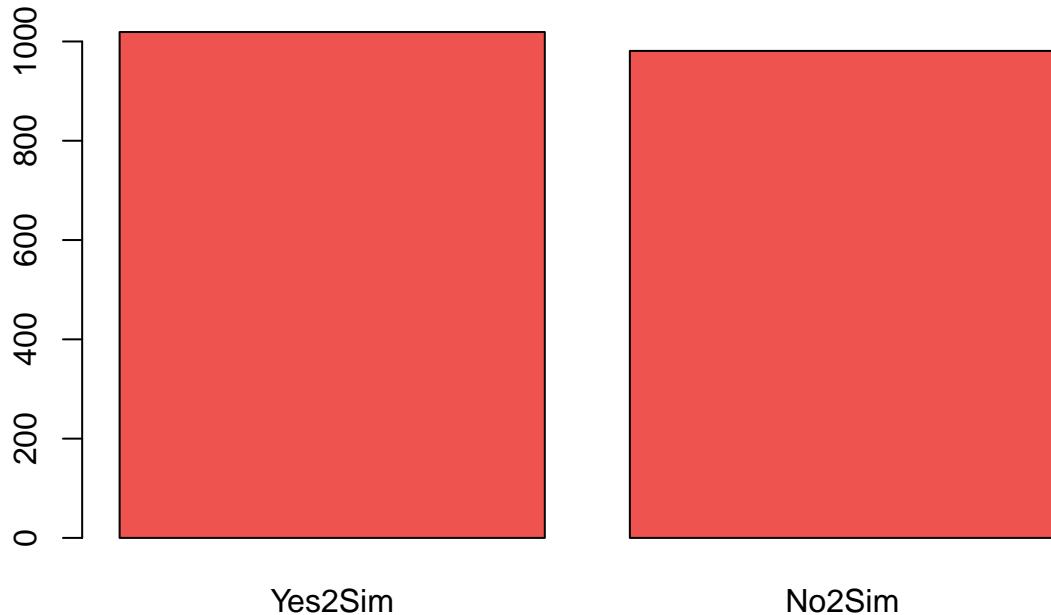
We can see that the number of phones with bluetooth is a little bit lower than the number of phones without it.

```
df$dual_sim <- factor(df$dual_sim, labels = c("No2Sim","Yes2Sim"))
summary(df$dual_sim)
```

```
##  No2Sim Yes2Sim
##      981     1019
```

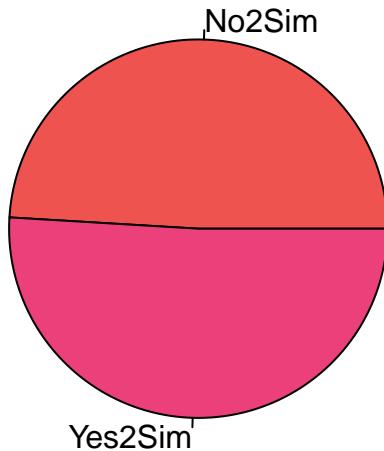
```
var <- df$dual_sim
barplot(sort(table(var)), decreasing = T), col = colors[1], main="Barplot of dual_sim");
```

Barplot of dual_sim



```
pie(table(var), col = colors, main="Pie of dual_sim")
```

Pie of dual_sim



We can also see that phones with dual sim and without are very balanced.

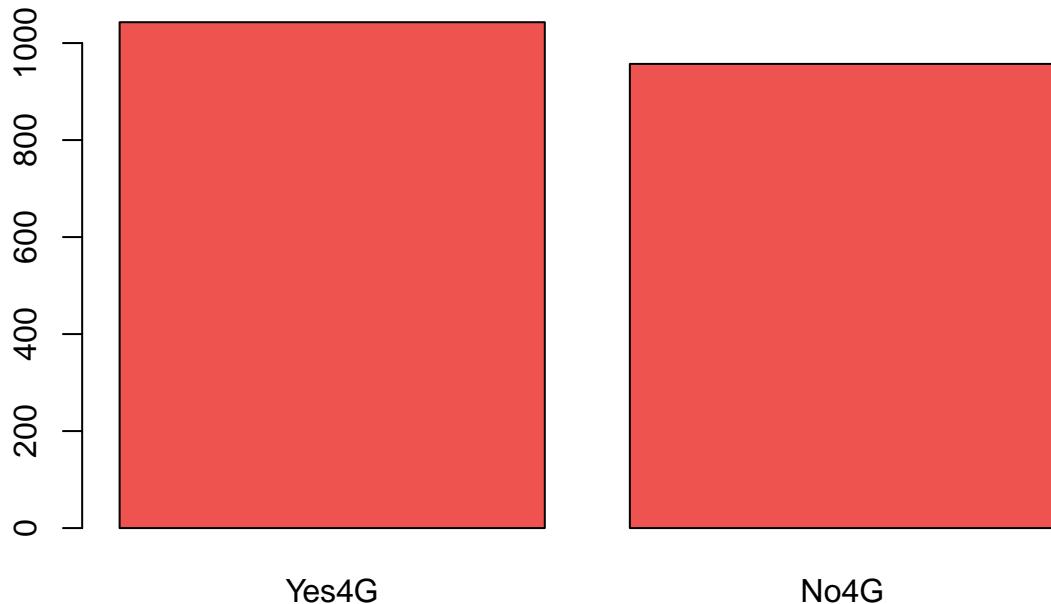
```
df$four_g <- factor(df$four_g, labels = c("No4G", "Yes4G"))
summary(df$four_g)
```

```
##  No4G  Yes4G
```

```
##   957 1043
```

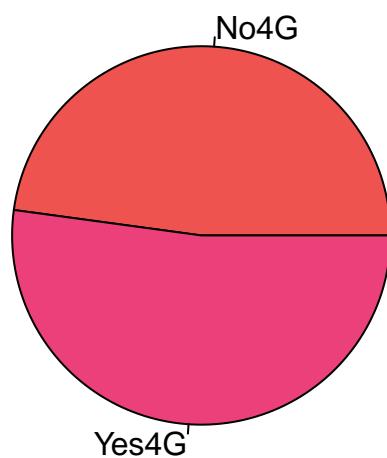
```
var <- df$four_g  
barplot(sort(table(var), decreasing = T), col = colors[1], main="Barplot of four_g");
```

Barplot of four_g



```
pie(table(var), col = colors, main="Pie of four_g")
```

Pie of four_g

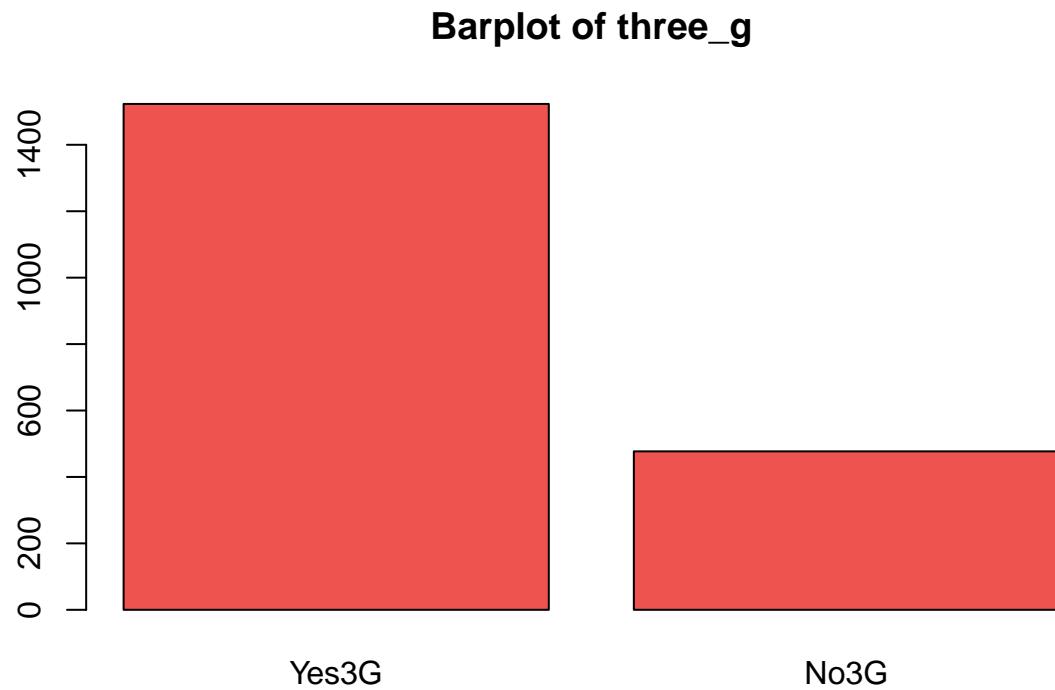


There is no big difference between the amount of phones that have and don't have 4G.

```
df$three_g <- factor(df$three_g, labels = c("No3G", "Yes3G"))
summary(df$three_g)

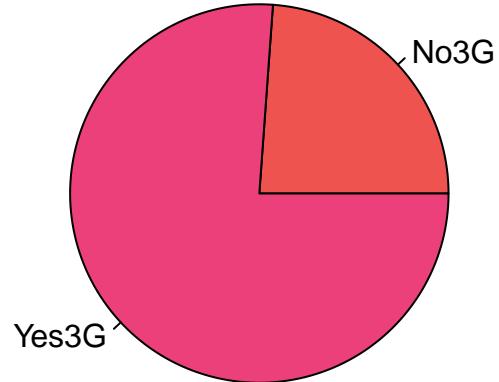
##  No3G Yes3G
##  477 1523

var <- df$three_g
barplot(sort(table(var)), decreasing = T), col = colors[1], main="Barplot of three_g");
```



```
pie(table(var), col = colors, main="Pie of three_g")
```

Pie of three_g



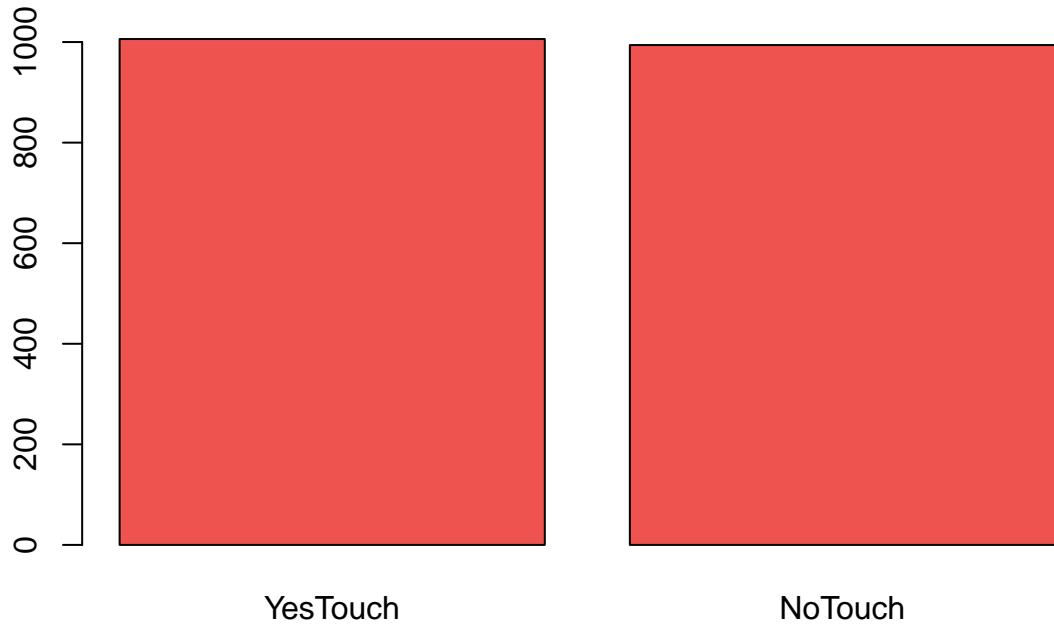
We can see a great difference between phones that have and don't have 3G, we have a difference of more than 1000 units.

```
df$touch_screen <- factor(df$touch_screen, labels = c("NoTouch", "YesTouch"))
summary(df$touch_screen)
```

```
##  NoTouch YesTouch
##      994     1006
```

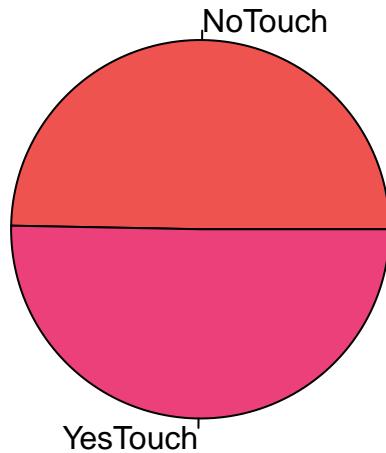
```
var <- df$touch_screen
barplot(sort(table(var)), decreasing = T), col = colors[1], main="Barplot of touch_screen");
```

Barplot of touch_screen



```
pie(table(var), col = colors, main="Pie of touch_screen")
```

Pie of touch_screen



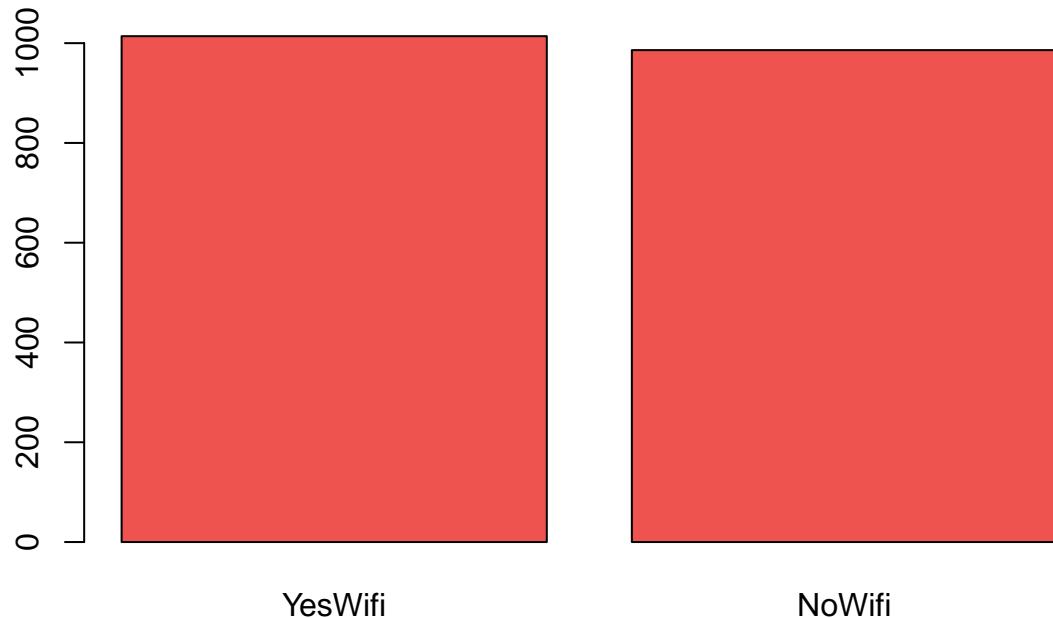
Same happens with the touch screen. The numbers of phones with touch screen are slightly higher than the ones that don't.

```
df$wifi <- factor(df$wifi, labels = c("NoWifi","YesWifi"))
summary(df$wifi)
```

```
##  NoWifi YesWifi  
##      986     1014
```

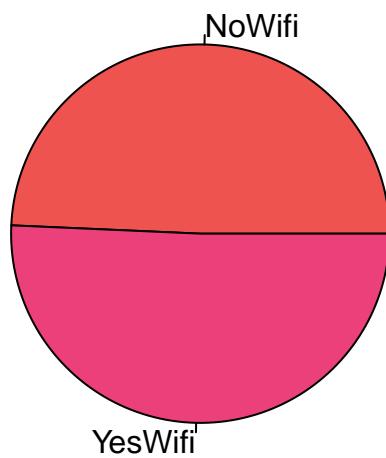
```
var <- df$wifi  
barplot(sort(table(var), decreasing = T), col = colors[1], main="Barplot of wifi");
```

Barplot of wifi



```
pie(table(var), col = colors, main="Pie of wifi")
```

Pie of wifi



We can observe that the number of phones with Wifi is a little bit higher than the ones who don't.

3 Data preprocessing

Before starting to analyze and extract information from the data, we have to prepare it in order to make a correct and clean study. We need to convert the erroneous values into valid data that makes sense.

In our dataset, we happen to have some 0 values in front camera megapixels, primary camera megapixels, pixel resolution height and screen width. We will suppose that the mobiles with 0 mp in the front camera or in the main camera are mobiles that don't have a camera.

Since the observations with a 0 value in the pixel resolution height are so little we will drop those rows. What we will do to correct the screen width values is the KNN method, so those mobile phones that have a 0 screen width are going to have a valid value based on the values of the rest of the data.

Once we have treated those values, our data should not have any more errors or missing data. Now that we cleaned the data, in order to meet project requirements, we have to transform some of the numeric variables into categorical variables.

Currently, our data has 14 numeric variables and 6 binary variables. We decided to put together the screen height and the screen width (`sc_h + sc_w`) into one categorical variable, Screen Dimension (`sc_d`), and divide the screen dimension by groups (small, medium, big). We decided to do the same for the pixel resolution (`px_h + px_w`), Pixel Resolution (`px_r`) and divide it by groups as well (small, medium, big). We chose to group those variables because they are directly related. Then, we decided to group the primary camera megapixels (`pc`) and the front camera megapixels (`fc`) according to quality (low, medium, high) depending on whether it has more or less megapixels. And lastly, we decided to group the battery power (`battery_power`) according to duration (short, medium, long) depending on whether it has more or less power. We chose to make those last 3 variables categorical because they have more market interest than the rest of the numeric variables.

After these transformations, we should get 5 categorical variables and 7 numeric variables, and all the data should be ready to start extracting information.

3.1 Binary variables

```
var_bin

## [1] "blue"           "dual_sim"        "four_g"          "three_g"         "touch_screen"
## [6] "wifi"
```

The binary variables have already been factored while doing the univariate descriptive statistics of raw variables, but we will keep the code.

```
df$blue <- factor(df$blue, labels = c("NoBth", "YesBth"))

df$dual_sim <- factor(df$dual_sim, labels = c("No2Sim", "Yes2Sim"))

df$four_g <- factor(df$four_g, labels = c("No4G", "Yes4G"))

df$three_g <- factor(df$three_g, labels = c("No3G", "Yes3G"))

df$touch_screen <- factor(df$touch_screen, labels = c("NoTouch", "YesTouch"))

df$wifi <- factor(df$wifi, labels = c("NoWifi", "YesWifi"))
```

3.2 Numerical Variables

```
var_num

## [1] "battery_power"  "clock_speed"    "int_memory"     "m_dep"
## [5] "mobile_wt"      "n_cores"       "talk_time"
```

3.2.1 Battery Power

```
summary(df$battery_power)

##      Min. 1st Qu. Median    Mean 3rd Qu.    Max.
## 501.0   851.8 1226.0 1238.5 1615.2 1998.0

batNa<-which( c(is.na(df$battery_power), df$battery_power <=0, df$battery_power > 4000)) #best mobiles !
batNa #doesn't have NA

## integer(0)

var_out<-calcQ(df$battery_power)
batOut<-which((df$battery_power>var_out$souts) | (df$battery_power<var_out$souti))
batOut #doesn't have severe outliers

## integer(0)
```

3.2.2 Clock Speed

```
summary(df$clock_speed)

##      Min. 1st Qu. Median    Mean 3rd Qu.    Max.
## 0.500   0.700  1.500  1.522  2.200  3.000

clkNa<-which( c(is.na(df$clock_speed), df$clock_speed <=0, df$clock_speed > 4))
clkNa #doesn't have NA

## integer(0)

var_out<-calcQ(df$clock_speed)
clkOut<-which((df$clock_speed>var_out$souts) | (df$clock_speed<var_out$souti))
clkOut #doesn't have severe outliers

## integer(0)
```

3.2.3 Internal Memory

```
summary(df$int_memory)

##      Min. 1st Qu. Median    Mean 3rd Qu.    Max.
## 2.00   16.00  32.00  32.05  48.00  64.00

memNa<-which( c(is.na(df$int_memory), df$int_memory <=0, df$int_memory > 128))
memNa #doesn't have NA

## integer(0)

var_out<-calcQ(df$int_memory)
memOut<-which((df$int_memory>var_out$souts) | (df$int_memory<var_out$souti))
memOut #doesn't have severe outliers

## integer(0)
```

3.2.4 Mobile Depth

```

summary(df$m_dep)

##      Min. 1st Qu. Median    Mean 3rd Qu.    Max.
## 0.1000  0.2000  0.5000  0.5018  0.8000  1.0000

depNa<-which( c(is.na(df$m_dep), df$m_dep <=0, df$m_dep > 1))
depNa #doesn't have NA

## integer(0)

var_out<-calcQ(df$m_dep)
depOut<-which((df$m_dep>var_out$souts) | (df$m_dep<var_out$souti))
depOut #doesn't have severe outliers

## integer(0)

```

3.2.5 Mobile Weigth

```

summary(df$mobile_wt)

##      Min. 1st Qu. Median    Mean 3rd Qu.    Max.
## 80.0   109.0  141.0  140.2  170.0  200.0

weigthsNa<-which( c(is.na(df$mobile_wt), df$mobile_wt <=0, df$mobile_wt > 200))
weigthsNa #doesn't have NA

## integer(0)

var_out<-calcQ(df$mobile_wt)
weigthsOut<-which((df$mobile_wt>var_out$souts) | (df$mobile_wt<var_out$souti))
weigthsOut #doesn't have severe outliers

## integer(0)

```

3.2.6 Number of Cores

```

summary(df$n_cores)

##      Min. 1st Qu. Median    Mean 3rd Qu.    Max.
## 1.000  3.000  4.000  4.521  7.000  8.000

coresNa<-which( c(is.na(df$n_cores), df$n_cores <=0, df$n_cores > 8))
coresNa #has one NA

## integer(0)

var_out<-calcQ(df$n_cores)
coresOut<-which((df$n_cores>var_out$souts) | (df$n_cores<var_out$souti))
coresOut #doesn't have severe outliers

## integer(0)

```

3.2.7 Talk Time

```

summary(df$talk_time)

##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.
##      2.00    6.00   11.00   11.01   16.00   20.00

talkNa<-which( c(is.na(df$talk_time), df$talk_time <=0, df$talk_time > 20))
talkNa #has one NA

## integer(0)

var_out<-calcQ(df$talk_time)
talkOut<-which((df$talk_time>var_out$souts) | (df$talk_time<var_out$souti))
talkOut #doesn't have severe outliers

## integer(0)

```

3.3 Categorical Variables

```
var_cat
```

```

## [1] "fc"          "pc"          "px_height"    "px_width"    "ram"
## [6] "sc_h"        "sc_w"        "price_range"

```

3.3.1 Treat missing values

#we find missing values in the pixel resolution height and we drop them
length(which(df\$px_height <= 0))

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

```
df<-df[-which(df$px_height <= 0),]
```

#we find missing values in the screen dimensions
length(which(df\$sc_h == 0))

```
## [1] 0
```

```
length(which(df$sc_w <= 1))
```

```
## [1] 389
```

```
df[df$sc_w <= 1,"sc_w"] <- NaN
```

#knn imputation of sc_w NaN values using all the other numerical variables
dfaux<-df[,c(var_num,"fc","px_height", "px_width","ram","sc_h","price_range")]
names(dfaux)

```

## [1] "battery_power" "clock_speed"    "int_memory"    "m_dep"
## [5] "mobile_wt"      "n_cores"       "talk_time"     "fc"
## [9] "px_height"      "px_width"       "ram"          "sc_h"
## [13] "price_range"

```

```

aux1 <- dfaux[!is.na(df$sc_w),]
aux2 <- dfaux[is.na(df$sc_w),]

knn.ing = knn(aux1, aux2, df$sc_w[!is.na(c(df$sc_w))])

df$sc_w[is.na(df$sc_w)] <- as.numeric(as.character(knn.ing))

which(is.na(df$sc_w))

## integer(0)

summary(df$sc_w)

```

```

##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   2.00   4.00   6.00   7.09  10.00  18.00

```

```

summary(df$sc_h)

##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   5.00   9.00  12.00  12.31  16.00  19.00

```

3.3.2 Screen dimension

```

#sc_h * sc_w = screen dimension (cm^2) into factor

df$sc_d <- df$sc_h * df$sc_w
summary(df$sc_d)

##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 10.00  36.00  72.00  93.87 132.00 342.00

df$sc_d <- factor(cut(df$sc_d, breaks = c(0,115,230,345)))
levels(df$sc_d) <- c("small","medium","big")
summary(df$sc_d)

## small medium   big
## 1377    491    130

```

3.3.3 Front camera quality

```

# fc into factor
summary(df$fc)

##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.000  1.000  3.000  4.309  7.000 19.000

length(which(df$fc == 0)) #no tienen camara frontal

## [1] 474

df$fc <- factor(cut(df$fc, breaks = c(-0.1,0,5,10,15,20)))
levels(df$fc) <- c("NoFrontCam","bad","regular","good","excellent")
summary(df$fc)

## NoFrontCam      bad     regular      good   excellent
##           474       875       428       179        42

```

3.3.4 Main camera quality

```
# pc into factor
summary(df$pc)

##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.
##    0.000   5.000 10.000   9.916 15.000 20.000

length(which(df$pc == 0)) #no tienen camara principal

## [1] 101

df$pc <- factor(cut(df$pc, breaks = c(-0.1,0,5,10,15,21)))
levels(df$pc) <- c("NoMainCam", "bad", "regular", "good", "excellent")
summary(df$pc)

## NoMainCam      bad    regular     good excellent
##          101       450       536       449       462
```

3.3.5 Pixel Dimension

```
#sqrt(df$px_height^2 + df$px_width)/sqrt(df$sc_h^2 + df$sc_w^2) = px_r into factor

diag <- (sqrt((df$sc_h^2) + (df$sc_w^2)))
diag <- diag * 0.393700787

df$px_r <- (sqrt(((df$px_height^2) + df$px_width^2)))/ diag
summary(df$px_r)

##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.
##    56.42  174.06 249.15 286.93 357.08 1028.66

df$px_r <- factor(cut(df$px_r, breaks = c(0,150,300,450,1500)))

levels(df$px_r) <- c("bad", "regular", "good", "excellent")
summary(df$px_r)

##      bad    regular     good excellent
##      359       903       463       273
```

3.3.6 Ram

```
#ram into factor
summary(df$ram)

##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.
##    256     1206    2146    2124    3064    3998

df$ram <- factor(cut(df$ram, breaks = c(255,2048,3072,4096)))
levels(df$ram) <- c("low", "mid", "high")
summary(df$ram)

## low mid high
## 947 555 496
```

3.3.7 Price range

```
#price range into factor
summary(df$price_range)

##      Min. 1st Qu. Median    Mean 3rd Qu.    Max.
## 0.000   0.250  1.500  1.499  2.000  3.000

df$price_range <- factor(df$price_range,labels = c("low","medium","high","very high"))
summary(df$price_range)

##      low     medium      high very high
## 500       499       500       499
```

4 Basic statistical descriptive analysis

... {Claudia}

5 PCA Analysis

In this section we will perform the Principal Component Analysis of the preprocessed dataset. Principal Component Analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data. To do this, we will only use the principal components that add up to 80% of the total variance, and ignore the rest.

5.1 Scree plot

First, we will generate a scree plot where we will see the contribution of the different factors to the total variance. This way, we will select the factors that contribute to approximately 80% of the variance and discard the rest.

The first step is to select the numerical variables of our dataset, as PCA only works with numerical variables.

```
#Set a list of numerical variables (with no missing values)
numeriques<-which(sapply(df,is.numeric))
dcon<-df[,numeriques]
```

Then, we use the prcomp r function to get the principal components of our dataset. We use scale=TRUE in order to make all the variables range between 0 and 1, and thus make them comparable. From this components, we will be able to generate the scree plot and to select the factors we will use.

```
# PRINCIPAL COMPONENT ANALYSIS OF dcon
pc1 <- prcomp(dcon, scale=TRUE)
print(pc1)
```

```
## Standard deviations (1, ..., p=11):
## [1] 1.2469270 1.1715607 1.0356597 1.0199135 1.0123871 1.0011882 0.9886505
## [8] 0.9791904 0.9574199 0.7747682 0.6923412
##
## Rotation (n x k) = (11 x 11):
##          PC1        PC2        PC3        PC4        PC5
## battery_power  0.01315361 -0.074298168  0.651890204 -0.10206810  0.003166279
## clock_speed    0.03682492 -0.003361396 -0.007639484 -0.16887331 -0.519063879
## int_memory     -0.02163329  0.064748610 -0.010459140 -0.68396753  0.300897754
## m_dep          -0.02726453 -0.132674501  0.424000488 -0.03529877  0.095807912
## mobile_wt      0.03107914 -0.103214636  0.186977827  0.22831710 -0.650115551
## n_cores         -0.03823146  0.035147021 -0.188617778  0.60730812  0.336993831
## px_height       -0.62967984 -0.313129277 -0.028021040 -0.05329146 -0.017020085
## px_width        -0.62412342 -0.326247236 -0.050306096  0.01919844 -0.004671116
## sc_h            -0.32583007  0.618248071  0.066706699 -0.02253173 -0.036802305
## sc_w            -0.32015872  0.610667504  0.136532320  0.07469643 -0.100806455
## talk_time       0.01176459 -0.033945879  0.546031940  0.24851294  0.287751845
##          PC6        PC7        PC8        PC9        PC10
## battery_power   0.24594697  0.208398201  0.3982780037 -0.54114032 -0.040596849
## clock_speed     0.71026012 -0.371012337  0.0278605808  0.22765805 -0.076582807
## int_memory      0.04068235 -0.293184997 -0.4776709520 -0.34129601  0.066887122
## m_dep           -0.39740688 -0.691592554  0.2696051724  0.28724809 -0.004466057
## mobile_wt       -0.34494716 -0.101138912 -0.4755059232 -0.35055941 -0.020256228
## n_cores          0.28230144 -0.460993689 -0.0442892768 -0.42554318 -0.056456912
## px_height        0.02516656  0.048644703 -0.0045692560 -0.01319523 -0.153312672
## px_width         0.04846524  0.021143664 -0.0282581474  0.01828160  0.156342275
## sc_h             -0.09378813 -0.005361925  0.0009331446  0.01223305 -0.684885032
## sc_w             0.01651889 -0.035480122  0.0306816204  0.01150815  0.682430440
## talk_time        0.25440009  0.166784652 -0.5566981909  0.38932662 -0.040852714
##          PC11
## battery_power   0.043496695
## clock_speed     0.006312069
## int_memory      0.011844876
## m_dep           -0.001305280
## mobile_wt       0.003934089
## n_cores         -0.029885964
```

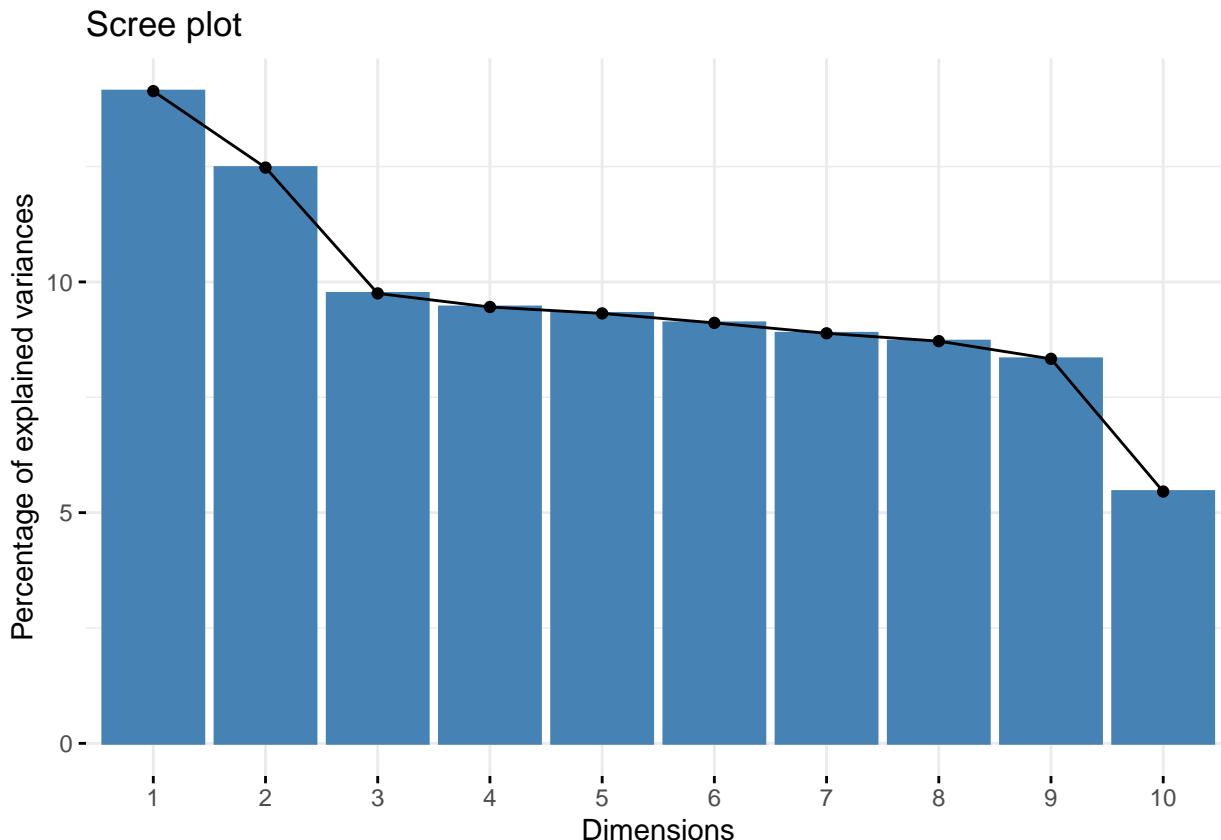
```

## px_height      -0.689085686
## px_width       0.687560914
## sc_h           0.165073392
## sc_w           -0.147933101
## talk_time      -0.017464871

```

Printing pc1, we can see the relations of the eleven principal components to the different variables. We can also see the percentage of total inertia represented in each subspace with the following scree plot.

```
#Which percentage of the total inertia is represented in subspaces?
fviz_eig(pc1)
```



```
#Cummulated Inertia in subspaces, from first principal component to the 11th dimension subspace
percInerAccum<-100*cumsum(pc1$sdev[1:dim(dcon)[2]]^2)/dim(dcon)[2]
percInerAccum
```

```

## [1] 14.13479 26.61256 36.36338 45.81996 55.13749 64.25001 73.13574
## [8] 81.85223 90.18544 95.64240 100.00000

```

It can be seen in the scree plot that with only 8 of the 11 dimensions we keep 80% of the inertia. From the relationships observed in the print of pc1, we will guess names for the different Principal Components. We think that PC1 almost certainly measures screen dimension, as px_height, px_width, sc_h and sc_w have large positive contributions on this component. PC2 can be pixel density, because px_height and px_width have positive contributions and sc_h and sc_w have negative contributions on this second component. The third component PC3 has large negative contributions from battery_power, m_dep and talk_time, so it is probably measuring slimness. The fourth component PC4 has large positive contributions from int_memory, and negative contributions in_ncores, so it could be measuring something like data slowness. The fifth component PC5 has large positive contributions from clock speed and mobile_wt, it could be measuring CPU power. The sixth component PC6 could be measuring antiquity, because it has large positive contributions from m_dep and mobile_wt but large negative contributions from clock_speed and n_cores. PC7 could be measuring portability, as it has large negative contributions from m_dep. Finally, PC8 is probably measuring battery management, because it has large positive contributions from battery power but negative contributions from talk time.

5.2 Factorial map visualisation

```

# Selection of significant dimensions (keep 80% of total inertia)
nd <- 8

# Storage of the eigenvalues, eigenvectors and projections in the nd dimensions
Psi = pc1$x[,1:nd]
dim(Psi)

## [1] 1998     8

Psi[dim(df)[1],]

##          PC1         PC2         PC3         PC4         PC5         PC6         PC7
## 0.6102968 0.9747248 -1.3164341 -0.4011152 -0.6957609 -1.2449485 -2.4021722
##          PC8
## -0.0843061

# STORAGE OF LABELS FOR INDIVIDUALS AND VARIABLES
iden = row.names(dcon)
etiq = names(dcon)
ze = rep(0,length(etiq))
# WE WILL NEED THIS VECTOR AFTERWARDS FOR THE GRAPHICS

```

5.2.1 Individuals projections

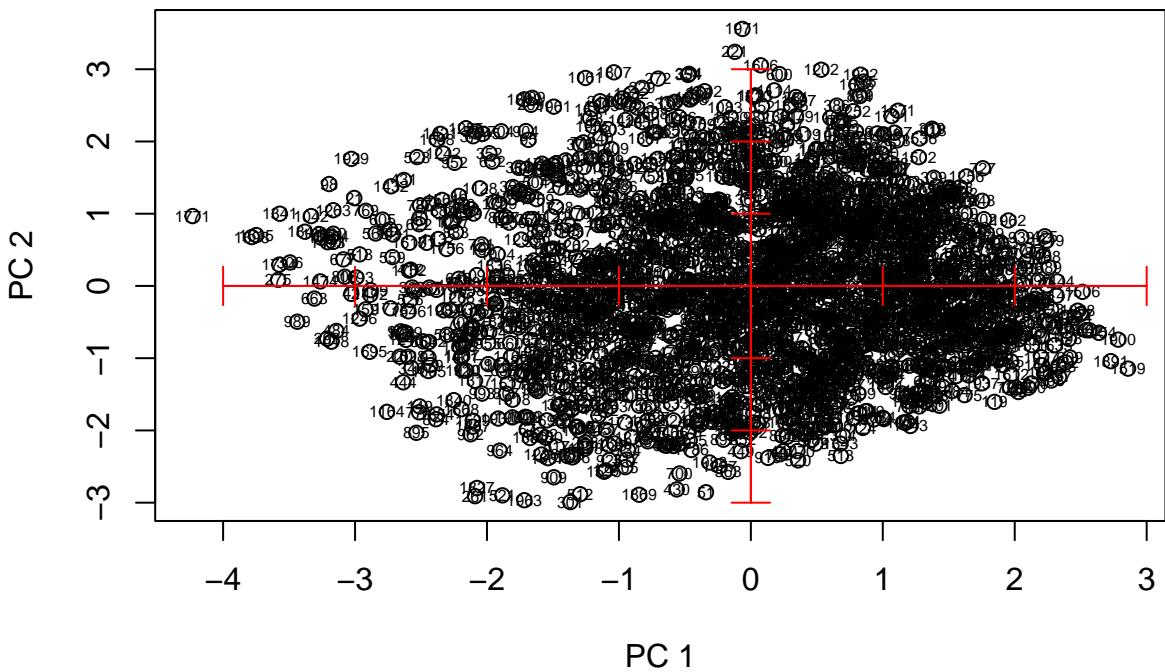
In this section we will analyse the individuals projection plot that we found are useful. All of the plots are found in the annex at the end of the document. After analyzing all the plots we found that the majority of plots have no meaningful information, they are just clouds of points. The only plot where it can be seen something is the plot where axis are PC1 and PC2, screen dimension and pixel density. It makes sense that these two variables are related.

```

eje1<-1
eje2<-2

plot(Psi[,eje1],Psi[,eje2], xlab=paste("PC",toString(1)) , ylab=paste("PC", toString(2)))
text(Psi[,eje1],Psi[,eje2],labels=iden, cex=0.5)
axis(side=1, pos= 0, labels = F, col="red")
axis(side=3, pos= 0, labels = F, col="red")
axis(side=2, pos= 0, labels = F, col="red")
axis(side=4, pos= 0, labels = F, col="red")

```



We can see that the cloud forms a rhomboid. This makes sense because individuals with big screens are not as likely to have small resolution, so the pixel density is bigger too, but with very large screens you would need a lot of resolution to have big pixel density, so the values get smaller. On the small screen side, we can see that they are more likely to have bigger pixel density. Finally, in middle values, pixel density can be big or small.

5.2.2 Common projection of numerical variables

Now, we will perform the projection of our numerical variables. Then, we will analyse these plots and talk about the ones that we find interesting.

```
#Projection of variables
Phi = cor(dcon,Psi)
Phi
```

	PC1	PC2	PC3	PC4	PC5
## battery_power	0.01640159	-0.087044811	0.675136424	-0.10410063	0.003205500
## clock_speed	0.04591799	-0.003938079	-0.007911906	-0.17223617	-0.525493569
## int_memory	-0.02697513	0.075856925	-0.010832110	-0.69758774	0.304625000
## m_dep	-0.03399688	-0.155436228	0.439120225	-0.03600170	0.096994693
## mobile_wt	0.03875342	-0.120922209	0.193645404	0.23286370	-0.658168589
## n_cores	-0.04767184	0.041176868	-0.195343835	0.61940177	0.341168203
## px_height	-0.78516477	-0.366849948	-0.029020262	-0.05435268	-0.017230914
## px_width	-0.77823632	-0.382218433	-0.052099998	0.01958075	-0.004728977
## sc_h	-0.40628630	0.724315129	0.069085441	-0.02298042	-0.037258178
## sc_w	-0.39921454	0.715434035	0.141401023	0.07618390	-0.102055153
## talk_time	0.01466959	-0.039769657	0.565503284	0.25346171	0.291316252
	PC6	PC7	PC8		
## battery_power	0.24623920	0.20603299	0.3899900078		
## clock_speed	0.71110404	-0.36680154	0.0272808139		
## int_memory	0.04073068	-0.28985750	-0.4677308225		
## m_dep	-0.39787907	-0.68374334	0.2639948034		
## mobile_wt	-0.34535702	-0.09999104	-0.4656108471		
## n_cores	0.28263686	-0.45576165	-0.0433676358		
## px_height	0.02519647	0.04809261	-0.0044741718		
## px_width	0.04852283	0.02090369	-0.0276701074		

```

## sc_h      -0.09389957 -0.00530107  0.0009137262
## sc_w      0.01653851 -0.03507744  0.0300431490
## talk_time 0.25470237  0.16489173 -0.5451135382

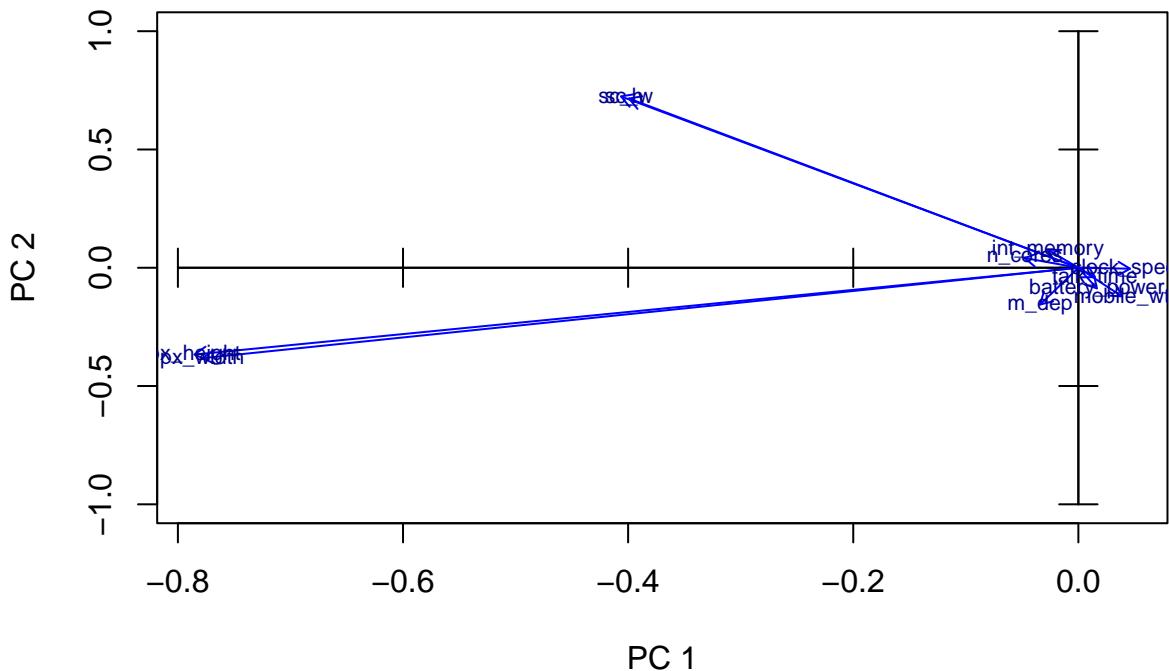
```

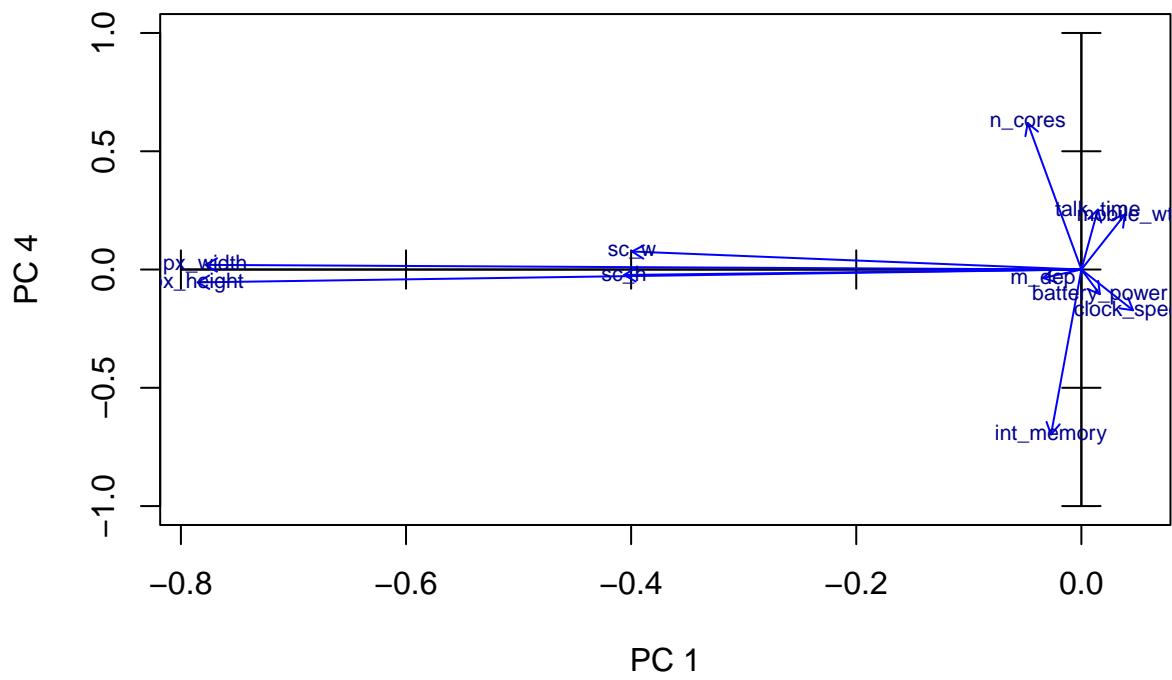
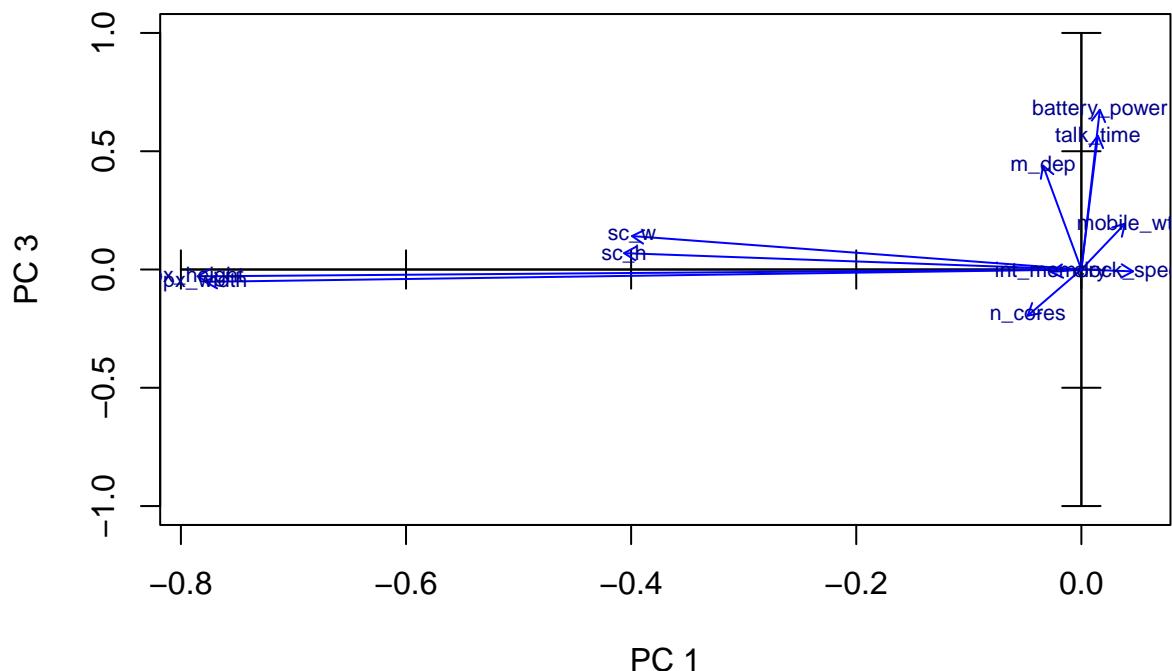
```

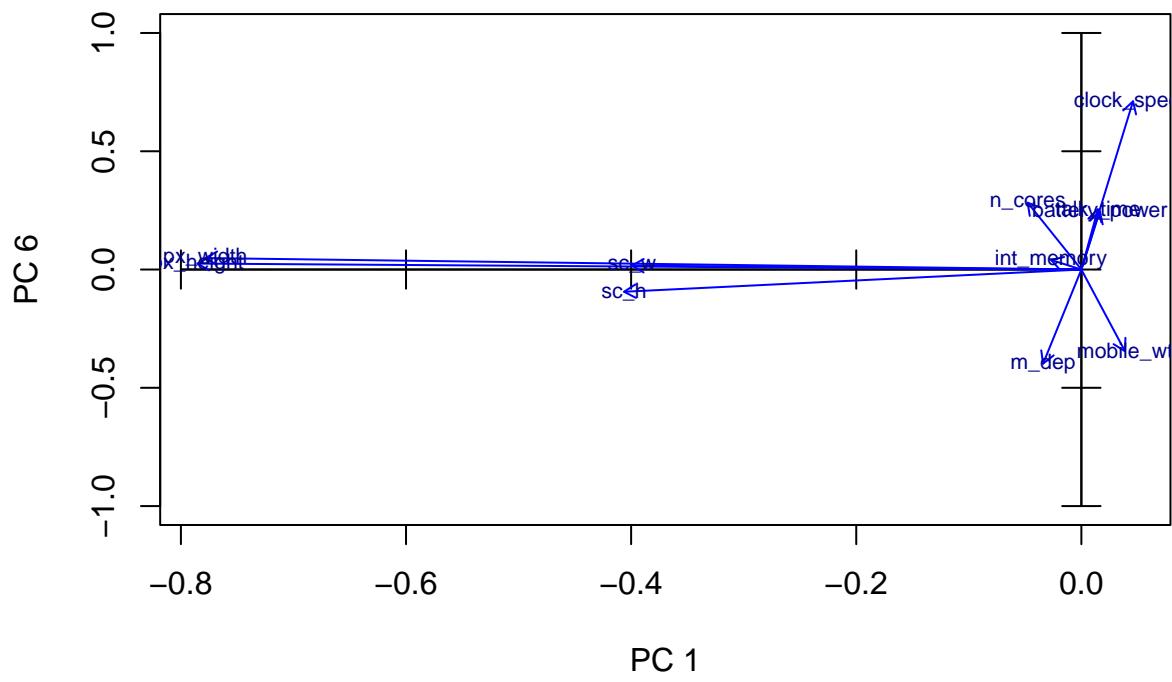
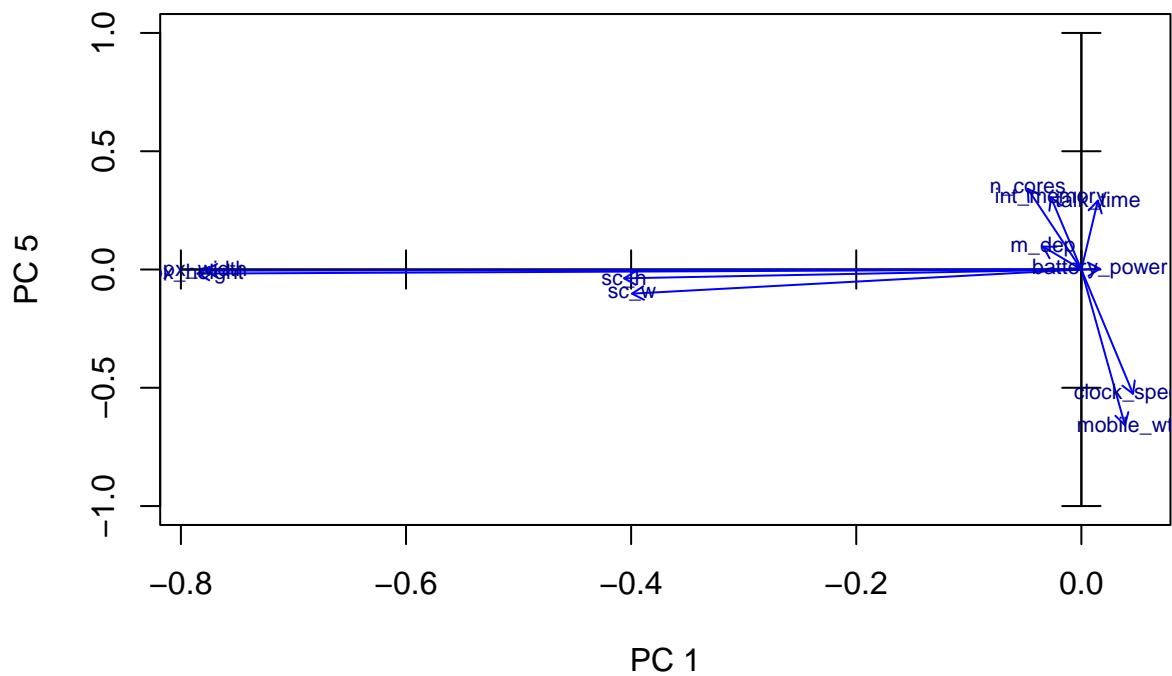
#select your axis
for(i in 1:2) {
  for (j in (i+1):8) {
    X<-Phi[,i]
    Y<-Phi[,j]

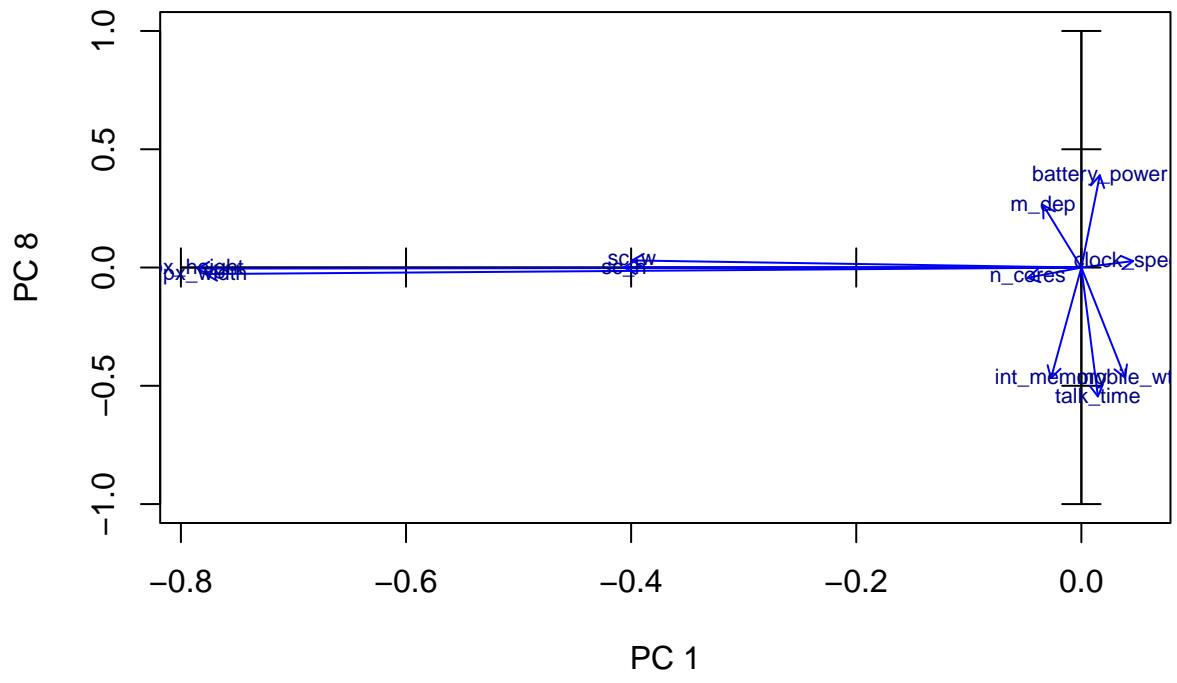
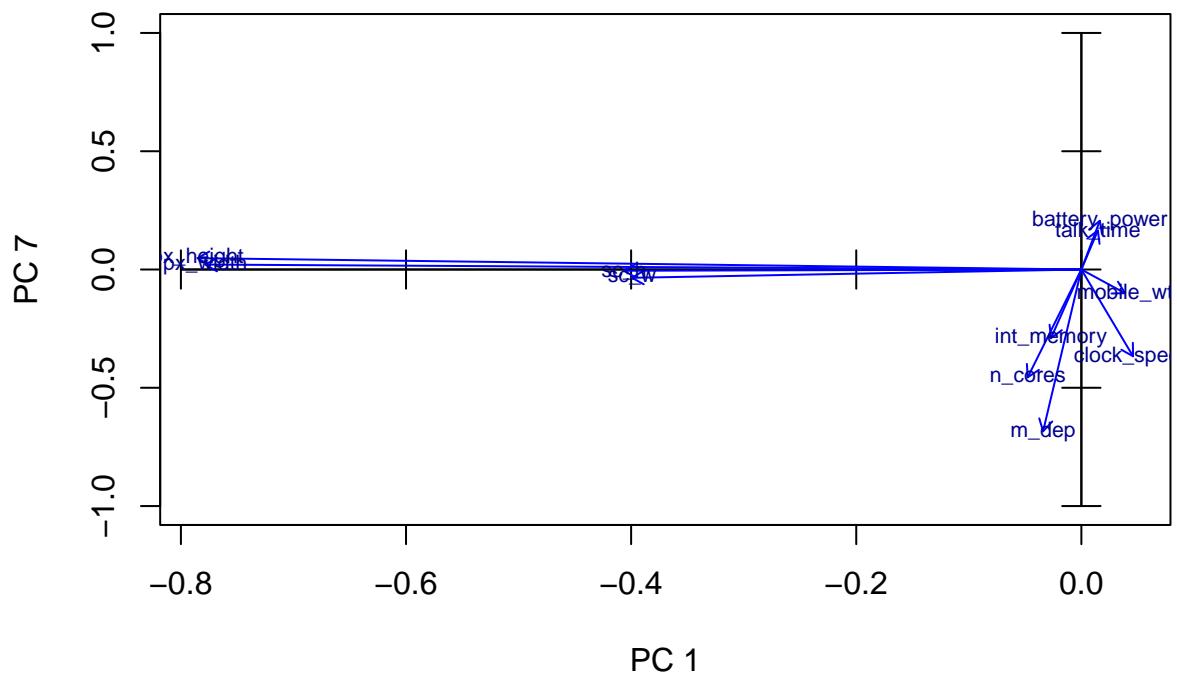
    #zoom
    plot(X, Y, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)), type="n", xlim=c(min(X,0),max(X,0)), ylim=c(min(Y,0),max(Y,0)))
    axis(side=1, pos= 0, labels = F)
    axis(side=3, pos= 0, labels = F)
    axis(side=2, pos= 0, labels = F)
    axis(side=4, pos= 0, labels = F)
    arrows(ze, ze, X, Y, length = 0.07,col="blue")
    text(X,Y,labels=etiq,col="darkblue", cex=0.7)
  }
}

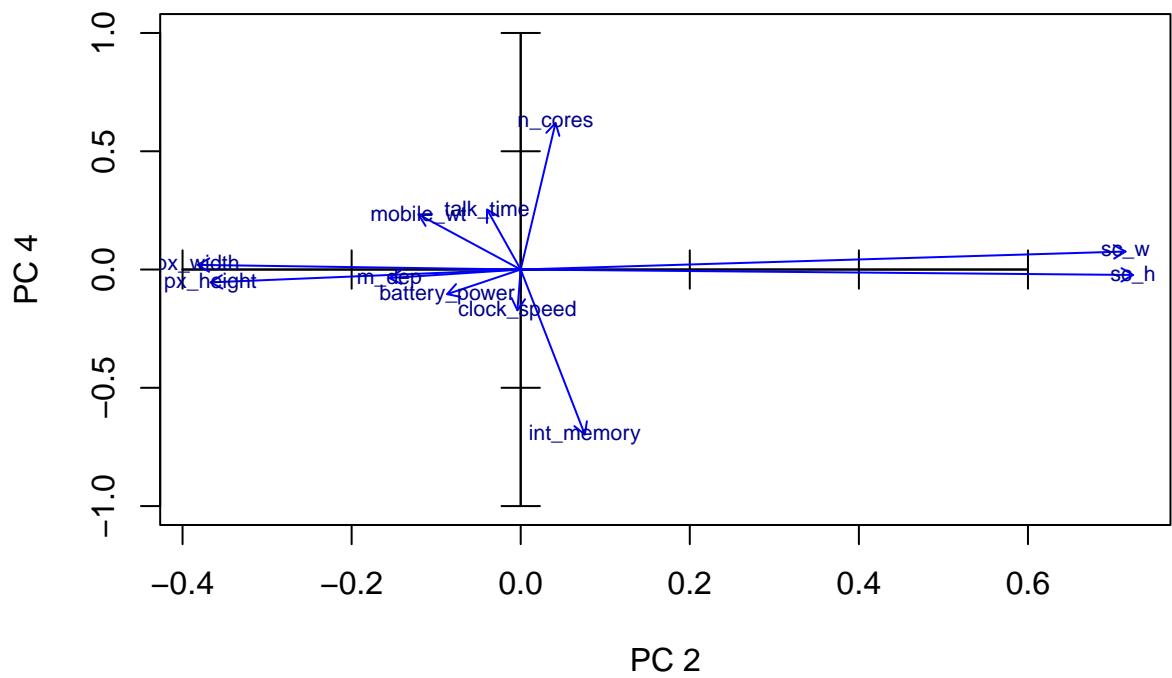
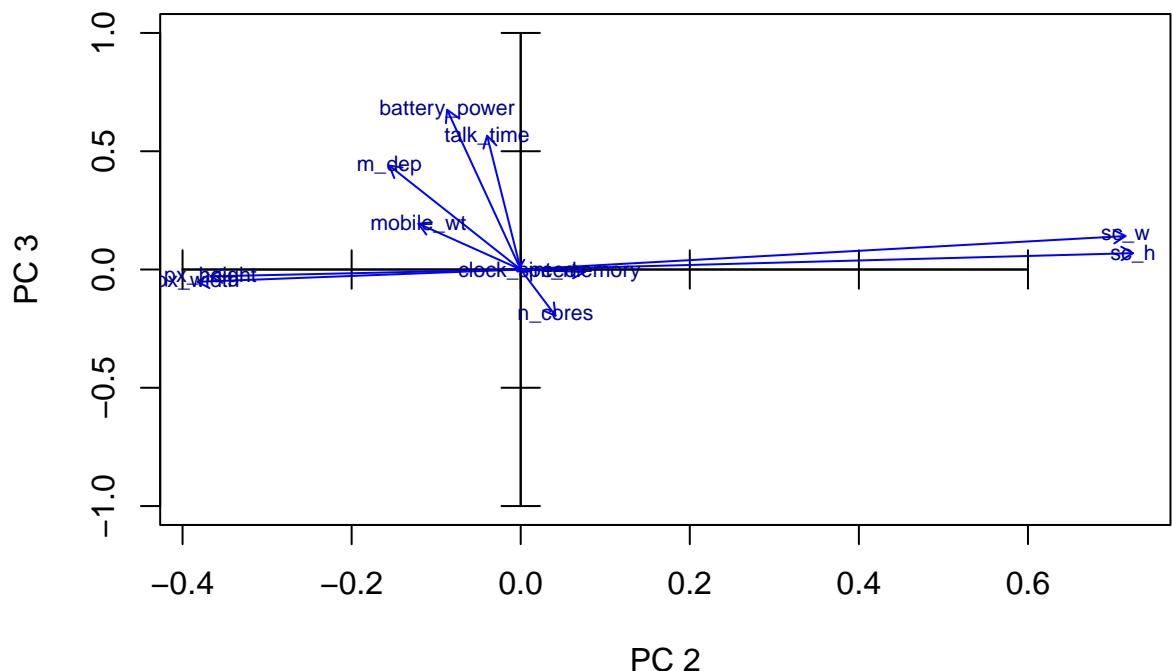
```

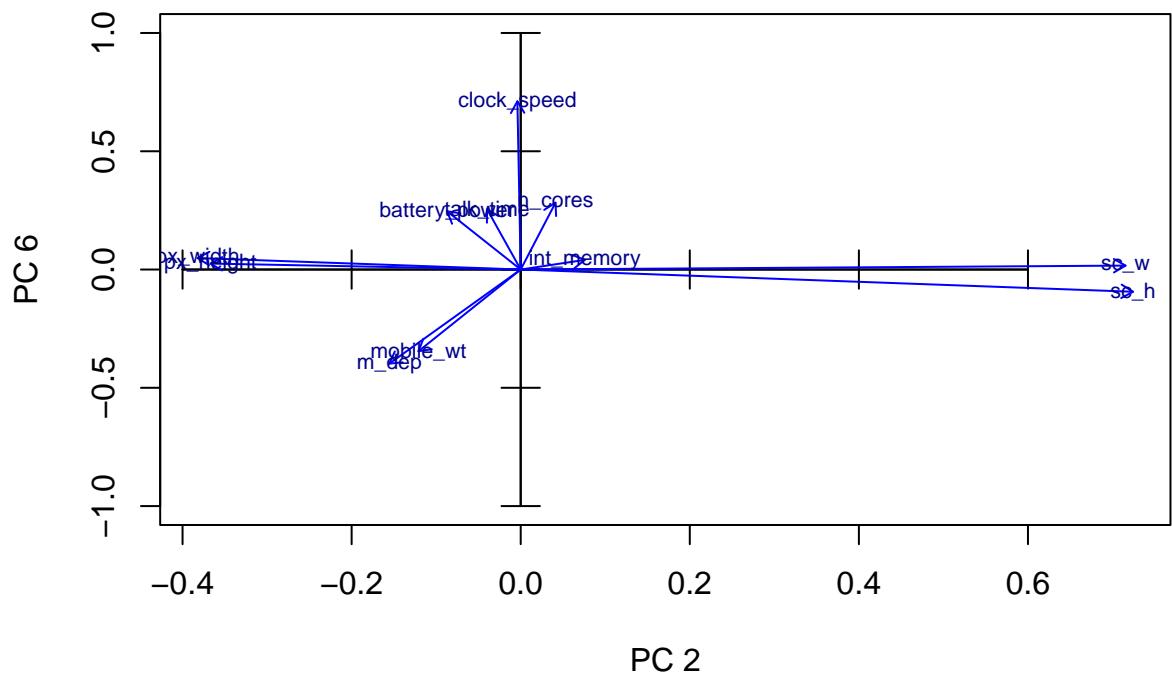
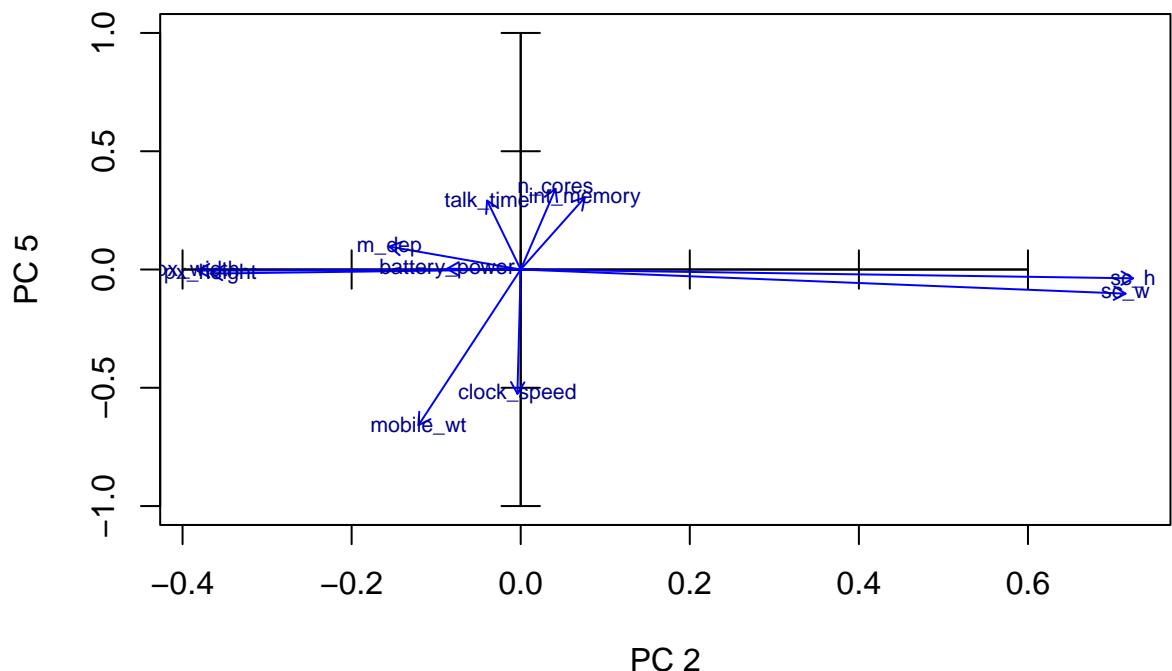


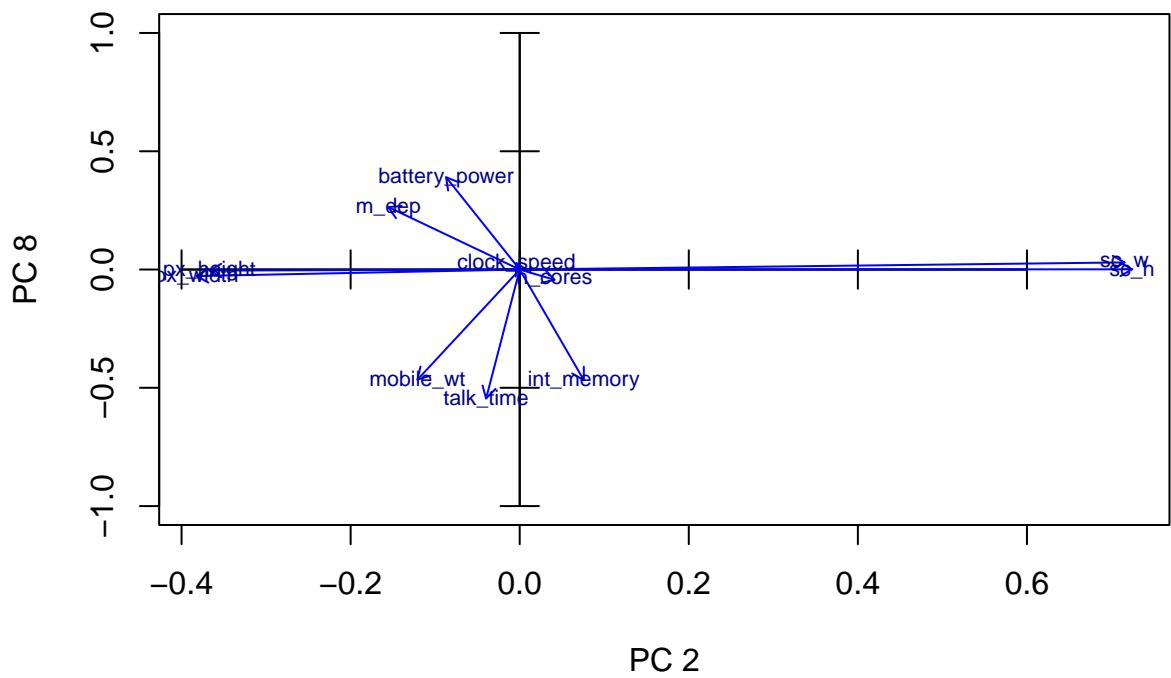
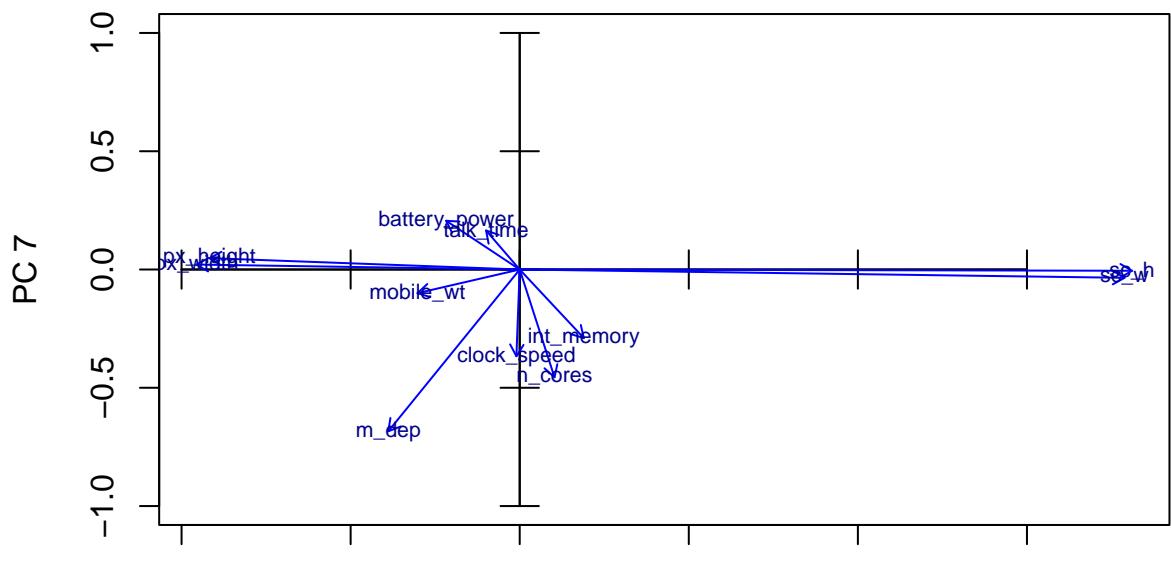












```
X<-Phi[,3]
Y<-Phi[,7]
```

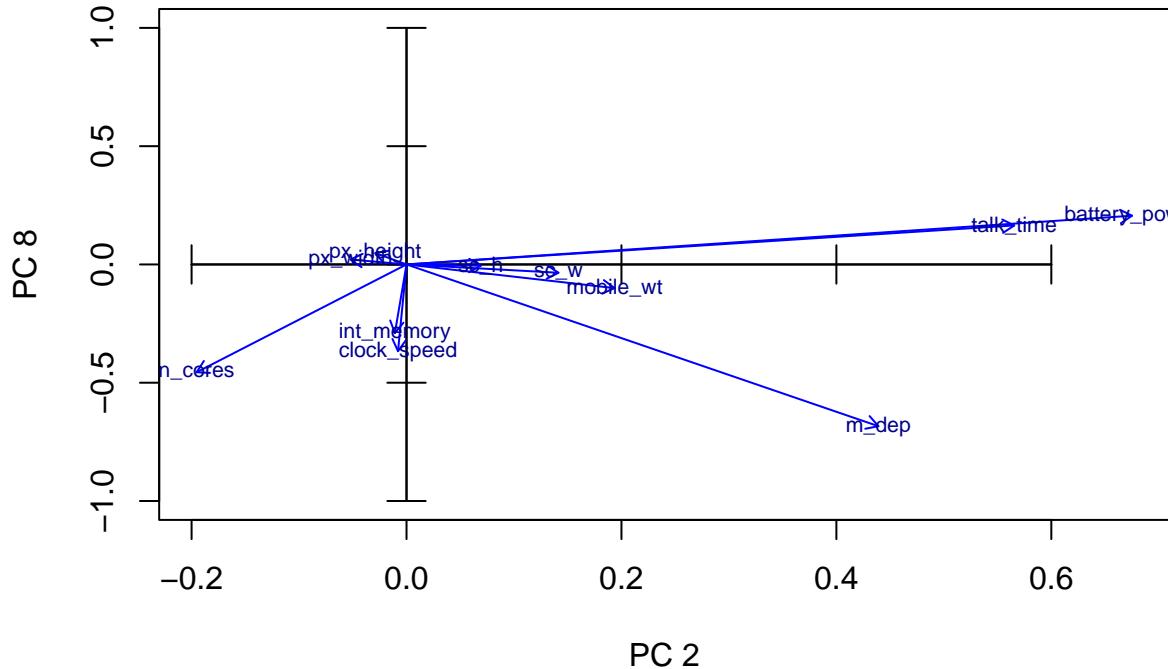
```
#zoom
```

```
plot(X, Y, xlab=paste("PC",toString(i)), ylab=paste("PC", toString(j)), type="n", xlim=c(min(X,0),max(X,0)), ylim=c(min(Y,0),max(Y,0)))
axis(side=1, pos= 0, labels = F)
axis(side=3, pos= 0, labels = F)
axis(side=2, pos= 0, labels = F)
```

```

axis(side=4, pos= 0, labels = F)
arrows(ze, ze, X, Y, length = 0.07,col="blue")
text(X,Y,labels=etiq,col="darkblue", cex=0.7)

```



As we can see, in all the plots that have PC1 in the x axis, variables px_width, px_height, sc_w and sc_h are very close to this axis, and are really large compared to the other variables, indicating that they are very related to screen dimension, which makes sense. A similar phenomena can be seen in plots with PC2 in the x axis, where px_height and px_width are large, really close to the axis and positive, whereas sc_h and sc_w are large, really close to the axis and negative, which again makes sense being PC2 pixel density. Moreover, in the plot with axis PC1 and PC2, these four variables are closer to PC1 than to PC2, but they make more of a diagonal between the two principal components, meaning that screen dimension and pixel density are related.

Another interesting plot is for example PC3 with PC7, where we can see that m_dep is almost 45 degrees negatively in both slimness and portability, meaning that phones that are fatter are less portable and quite obviously less slim.

5.2.3 Projection of qualitative variables

In this section, we will do the same individual projection plots as we did before, but now we will colour the dots based on the dataset's categorical variables. We will comment on some plots, the one's that we have found that have no valuable information to analyse can be found in the annex.

```

for(i in 2) {
  for (j in c(1,3,4,5,6,7,8)) {
    varcat<-df$sc_d
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)), ylab=paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
    legend("bottomleft",levels(varcat),pch=16,col=c(1:3), cex=0.6)

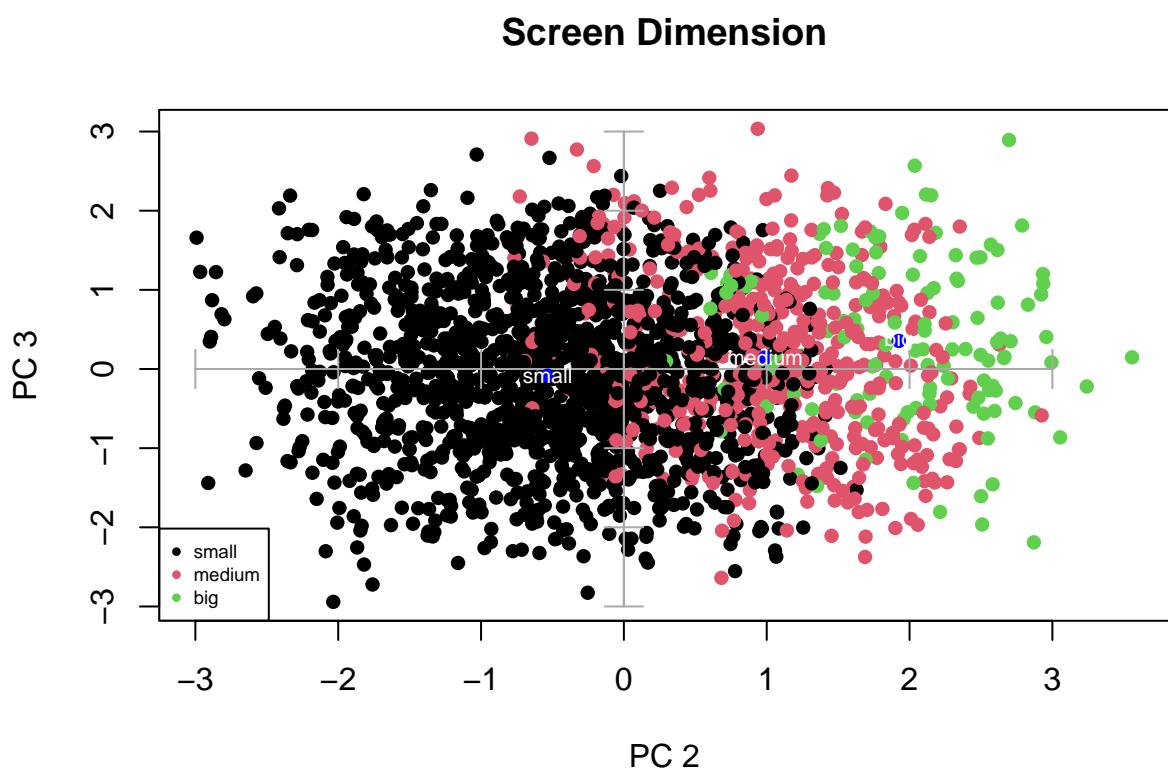
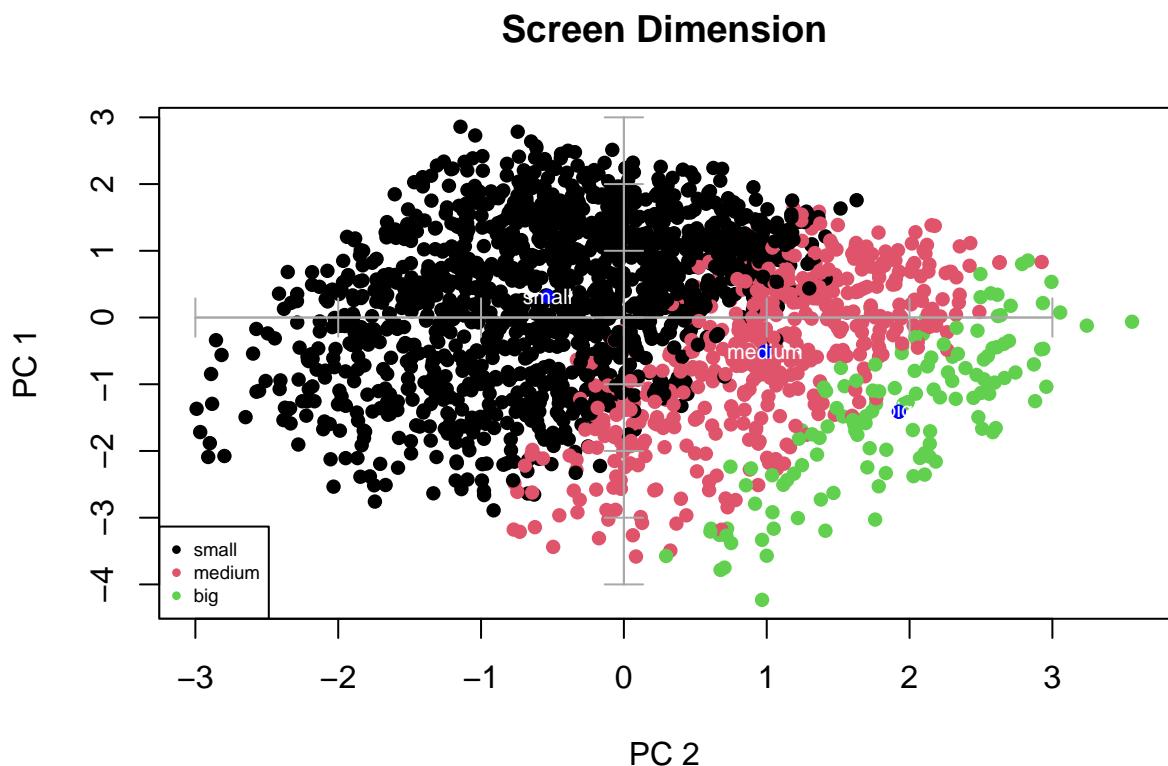
    #select your qualitative variable
    fdic1 = tapply(Psi[,i],varcat,mean)
    fdic2 = tapply(Psi[,j],varcat,mean)
  }
}

```

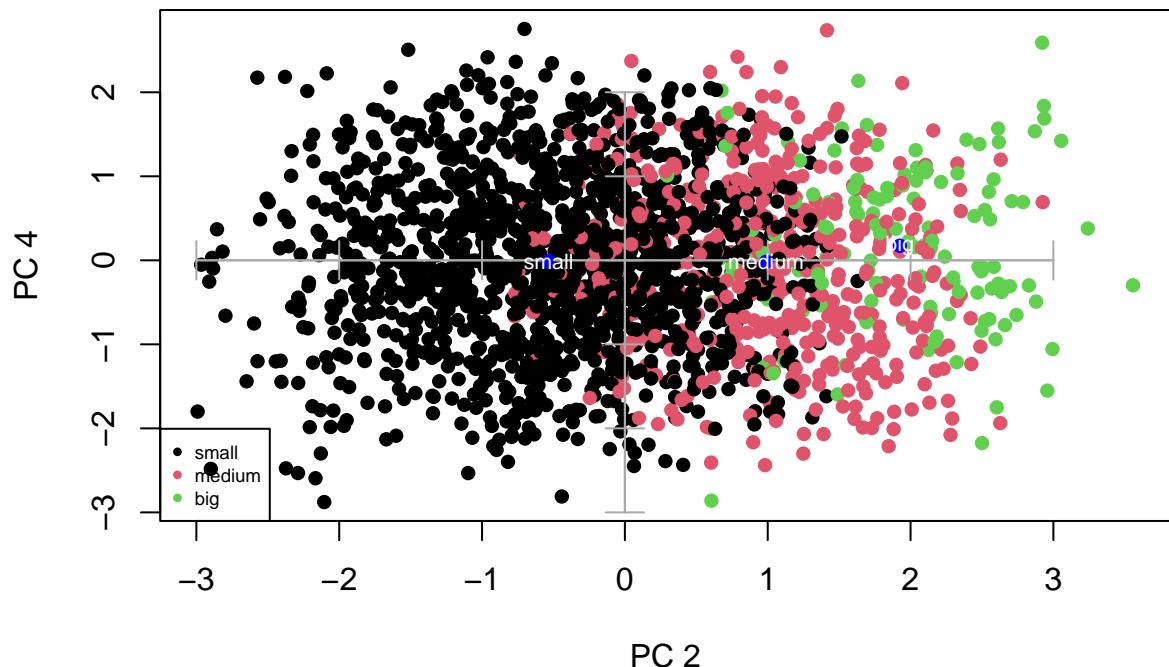
```

    points(fdic1,fdic2,pch=16,col="blue")
    text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
}
}

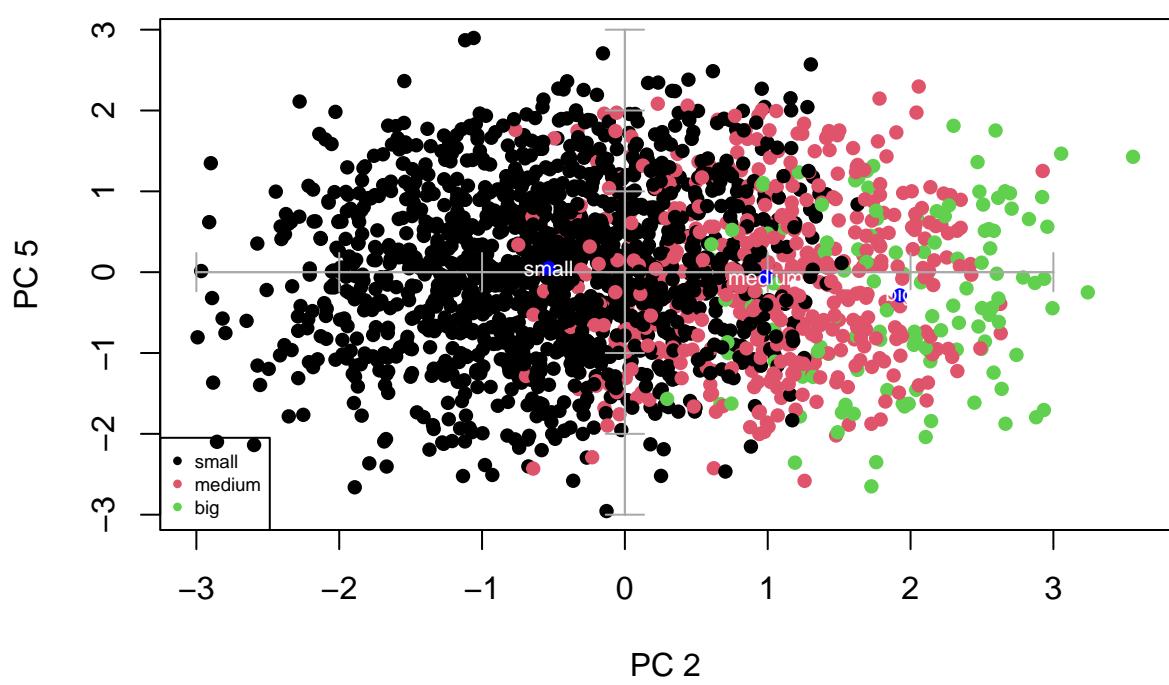
```



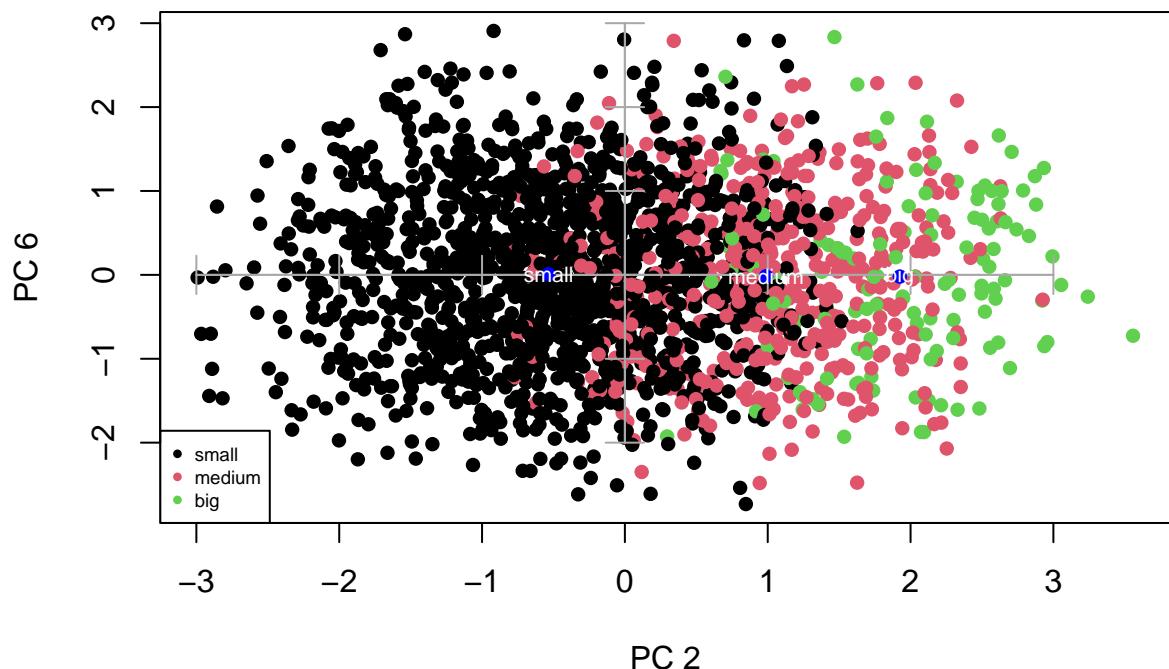
Screen Dimension



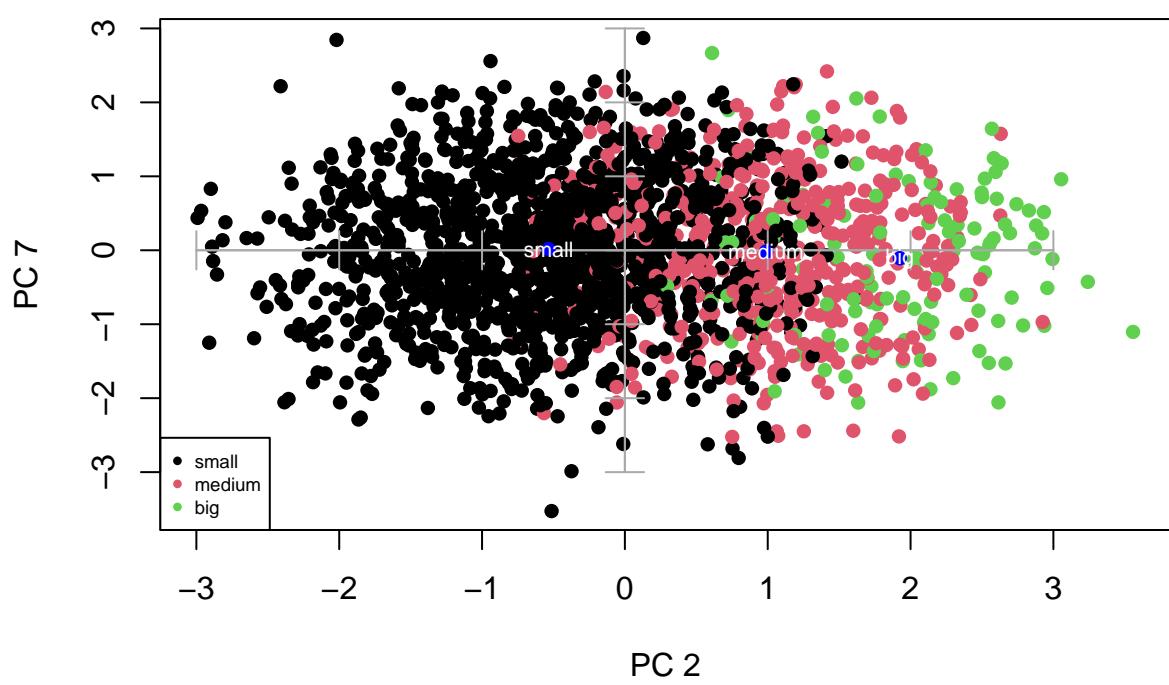
Screen Dimension

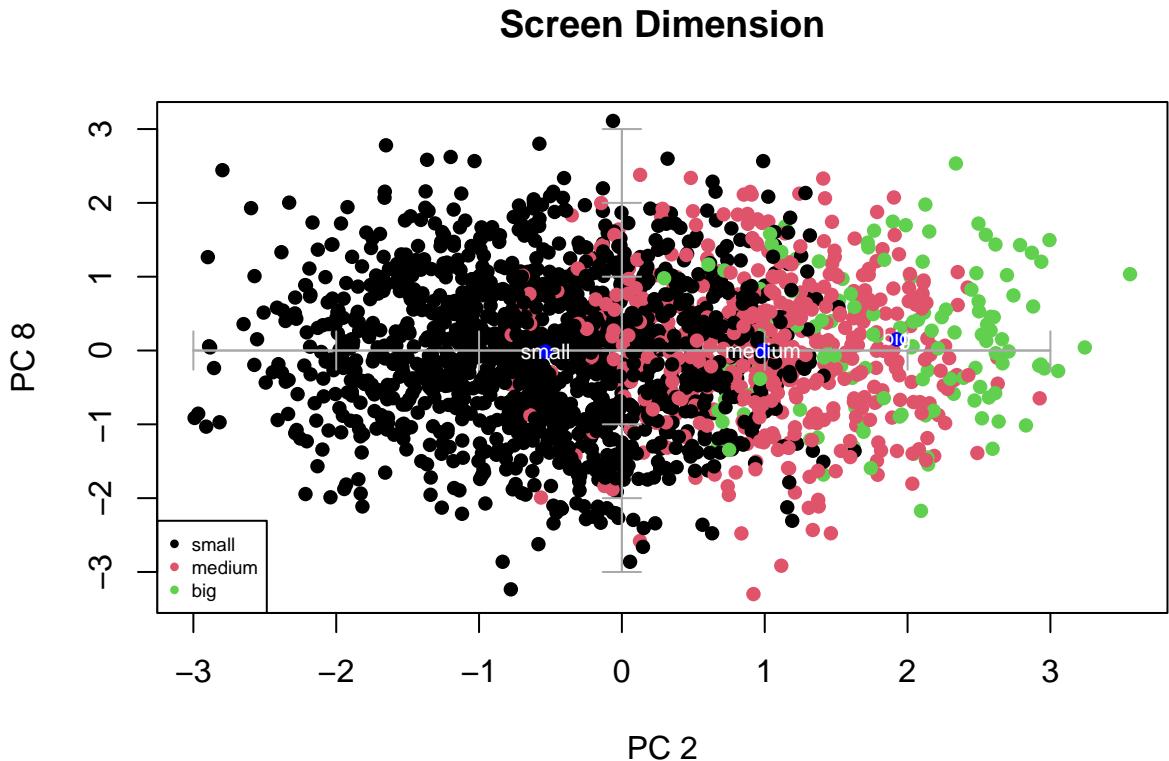


Screen Dimension



Screen Dimension



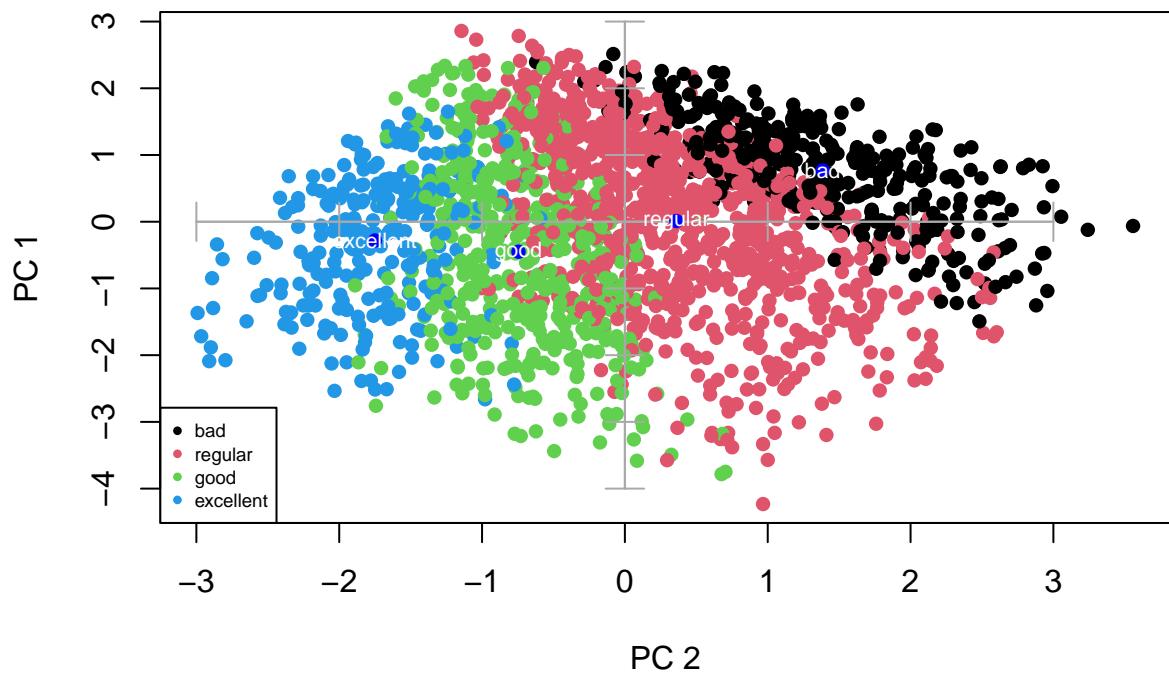


```

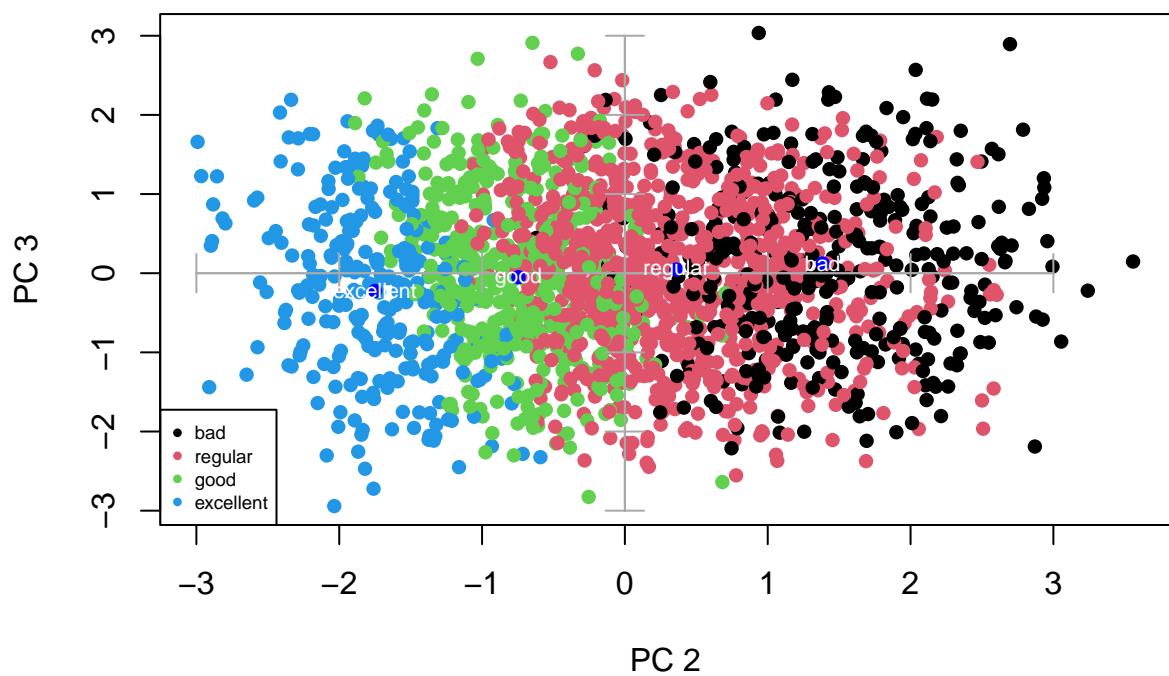
for(i in 2) {
  for (j in c(1,3,4,5,6,7,8)) {
    varcat<-df$px_r
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab= paste("PC",toString(i)), ylab= paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
    legend("bottomleft",levels(varcat),pch=16,col=c(1:4), cex=0.6)

    #select your qualitative variable
    fdic1 = tapply(Psi[,i],varcat,mean)
    fdic2 = tapply(Psi[,j],varcat,mean)
    points(fdic1,fdic2,pch=16,col="blue")
    text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
  }
}
  
```

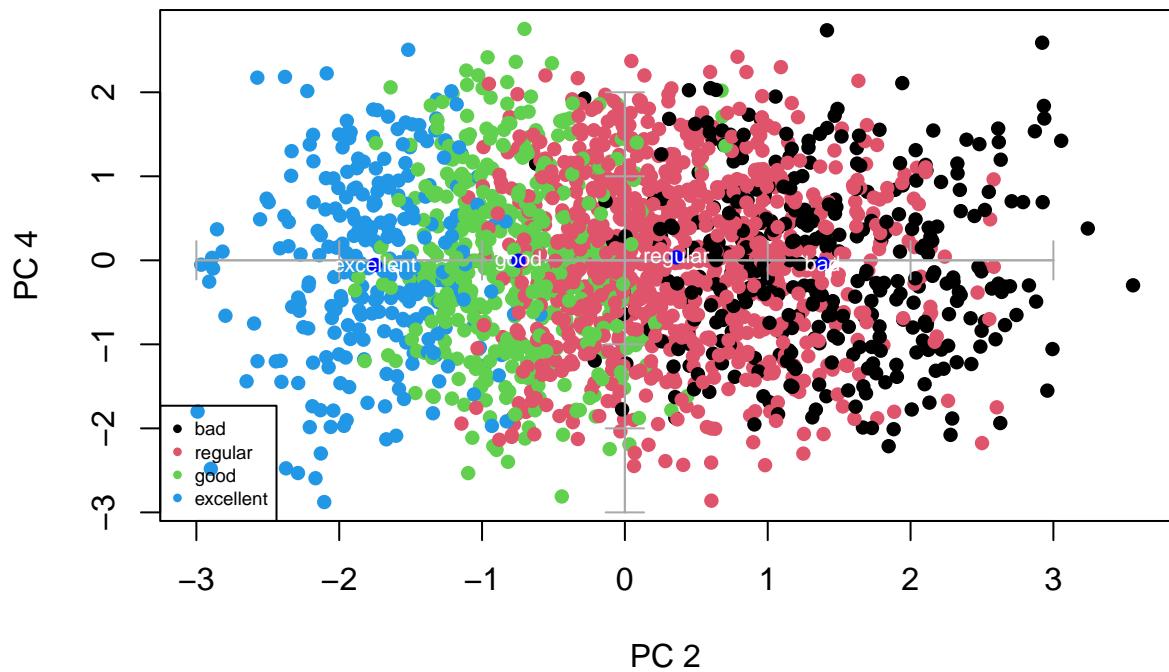
Pixel Resolution



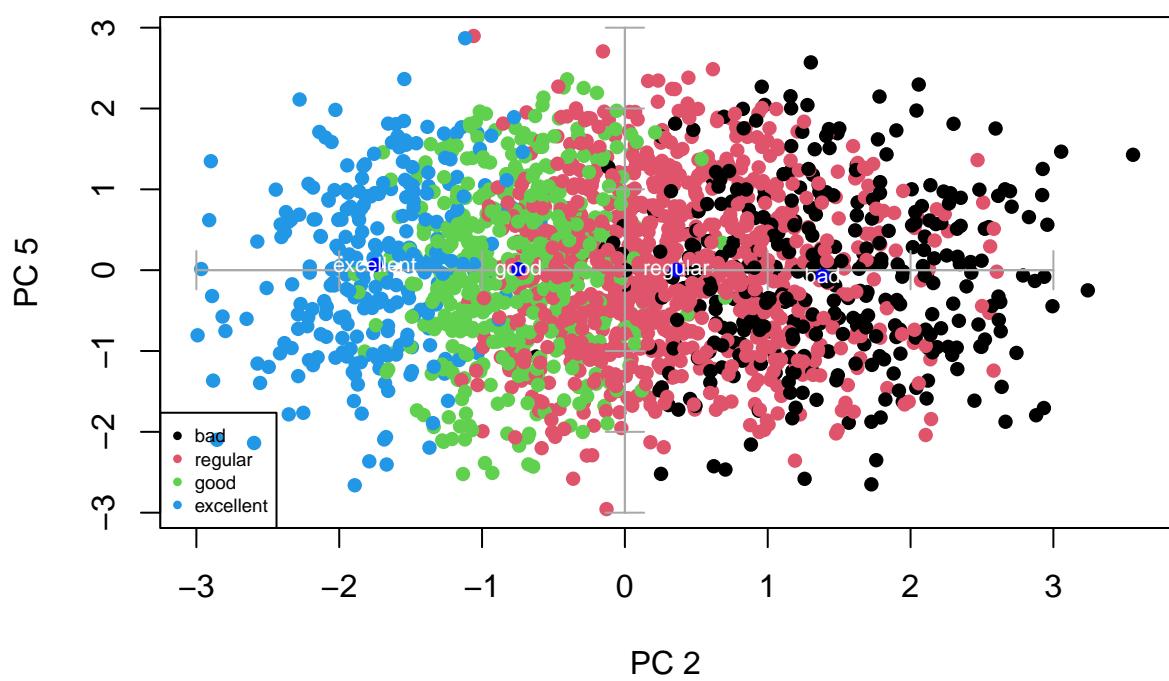
Pixel Resolution



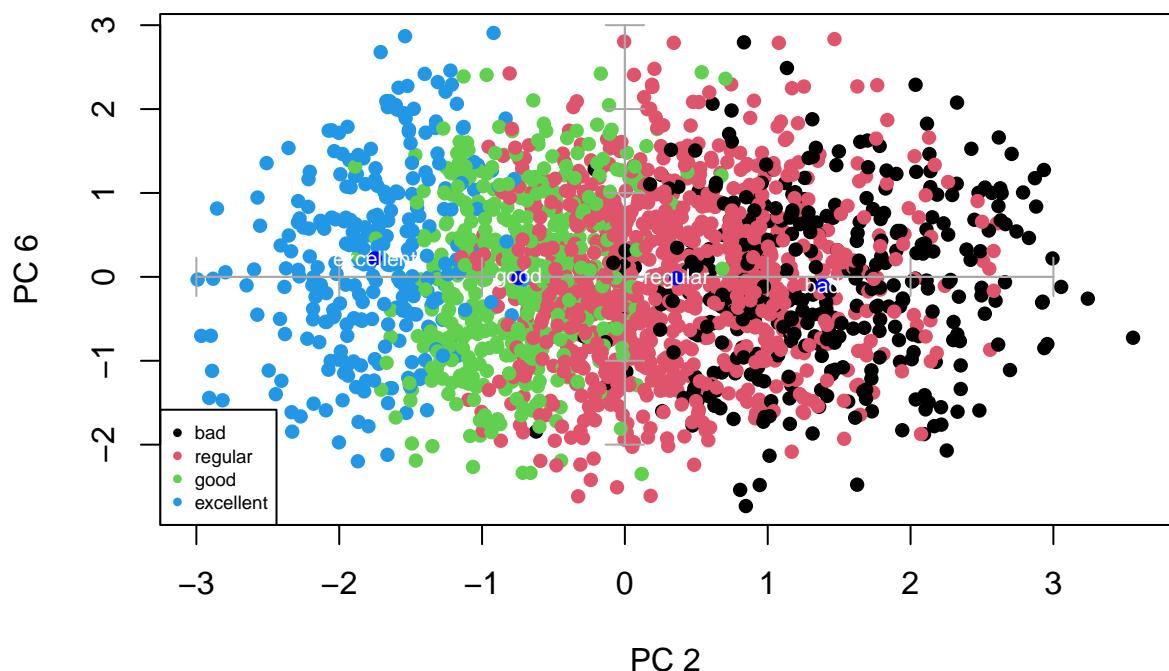
Pixel Resolution



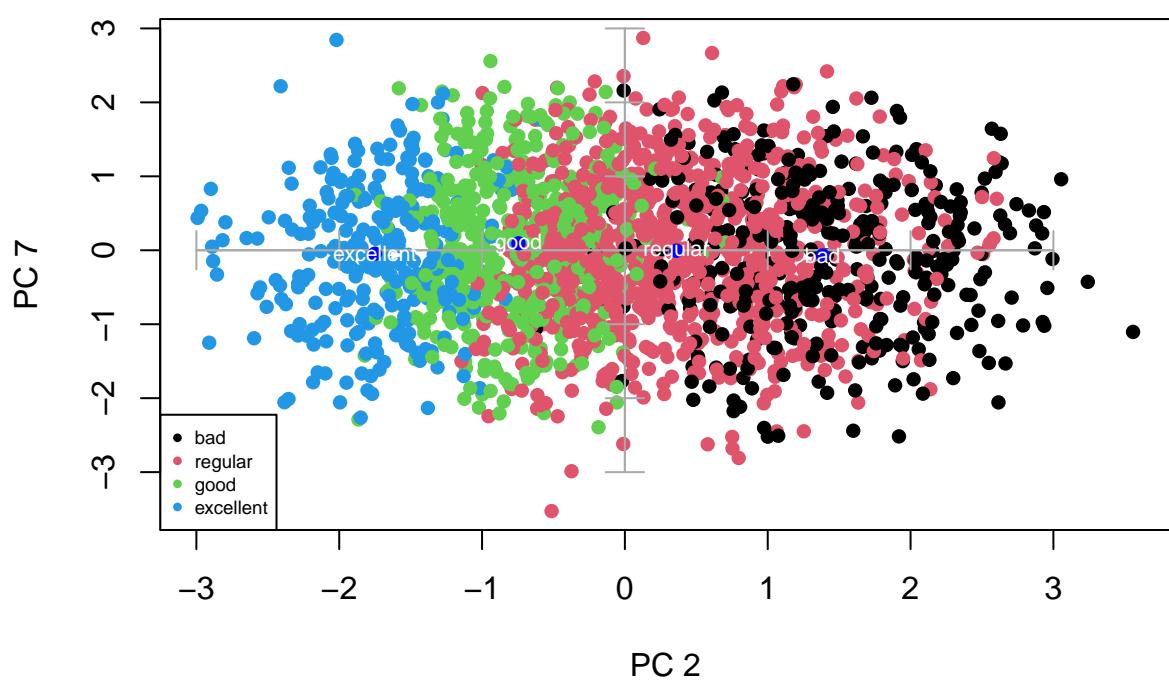
Pixel Resolution



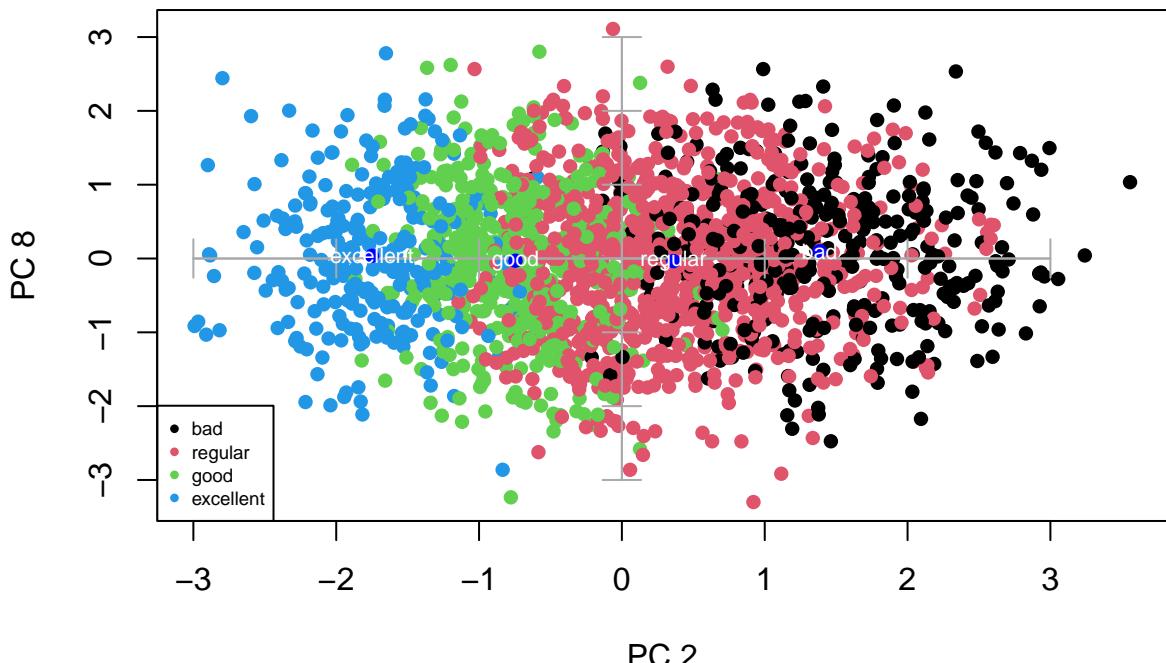
Pixel Resolution



Pixel Resolution



Pixel Resolution



As we can see, in the plots with PC2 pixel density as the x axis, we have a clear discretization of individuals colored with both sc_d (screen dimension) and px_r (pixel resolution). This suggests that the pixel density name that we guessed is correct, because it makes sense that small screens (black in Screen Dimesion plots) have a bigger pixel density than big screens (green in Screen Dimension plots). Furthermore, in Pixel Resolution Plots, higher resolution devices have higher values of pixel density than lower resolution devices.

5.2.4 Interpretation of relationships among variables

As we have already commented before, we see clear relationships between screen dimension and pixel density, and with slimness and portability. These relationship show that having a smaller or bigger screen impacts pixel density, and that a phone that is not very slim becomes less portable.

5.2.5 Conclusions

Although we have seen some relationships, most of the variables do not seem to be very well related, especially with the individuals plots. Furthermore, the relationships found, although really clear, are not actually very useful. We have found that the dataset has a lot of values that do not really make sense, because some data seems to indicate that the mobile phones are smartphones, for example a lot of them have 3G, 4G and touchscreens, but in spite of this they have really small screens and there is a variable that is talk time and not screen time which would be more suitable for smartphones. We think that PCA does not reveal a lot of useful data as a result of this inconsistencies in the data set.

6 Hierarchical Clustering

```
#dfNum <- df[,var_num]
#summary(dfNum)
#dfNum <- scale(dfNum)
#summary(dfNum)

# Dissimilarity matrix
#d <- dist(dfNum, method = "euclidean")
names(df[,c(1:11,14,17:20,22:23)])
```

```

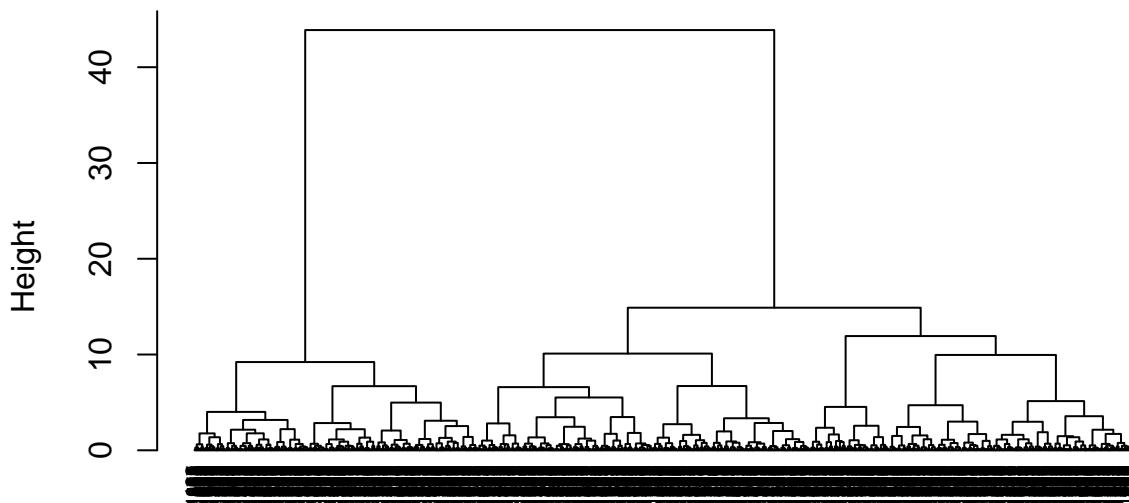
## [1] "battery_power"  "blue"          "clock_speed"   "dual_sim"
## [5] "fc"             "four_g"         "int_memory"    "m_dep"
## [9] "mobile_wt"      "n_cores"        "pc"            "ram"
## [13] "talk_time"      "three_g"        "touch_screen" "wifi"
## [17] "sc_d"           "px_r"         

dissimMatrix <- daisy(df[,c(1:11,14,17:20,22:23)], metric = "gower", stand=TRUE)
distMatrix<-dissimMatrix^2

# Hierarchical clustering using Complete Linkage
hc1 <- hclust(distMatrix, method = "ward.D" )
# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1)

```

Cluster Dendrogram



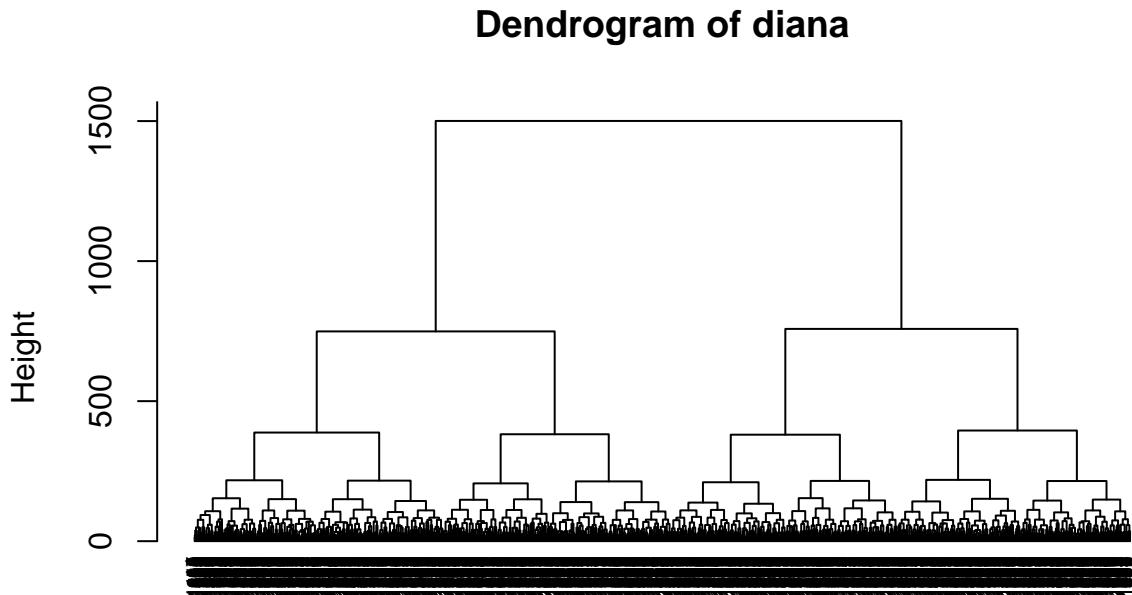
distMatrix
hclust (*, "ward.D")

```

# compute divisive hierarchical clustering
hc4 <- diana(df[,c(1:11,14,17:20,22:23)])

# Divise coefficient
#hc4$dc
# Dendrogram of diana
pltree(hc4, cex = 0.6, hang = -1, main = "Dendrogram of diana")

```



```
df[, c(1:11, 14, 17:20, 22:23)]
diana (*, "NA")
```

```
# Cluster plot
clust <- cutree(hc4, k = 4)

#fviz_cluster(list(data = df[,c(1:11,14,17:23)], cluster = clust), geom="point")
```

6.1 Precise description of the data used

6.2 Clustering method used, metrics and aggregation criteria used

6.3 Resulting Dendrogram

6.4 Discuss about how to get the final number of clusters

6.5 Table with a description of the clusters size

7 Profiling of clusters

We will check which variables are significant for each cluster we got in the last section. Then we will try to find differences and similarities between clusters and which groups of individuals characterize them.

```
dades<-as.data.frame(df)
K<-dim(dades)[2]
n<-dim(dades)[1]

P<-clust
nameP<-"Clustering classes"
nc<-length(levels(factor(P)))
pvalk <- matrix(data=0,nrow=nc,ncol=K, dimnames=list(levels(P),names(dades)))
```

7.1 Numeric Variables

```

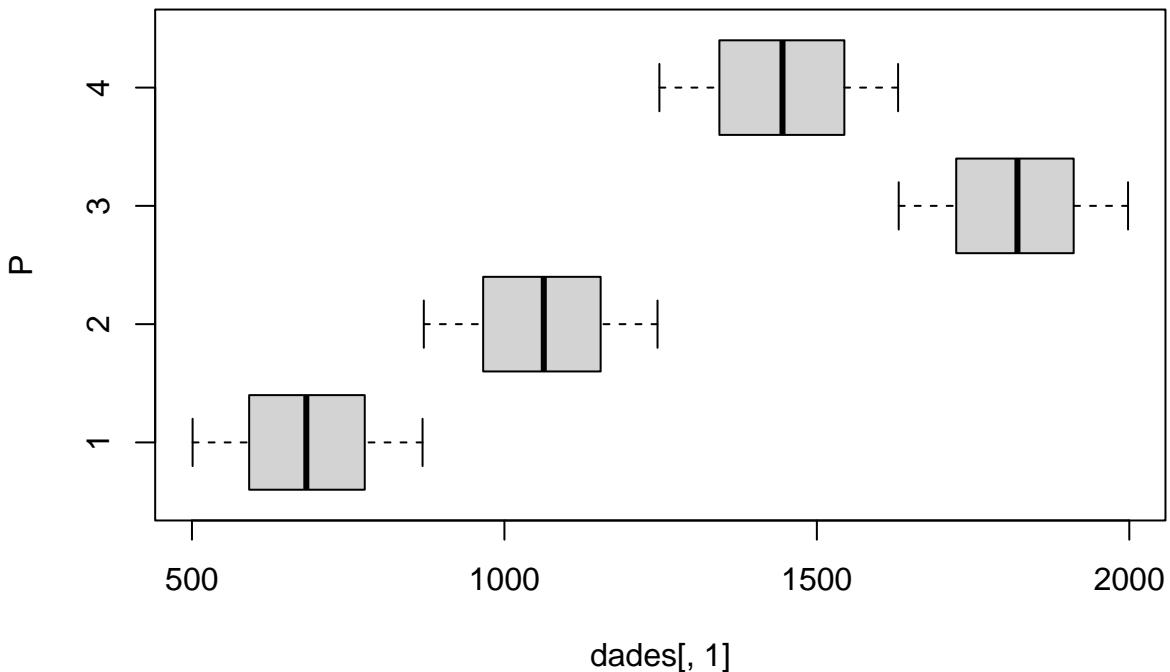
paste("Anàlisi per classes de la Variable:", names(dades)[1])

## [1] "Anàlisi per classes de la Variable: battery_power"

boxplot(dades[, 1]~P, main=paste("Boxplot of", names(dades)[1], "vs", nameP ), horizontal=TRUE)

```

Boxplot of battery_power vs Clustering classes



```

#Estadistics per groups:
for(s in levels(as.factor(P))) {print(summary(dades[P==s,1]))}

```

```

##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 501.0    591.5   683.0  684.3  776.5  869.0
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 871     966    1063  1059   1154  1245
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 1631   1723    1821  1819   1911  1998
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 1248   1344    1445  1443   1544  1630

```

```

kw<-kruskal.test(dades[,1]~P)
paste("p-value Kruskal-Wallis:", kw$p.value)

```

```

## [1] "p-value Kruskal-Wallis: 0"

```

```

pvalk[,1]<-ValorTestXnum(dades[,1], P)
#p-values ValorsTest:
pvalk[,1]

```

```

## [1] 0.000000e+00 0.000000e+00 3.864745e-194 3.062423e-32

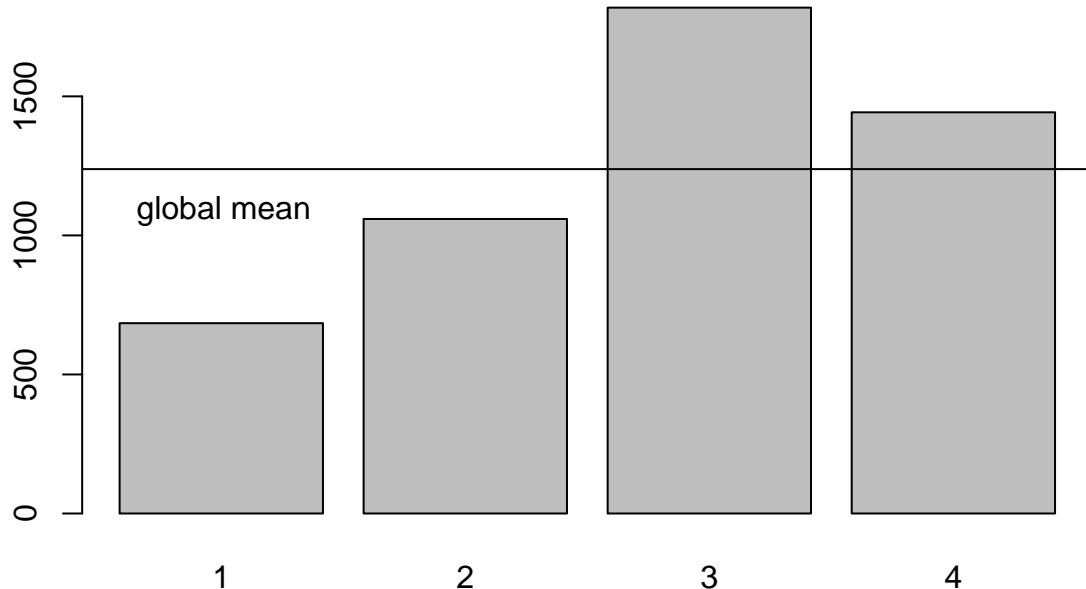
```

```

barplot(tapply(dades[[1]], P, mean), main=paste("Means of", names(dades)[1], "by", nameP ))
abline(h=mean(dades[[1]]))
legend(0,mean(dades[[1]]), "global mean", bty="n")

```

Means of battery_power by Clustering classes



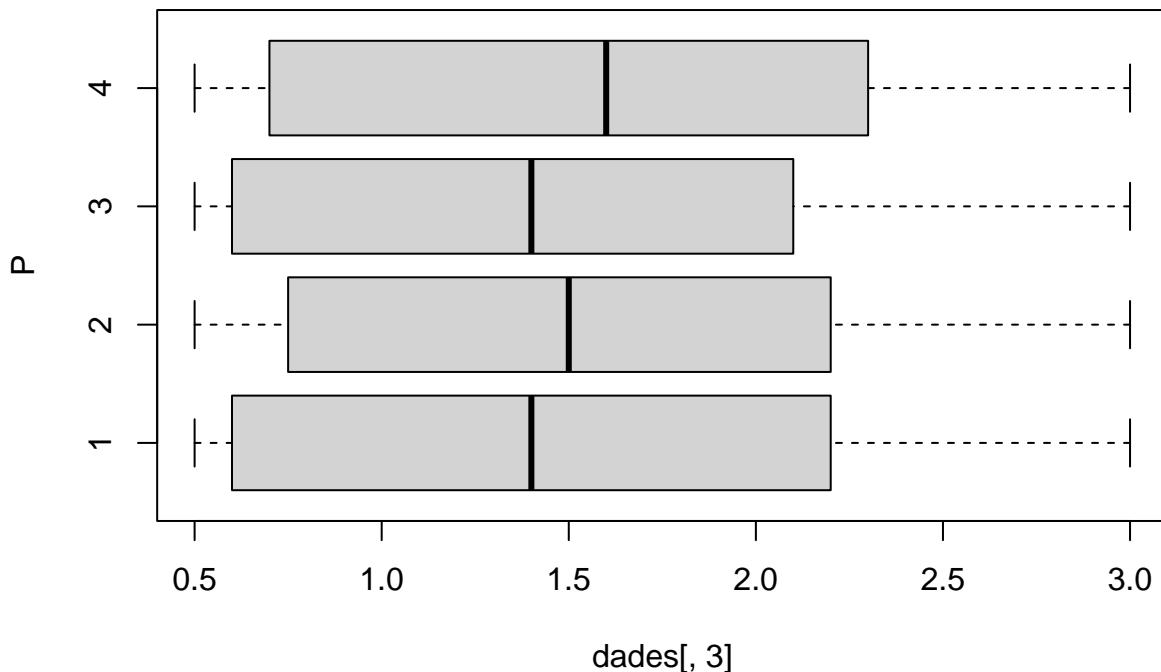
We can see in the boxplots plotted that each cluster is represented by different values of battery power. Checking the statistics by cluster groups we can appreciate the same behaviour, as the intervals of min - max values don't overlap and they have very different quartile distributions, means and medians. Doing the Kruskal-Wallis test, since our data is not normally distributed, we can reject the null hypothesis and confirm that the variable battery power presents significant differences between the cluster classes. Because of that, we will do the test-values to find the differences between the battery power mean of each class with respect to the global mean of the variable. The p-values being close to zero indicates that there are strong differences between the global mean and the mean of the cluster.

```
paste("Anàlisi per classes de la Variable:", names(dades)[3])
```

```
## [1] "Anàlisi per classes de la Variable: clock_speed"
```

```
boxplot(dades[, 3]~P, main=paste("Boxplot of", names(dades)[3], "vs", nameP ), horizontal=TRUE)
```

Boxplot of clock_speed vs Clustering classes



```
#Estadistics per groups:
```

```
for(s in levels(as.factor(P))) {print(summary(dades[P==s,3]))}
```

```
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.500    0.600  1.400  1.481  2.200  3.000
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.500    0.750  1.500  1.547  2.200  3.000
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.500    0.600  1.400  1.475  2.100  3.000
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.500    0.725  1.600  1.586  2.300  3.000
```

```
kw<-kruskal.test(dades[,3]~P)
paste("p-value Kruskal-Wallis:", kw$p.value)
```

```
## [1] "p-value Kruskal-Wallis: 0.0648137214196474"
```

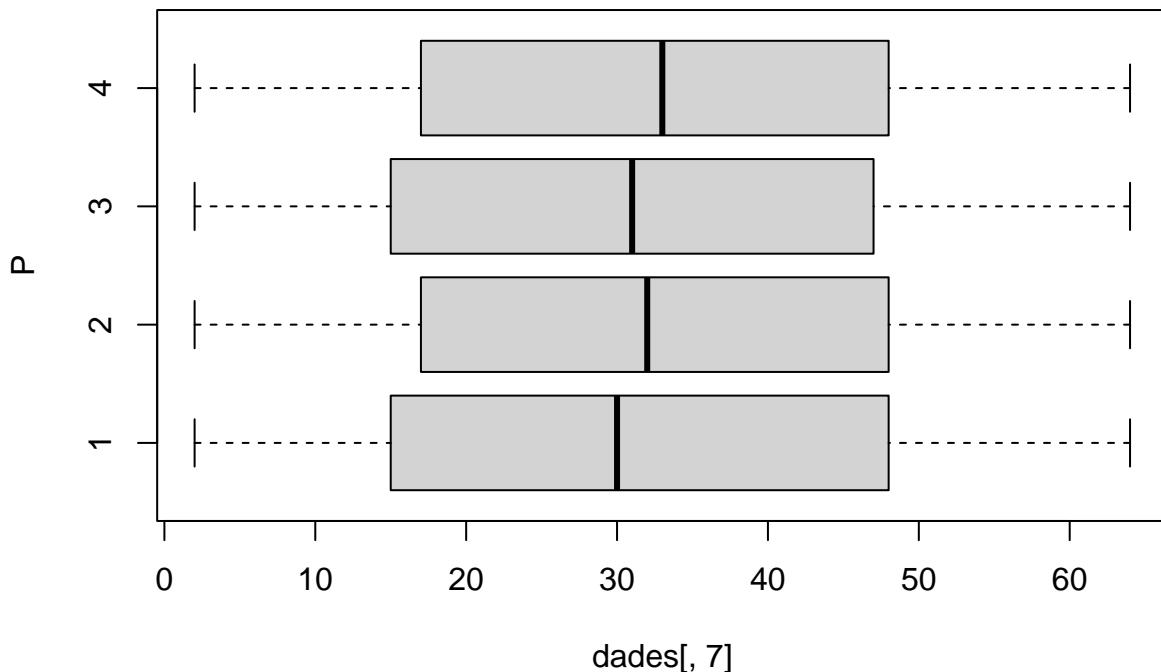
Analysing the clock speed variable we can see that it doesn't seems to be a variable that explains the clusters. The boxplots and the group statistics show that all the groups have the same quartile distribution, with just changing a little bit the mean and the median. To do a final test, we check that the p-value is not low enough so it's not a significant statistic.

```
paste("Anàlisi per classes de la Variable:", names(dades)[7])
```

```
## [1] "Anàlisi per classes de la Variable: int_memory"
```

```
boxplot(dades[, 7]~P, main=paste("Boxplot of", names(dades)[7], "vs", nameP ), horizontal=TRUE)
```

Boxplot of int_memory vs Clustering classes



```
#Estadistics per groups:
for(s in levels(as.factor(P))) {print(summary(dades[P==s,7]))}
```

```
##   Min. 1st Qu. Median Mean 3rd Qu. Max.
##   2.0   15.0   30.0  31.5  48.0   64.0
##   Min. 1st Qu. Median Mean 3rd Qu. Max.
##   2.00  17.00  32.00 32.74  48.00  64.00
##   Min. 1st Qu. Median Mean 3rd Qu. Max.
##   2.00  15.00  31.00 31.27  47.00  64.00
##   Min. 1st Qu. Median Mean 3rd Qu. Max.
##   2.00  17.25  33.00 32.75  48.00  64.00
```

```
kw<-kruskal.test(dades[,7]~P)
paste("p-value Kruskal-Wallis:", kw$p.value)
```

```
## [1] "p-value Kruskal-Wallis: 0.406972849254418"
```

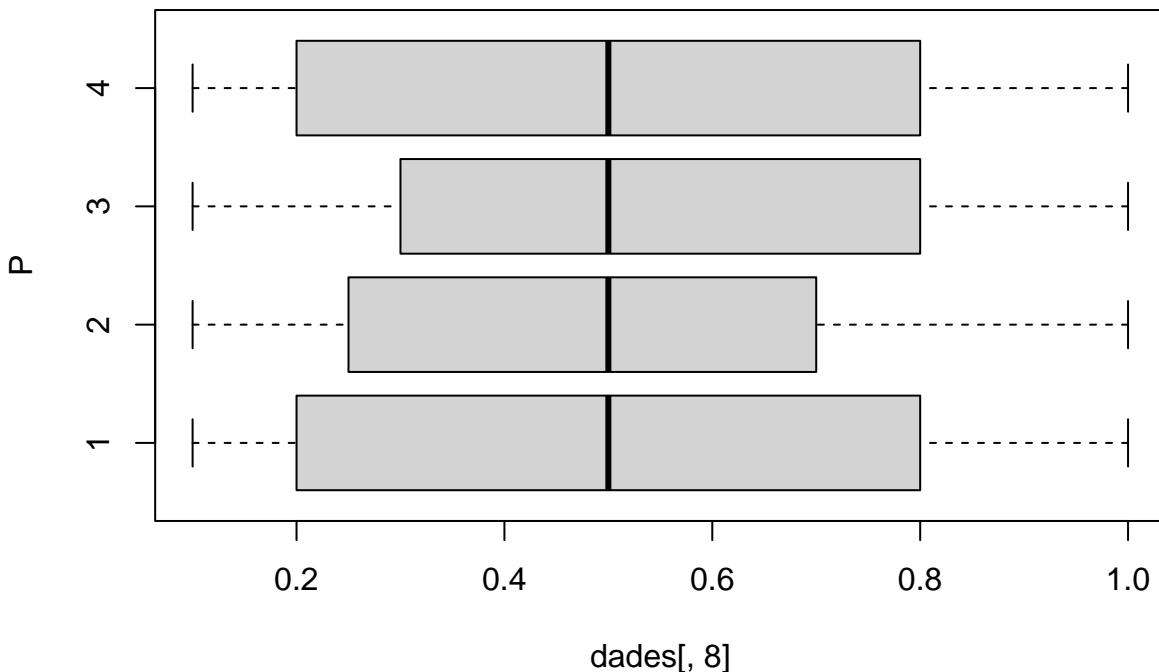
We can find a similar behaviours as the previous variable with the internal memory variable. The quartiles are similar distributed, with the same minimum and maximum values and almost the same median. The p-value of the Kruskall-Wallis test is a little bit lower than before, but not enough to be considered significant.

```
paste("Anàlisi per classes de la Variable:", names(dades)[8])
```

```
## [1] "Anàlisi per classes de la Variable: m_dep"
```

```
boxplot(dades[, 8]~P, main=paste("Boxplot of", names(dades)[8], "vs", nameP ), horizontal=TRUE)
```

Boxplot of m_dep vs Clustering classes



```
#Estadistics per groups:
for(s in levels(as.factor(P))) {print(summary(dades[P==s,8]))}
```

```
##    Min. 1st Qu. Median Mean 3rd Qu. Max.
## 0.1000 0.2000 0.5000 0.4879 0.8000 1.0000
##    Min. 1st Qu. Median Mean 3rd Qu. Max.
## 0.1000 0.2500 0.5000 0.5057 0.7000 1.0000
##    Min. 1st Qu. Median Mean 3rd Qu. Max.
## 0.1000 0.3000 0.5000 0.5243 0.8000 1.0000
##    Min. 1st Qu. Median Mean 3rd Qu. Max.
## 0.1000 0.2000 0.5000 0.4914 0.8000 1.0000
```

```
kw<-kruskal.test(dades[,8]~P)
paste("p-value Kruskal-Wallis:", kw$p.value)
```

```
## [1] "p-value Kruskal-Wallis: 0.196010293587507"
```

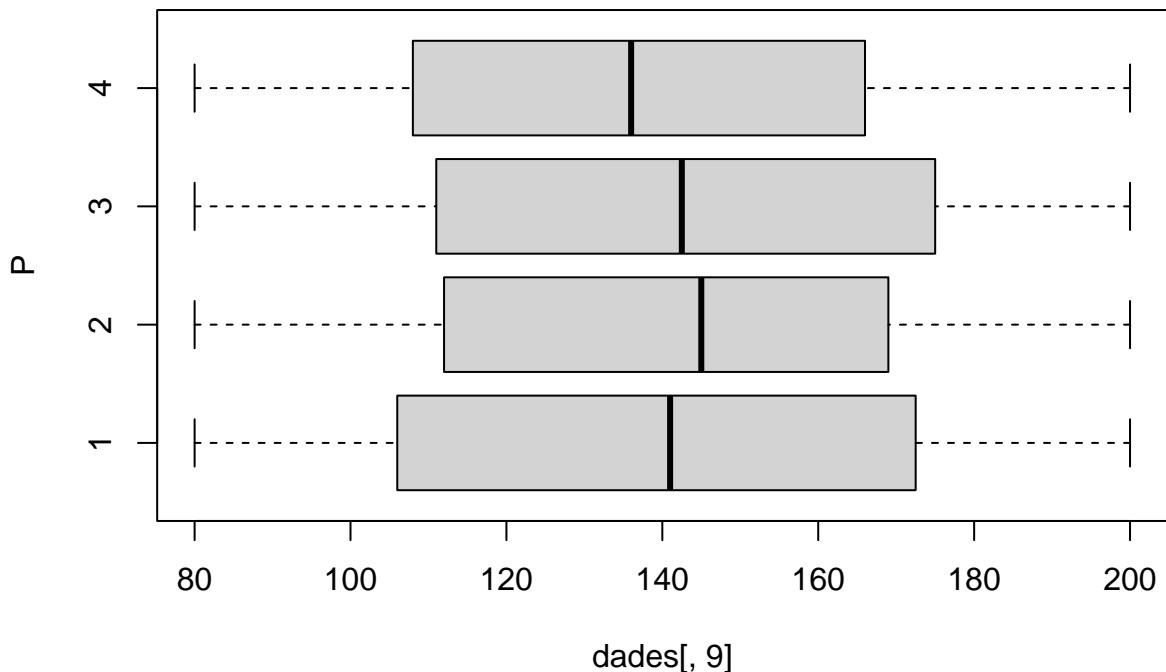
With the mobile depth variable we see a different quartile distribution, even though the intervals are still overlapping between themselves. In this case, the median is exactly the same for all 4 groups, but the quartiles of group 1 and 4 are wider than the others. To check if this group distribution is significant enough to explain the different groups we do the Kruskal-Wallis test, which shows that is not, although is smaller than the two previous variables.

```
paste("Anàlisi per classes de la Variable:", names(dades)[9])
```

```
## [1] "Anàlisi per classes de la Variable: mobile_wt"
```

```
boxplot(dades[, 9]~P, main=paste("Boxplot of", names(dades)[9], "vs", nameP ), horizontal=TRUE)
```

Boxplot of mobile_wt vs Clustering classes



```
#Estadistics per groups:
for(s in levels(as.factor(P))) {print(summary(dades[P==s,9]))}
```

```
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 80.0    106.0 141.0 139.5 172.5 200.0
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 80.0    112.0 145.0 142.1 169.0 200.0
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 80.0    111.0 142.5 142.2 174.8 200.0
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 80.0    108.0 136.0 137.4 166.0 200.0
```

```
kw<-kruskal.test(dades[,9]~P)
paste("p-value Kruskal-Wallis:", kw$p.value)
```

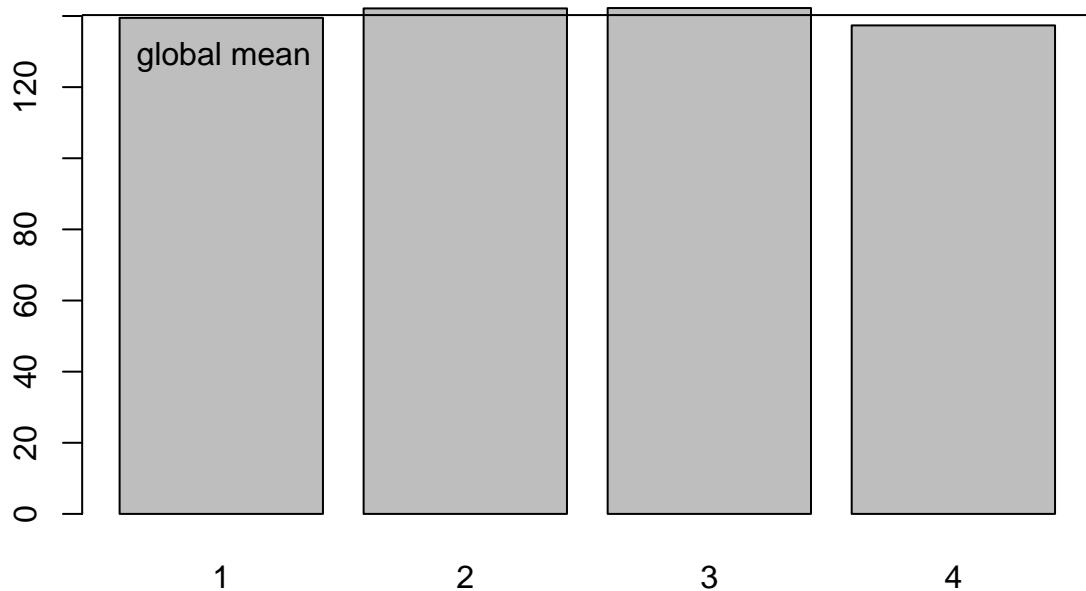
```
## [1] "p-value Kruskal-Wallis: 0.0931494961576181"
```

```
pvalk[,9]<-ValorTestXnum(dades[,9], P)
#p-values ValorsTest:
pvalk[,9]
```

```
## [1] 0.27615438 0.09053800 0.07963320 0.01733918
```

```
barplot(tapply(dades[[9]], P, mean), main=paste("Means of", names(dades)[9], "by", nameP ))
abline(h=mean(dades[[9]]))
legend(0,mean(dades[[9]]), "global mean", bty="n")
```

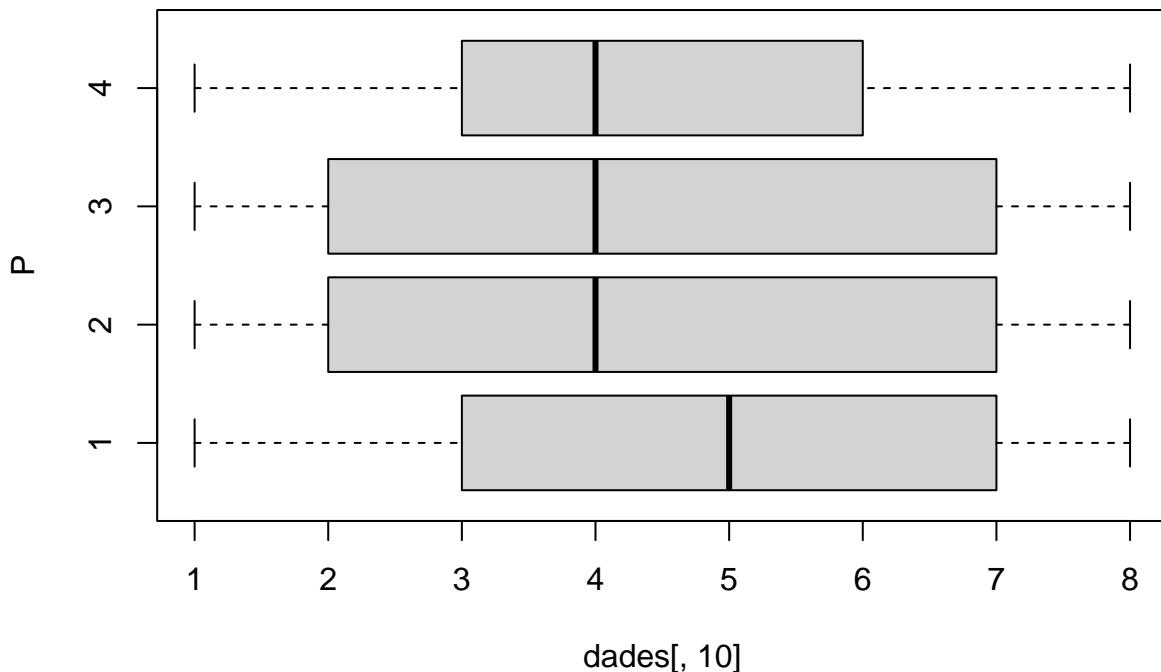
Means of mobile_wt by Clustering classes



With the mobile weight can see, again, that this variable is similar distributed in all the clusters, so it doesn't seems to be a explanatory variable for the cluster division. Doing the Kruskal-Wallis test we confirm our thoughts, despite the fact that the p-value is very low. Doing the test value just in case we find that only the last group has the enough difference between its median of the weight variable in comparison whit the global mean of the variable.

```
paste("Anàlisi per classes de la Variable:", names(dades)[10])  
  
## [1] "Anàlisi per classes de la Variable: n_cores"  
  
boxplot(dades[, 10]~P, main=paste("Boxplot of", names(dades)[10], "vs", nameP), horizontal=TRUE)
```

Boxplot of n_cores vs Clustering classes



```
#Estadistics per groups:
for(s in levels(as.factor(P))) {print(summary(dades[P==s,10]))}
```

```
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 1.000   3.000  5.000 4.652 7.000 8.000
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 1.000   2.000  4.000 4.517 7.000 8.000
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 1.000   2.000  4.000 4.525 7.000 8.000
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 1.000   3.000  4.000 4.373 6.000 8.000
```

```
kw<-kruskal.test(dades[,10]~P)
paste("p-value Kruskal-Wallis:", kw$p.value)
```

```
## [1] "p-value Kruskal-Wallis: 0.287755579757102"
```

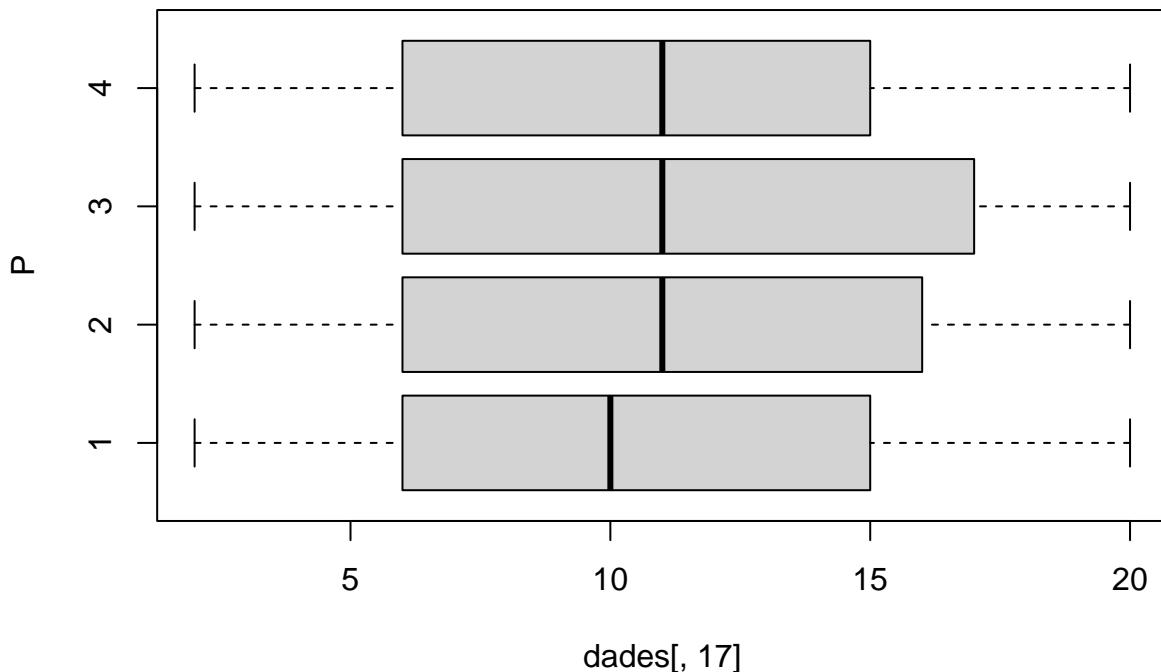
We see that the distribution of the variable number of cores is a bit different for each cluster, the 4th and the 1st cluster above all. All groups have the same median except the 1st one, and the 2nd and 3rd ones are almost the same. Because of that, the Kruskal-Wallis test shows that this variable is not representative of the clusters.

```
paste("Anàlisi per classes de la Variable:", names(dades)[17])
```

```
## [1] "Anàlisi per classes de la Variable: talk_time"
```

```
boxplot(dades[, 17]~P, main=paste("Boxplot of", names(dades)[17], "vs", nameP ), horizontal=TRUE)
```

Boxplot of talk_time vs Clustering classes



```
#Estadistics per groups:
for(s in levels(as.factor(P))) {print(summary(dades[P==s,17]))}
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##      2.00    6.00   10.00    10.68   15.00    20.00
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##      2.00    6.00   11.00    11.13   16.00    20.00
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##      2.00    6.00   11.00    11.37   17.00    20.00
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##      2.00    6.00   11.00    10.9    15.0     20.0
```

```
kw<-kruskal.test(dades[,17]~P)
paste("p-value Kruskal-Wallis:", kw$p.value)
```

```
## [1] "p-value Kruskal-Wallis: 0.221767146688966"
```

The last numerical variable is the talk time of the phones. This variable shows the same quartile distributions for all groups, with the same means except for the 1st group and a very similar value of the 1st and 3rd quartile. The p-value of the Kruskal-Wallis test is not significant enough to consider this variables as a group representative.

7.2 Binary Variables

```
paste("Variable", names(dades)[2])
```

```
## [1] "Variable blue"
```

```
#Cross-table
table<-table(P,dades[,2])
table
```

```

##  

## P      NoBth YesBth  

## 1      272    259  

## 2      247    240  

## 3      233    249  

## 4      257    241  

rowperc<-prop.table(table,1)  

rowperc

```

```

##  

## P      NoBth     YesBth  

## 1 0.5122411 0.4877589  

## 2 0.5071869 0.4928131  

## 3 0.4834025 0.5165975  

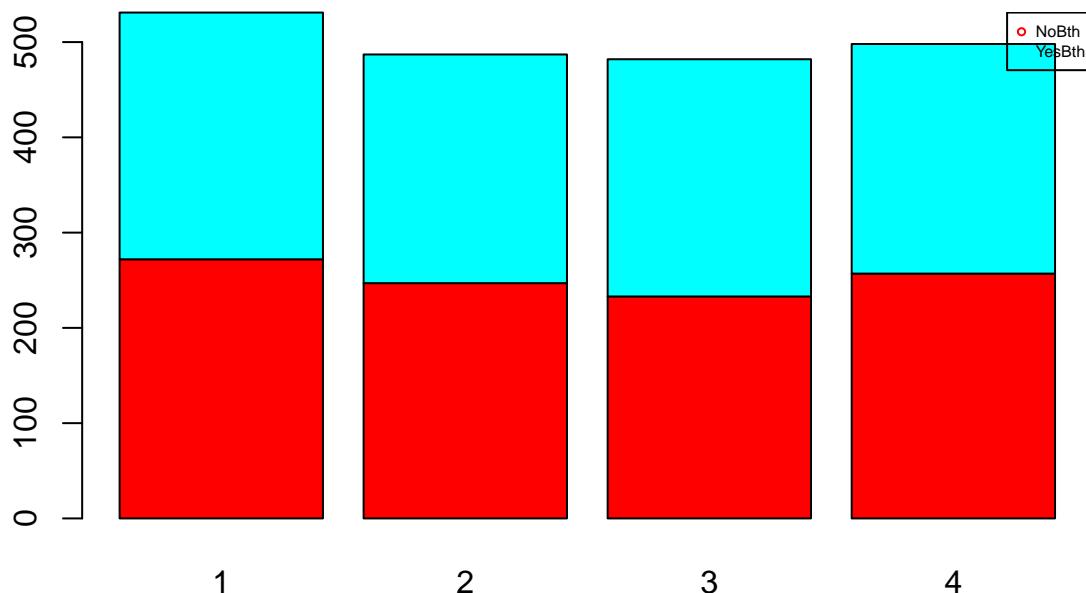
## 4 0.5160643 0.4839357

```

```

#diagrammes de barres apilades
paleta<-rainbow(length(levels(dades[,2])))
barplot(table(dades[,2], as.factor(P)), beside=FALSE, col=paleta )
legend("topright", levels(as.factor(dades[,2])), pch=1, cex=0.5, col=paleta)

```



```

#Test Chi quadrat:  

chisq.test(dades[,2] , as.factor(P))

```

```

##  

## Pearson's Chi-squared test  

##  

## data: dades[, 2] and as.factor(P)  

## X-squared = 1.264, df = 3, p-value = 0.7377

```

The first binary variable we will check is the Bluetooth Yes/No variable. As we can see if we do the contingency table and the proportions by group, the variable is distributed equitably for all 4 groups, so it doesn't seem

to explain any group in particular. The stacked bars shows a similar distributions within the clusters, which is confirmed by the Chi-Squared test. Since the p-value is not low enough to be significant, we can confirm that the clusters and the variable are not dependant. The proportions by category and the attached bars barplot can be found in the annex.

```

paste("Variable", names(dades)[4])

## [1] "Variable dual_sim"

#Cross-table
table<-table(P,dades[,4])
table

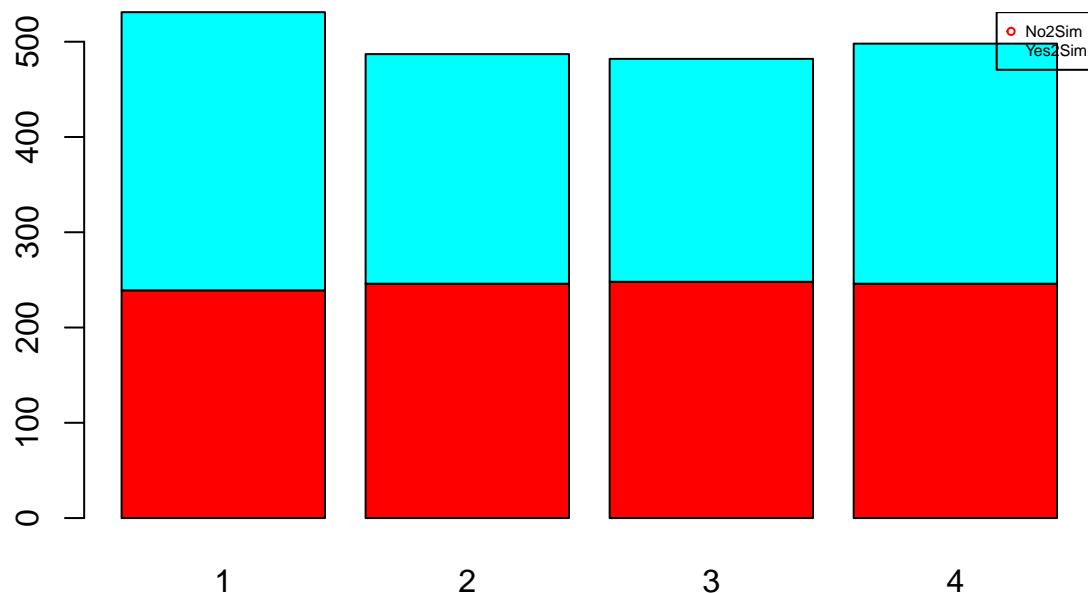
## 
## P      No2Sim Yes2Sim
## 1      239     292
## 2      246     241
## 3      248     234
## 4      246     252

rowperc<-prop.table(table,1)
rowperc

## 
## P      No2Sim Yes2Sim
## 1 0.4500942 0.5499058
## 2 0.5051335 0.4948665
## 3 0.5145228 0.4854772
## 4 0.4939759 0.5060241

#diagrammes de barres apilades
paleta<-rainbow(length(levels(dades[,4])))
barplot(table(dades[,4], as.factor(P)), beside=FALSE,col=paleta )
legend("topright",levels(as.factor(dades[,4])),pch=1,cex=0.5, col=paleta)

```



```
#Test Chi quadrat:  
chisq.test(dades[,4], as.factor(P))
```

```
##  
## Pearson's Chi-squared test  
##  
## data: dades[, 4] and as.factor(P)  
## X-squared = 5.0215, df = 3, p-value = 0.1702
```

As we have seen in the first binary variable, the Dual Sim variable also has a similar distribution in all groups, as the contingency table shows. The only group with a little difference is the first one, given that more mobiles with Dual Sim fall into that group. The Chi-Squared test confirms our thoughts, the p-values in not significant enough to consider this variable as an explanatory for the chosen clusters.

```
paste("Variable", names(dades)[6])
```

```
## [1] "Variable four_g"
```

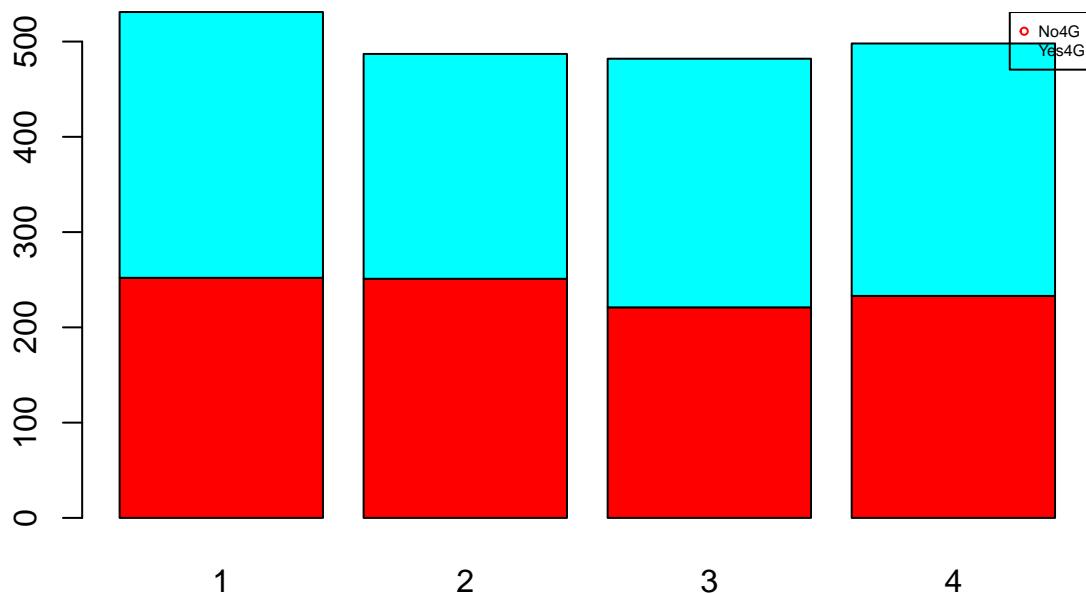
```
#Cross-table  
table<-table(P,dades[,6])  
table
```

```
##  
## P      No4G Yes4G  
## 1     252   279  
## 2     251   236  
## 3     221   261  
## 4     233   265
```

```
rowperc<-prop.table(table,1)  
rowperc
```

```
##  
## P          No4G      Yes4G  
## 1 0.4745763 0.5254237  
## 2 0.5154004 0.4845996  
## 3 0.4585062 0.5414938  
## 4 0.4678715 0.5321285
```

```
#diagrammes de barres apilades  
paleta<-rainbow(length(levels(dades[,6])))  
barplot(table(dades[,6], as.factor(P)), beside=FALSE, col=paleta )  
legend("topright", levels(as.factor(dades[,6])), pch=1, cex=0.5, col=paleta)
```



```
#Test Chi quadrat:
chisq.test(dades[,6], as.factor(P))
```

```
##
## Pearson's Chi-squared test
##
## data: dades[, 6] and as.factor(P)
## X-squared = 3.6856, df = 3, p-value = 0.2975
```

The 4G Yes/No variable seems to have a uneven distribution in comparison with the other variables we checked. The third and fourth groups have more mobiles with 4G. The Chi-Squared test gives us a p-value even higher than the Dual Sim variable, so this variable is also not accepted as a explanatory variable for the groups.

```
paste("Variable", names(dades)[18])
```

```
## [1] "Variable three_g"
```

```
#Cross-table
table<-table(P,dades[,18])
table
```

```
##
## P      No3G  Yes3G
## 1     128    403
## 2     118    369
## 3     111    371
## 4     120    378
```

```
rowperc<-prop.table(table,1)
rowperc
```

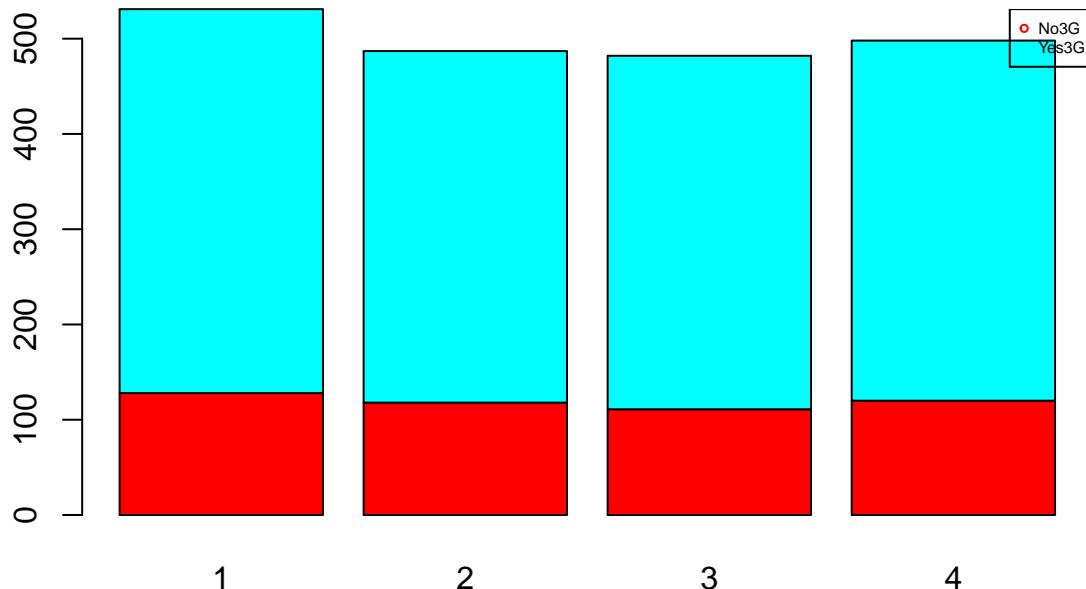
```
##
## P          No3G      Yes3G
```

```

##   1 0.2410546 0.7589454
##   2 0.2422998 0.7577002
##   3 0.2302905 0.7697095
##   4 0.2409639 0.7590361

#diagrammes de barres apiladas
paleta<-rainbow(length(levels(dades[,18])))
barplot(table(dades[,18], as.factor(P)), beside=FALSE, col=paleta )
legend("topright", levels(as.factor(dades[,18])), pch=1, cex=0.5, col=paleta)

```



```

#Test Chi quadrat:
chisq.test(dades[,18], as.factor(P))

```

```

##
## Pearson's Chi-squared test
##
## data: dades[, 18] and as.factor(P)
## X-squared = 0.25251, df = 3, p-value = 0.9687

```

Seeing the contingency table and the proportions by group we can see that the 3G binary variable has a special distribution. Even though all groups have similar proportions, those proportions are not close to 50-50 as we have seen in the previous binary variables. All the groups have approximately 75% of mobiles with 3G and 25% of mobiles without them, which is obvious since this variable is very unbalanced. The p-value of the Chi-Squared test, again, confirms that this variable doesn't present significant differences between the clusters.

```
paste("Variable", names(dades)[19])
```

```
## [1] "Variable touch_screen"
```

```
#Cross-table
table<-table(P,dades[,19])
table
```

```

##  

## P      NoTouch YesTouch  

## 1      263     268  

## 2      241     246  

## 3      244     238  

## 4      246     252  

rowperc<-prop.table(table,1)  

rowperc

```

```

##  

## P      NoTouch YesTouch  

## 1 0.4952919 0.5047081  

## 2 0.4948665 0.5051335  

## 3 0.5062241 0.4937759  

## 4 0.4939759 0.5060241

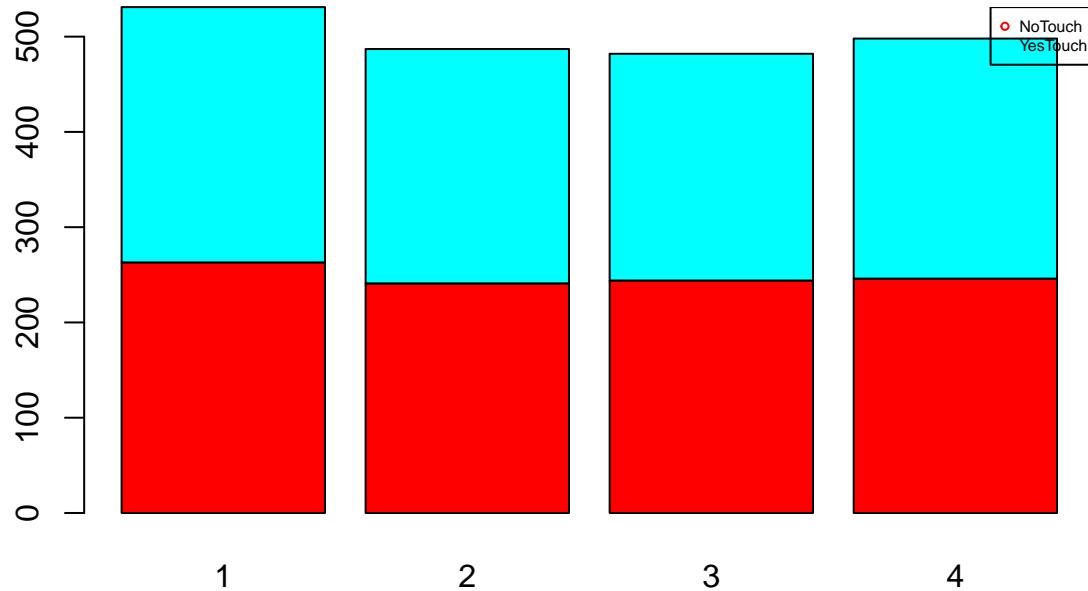
```

#diagrammes de barres apilades

```

paleta<-rainbow(length(levels(dades[,19])))
barplot(table(dades[,19], as.factor(P)), beside=FALSE, col=paleta )
legend("topright", levels(as.factor(dades[,19])), pch=1, cex=0.5, col=paleta)

```



#Test Chi quadrat:

```

chisq.test(dades[,19], as.factor(P))

```

```

##  

## Pearson's Chi-squared test  

##  

## data: dades[, 19] and as.factor(P)  

## X-squared = 0.19535, df = 3, p-value = 0.9783

```

We find a similar distribution with the first binary variables we checked and the Touch Screen variable. Again, the distribution is the same for all the classes and the Chi-Squared Test confirms that this variable is not important to describe the groups.

```
paste("Variable", names(dades)[20])
```

```
## [1] "Variable wifi"
```

```
#Cross-table
```

```
table<-table(P,dades[,20])
table
```

```
##
```

```
## P      NoWifi YesWifi
## 1      255     276
## 2      248     239
## 3      246     236
## 4      235     263
```

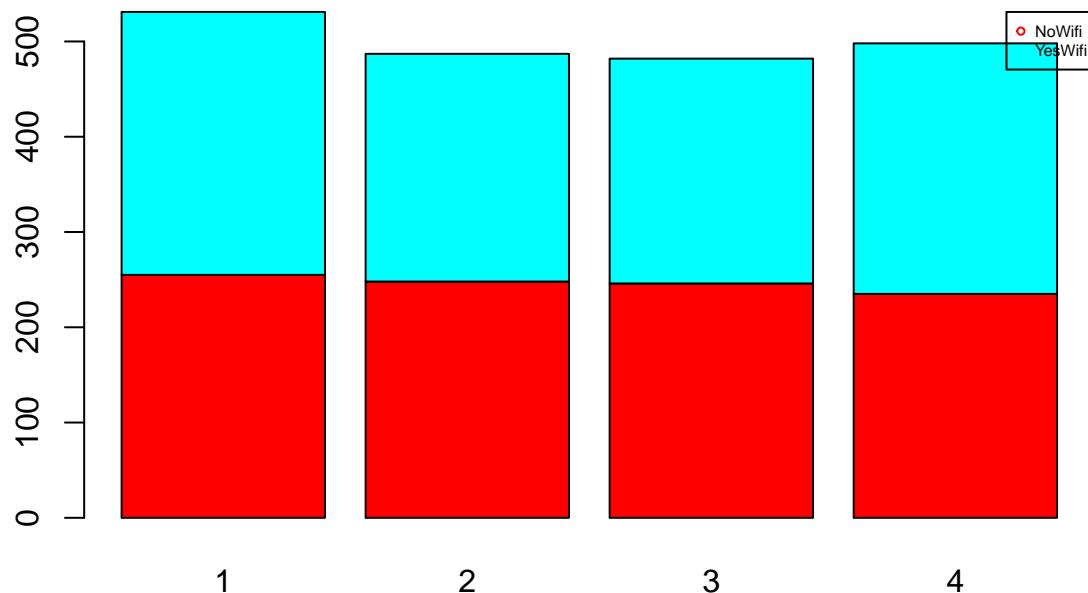
```
rowperc<-prop.table(table,1)
rowperc
```

```
##
```

```
## P      NoWifi YesWifi
## 1 0.4802260 0.5197740
## 2 0.5092402 0.4907598
## 3 0.5103734 0.4896266
## 4 0.4718876 0.5281124
```

```
#diagrammes de barres apilades
```

```
paleta<-rainbow(length(levels(dades[,20])))
barplot(table(dades[,20], as.factor(P)), beside=FALSE, col=paleta )
legend("topright", levels(as.factor(dades[,20])), pch=1, cex=0.5, col=paleta)
```



```
#Test Chi quadrat:
```

```
chisq.test(dades[,20], as.factor(P))
```

```

## 
## Pearson's Chi-squared test
## 
## data: dades[, 20] and as.factor(P)
## X-squared = 2.3287, df = 3, p-value = 0.5071

```

The last binary variable we are going to analyse is the WiFi variable. The first and last cluster seem to have more mobiles with WiFi than the other groups in proportion, but not enough to be significant to represent them.

7.3 Categorical Variables

```
paste("Variable", names(dades)[5])
```

```
## [1] "Variable fc"
```

```
#Cross-table
table<-table(P,dades[,5])
table
```

```

## 
## P      NoFrontCam bad regular good excellent
## 1          128 234     119    40      10
## 2          117 211     104    47       8
## 3          102 215     108    44      13
## 4          127 215      97    48      11

```

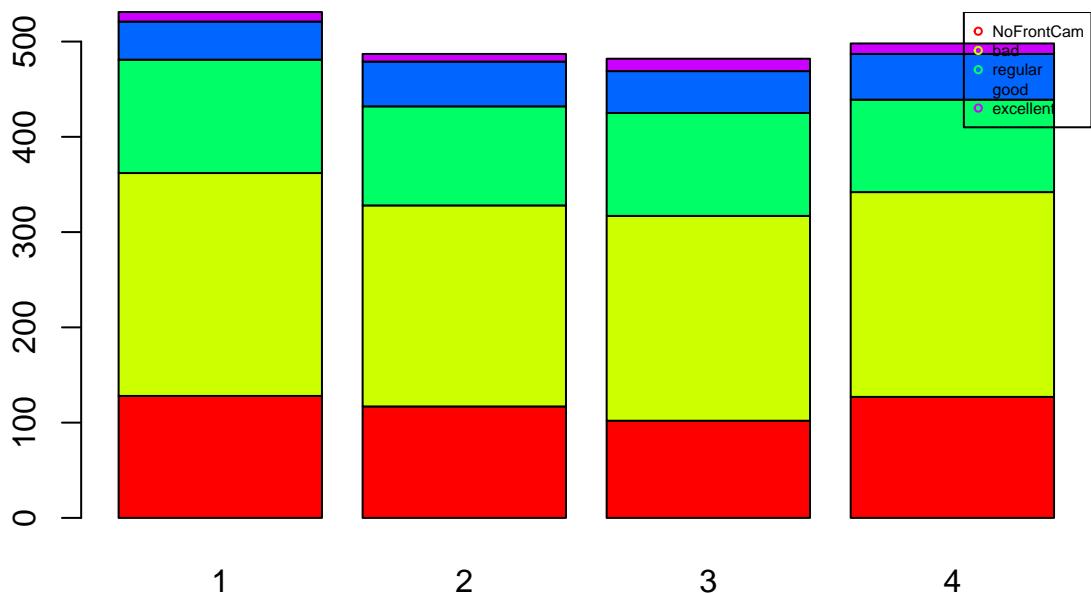
```
rowperc<-prop.table(table,1)
rowperc
```

```

## 
## P      NoFrontCam      bad      regular      good      excellent
## 1 0.24105461 0.44067797 0.22410546 0.07532957 0.01883239
## 2 0.24024641 0.43326489 0.21355236 0.09650924 0.01642710
## 3 0.21161826 0.44605809 0.22406639 0.09128631 0.02697095
## 4 0.25502008 0.43172691 0.19477912 0.09638554 0.02208835

```

```
#diagrammes de barres apilades
paleta<-rainbow(length(levels(dades[,5])))
barplot(table(dades[,5], as.factor(P)), beside=FALSE,col=paleta )
legend("topright",levels(as.factor(dades[,5])),pch=1,cex=0.5, col=paleta)
```



```
#Test Chi quadrat:
chisq.test(dades[,5], as.factor(P))
```

```
##
## Pearson's Chi-squared test
##
## data: dades[, 5] and as.factor(P)
## X-squared = 6.7248, df = 12, p-value = 0.8753
```

The first categorical variable we are going to profile is the front camera quality. We can see in the contingency table and the proportion by groups that all the groups are not distributed similarly, but it doesn't seem to be enough differences between them to explain each group by the variable categories. The barplots shows us the same, so we will check the Chi-Squared test to find if the variable and the clusters are independent. As we see, the p-value is not significant enough to consider the variable and the groups dependant, so we can conclude saying that the front camera quality is not a variable that explains the different clusters.

```
paste("Variable", names(dades)[11])
## [1] "Variable pc"

#Cross-table
table<-table(P,dades[,11])
table

##
##   P   NoMainCam bad regular good excellent
##   1       23 128    148   127      105
##   2       26 106    129   118      108
##   3       27 103    124   105      123
##   4       25 113    135    99      126
```

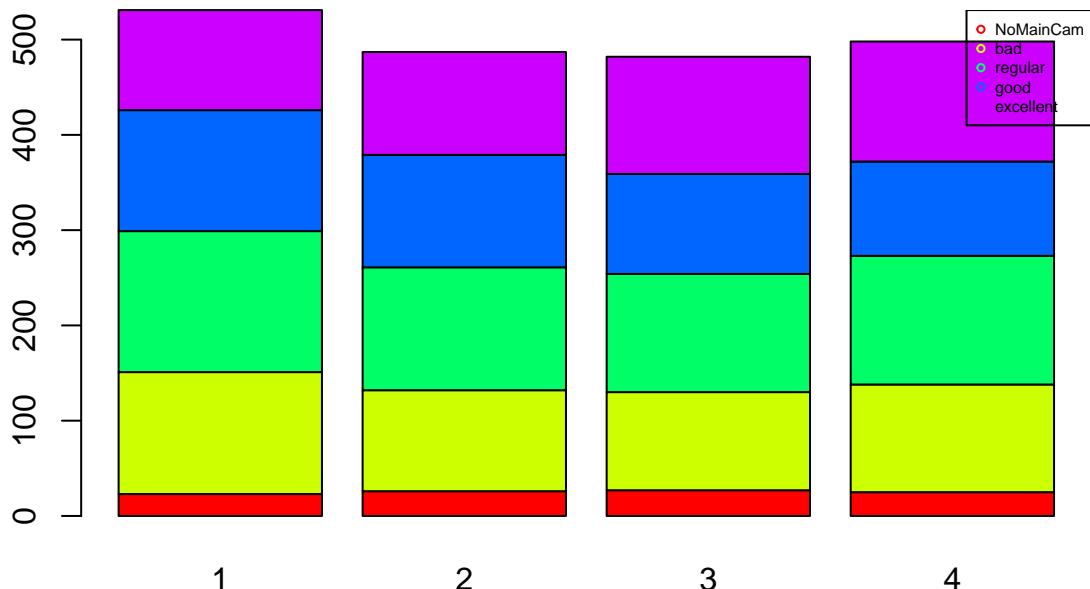
```

rowperc<-prop.table(table,1)
rowperc

##
## P      NoMainCam      bad    regular     good   excellent
## 1 0.04331450 0.24105461 0.27871940 0.23917137 0.19774011
## 2 0.05338809 0.21765914 0.26488706 0.24229979 0.22176591
## 3 0.05601660 0.21369295 0.25726141 0.21784232 0.25518672
## 4 0.05020080 0.22690763 0.27108434 0.19879518 0.25301205

#diagrammes de barres apilades
paleta<-rainbow(length(levels(dades[,11])))
barplot(table(dades[,11], as.factor(P)), beside=FALSE,col=paleta )
legend("topright",levels(as.factor(dades[,11])),pch=1,cex=0.5, col=paleta)

```



```

#Test Chi quadrat:
chisq.test(dades[,11], as.factor(P))

```

```

##
## Pearson's Chi-squared test
##
## data: dades[, 11] and as.factor(P)
## X-squared = 10.125, df = 12, p-value = 0.605

```

We can find a similar behaviour with the variable main camera quality. The groups have a distribution that varies a little for each group, but again, it doesn't seem to be different enough to consider the main camera a variable that explains the clusters. The Chi-Squared test confirms us that, even though the p-value is lower than in the first categorical variable.

```

paste("Variable", names(dades)[14])

```

```

## [1] "Variable ram"

```

```
#Cross-table
table<-table(P,dades[,14])
table
```

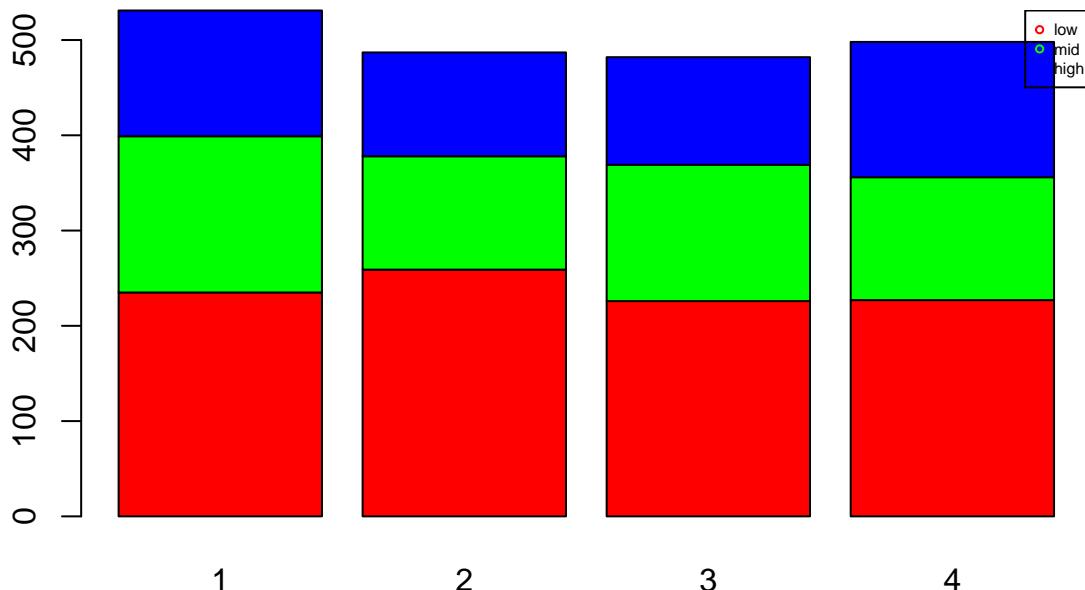
```
##
## P      low   mid  high
## 1    235 164 132
## 2    259 119 109
## 3    226 143 113
## 4    227 129 142
```

```
rowperc<-prop.table(table,1)
rowperc
```

```
##
## P          low        mid        high
## 1 0.4425612 0.3088512 0.2485876
## 2 0.5318275 0.2443532 0.2238193
## 3 0.4688797 0.2966805 0.2344398
## 4 0.4558233 0.2590361 0.2851406
```

#diagrammes de barres apilades

```
paleta<-rainbow(length(levels(dades[,14])))
barplot(table(dades[,14], as.factor(P)), beside=FALSE, col=paleta )
legend("topright",levels(as.factor(dades[,14])),pch=1,cex=0.5, col=paleta)
```



#Test Chi quadrat:

```
chisq.test(dades[,14], as.factor(P))
```

```
##
## Pearson's Chi-squared test
##
## data: dades[, 14] and as.factor(P)
## X-squared = 14.243, df = 6, p-value = 0.02704
```

```

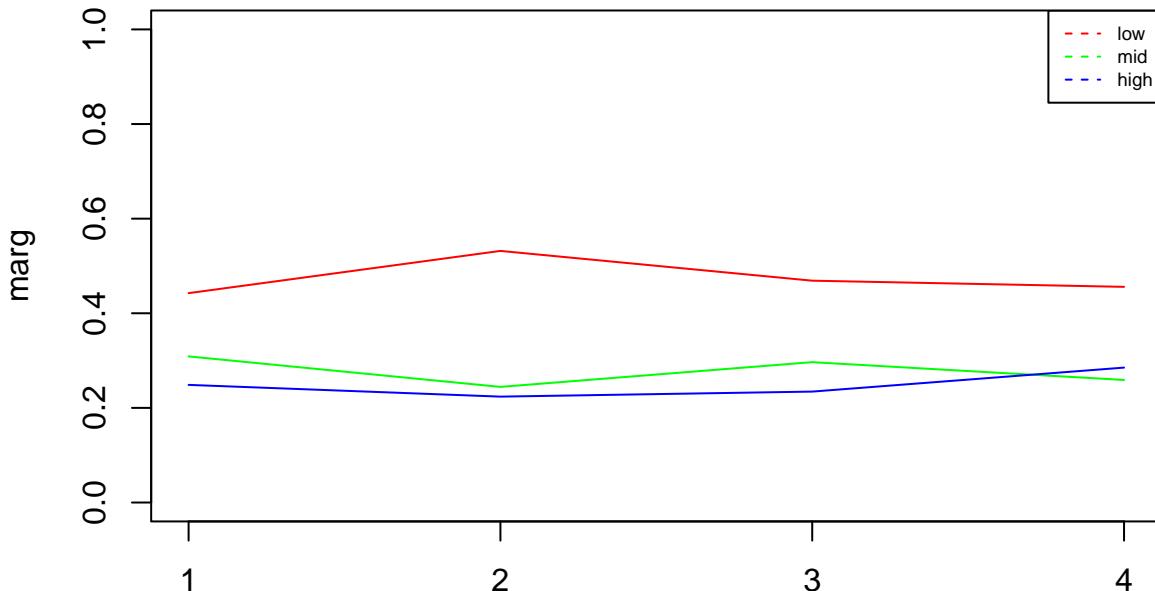
#relevant variable
#valorsTest:
ValorTestXquali(P,dades[,14])

## $rowpf
##   Xquali
## P      low      mid      high
## 1 0.4425612 0.3088512 0.2485876
## 2 0.5318275 0.2443532 0.2238193
## 3 0.4688797 0.2966805 0.2344398
## 4 0.4558233 0.2590361 0.2851406
##
## $vtest
##   Xquali
## P      low      mid      high
## 1 -1.69182446 1.86567158 0.02112332
## 2 2.94021491 -1.89370345 -1.43500828
## 3 -0.25714366 1.06368196 -0.80562910
## 4 -0.93622199 -1.07768055 2.19949186
##
## $pval
##   Xquali
## P      low      mid      high
## 1 0.045339725 0.031043664 0.491573642
## 2 0.001639923 0.029132191 0.075642369
## 3 0.398533938 0.143736401 0.210228370
## 4 0.174579451 0.140588171 0.013921484

marg <- table(as.factor(P))/n
#condicionades a classes amb llegenda
plot(marg,type="n",ylim=c(0,1),main= paste("Prop. of pos & neg by",names(dades)[14]))
paleta<-rainbow(length(levels(dades[,14])))
for(c in 1:length(levels(dades[,14]))) {lines(rowperc[,c],col=paleta[c])}
legend("topright", levels(dades[,14]), col=paleta, lty=2, cex=0.6)

```

Prop. of pos & neg by ram



With this variable we can start to find some differences between clusters. Having a lot of megabytes of RAM seems to be a key factor to represent the fourth group, and having a small and middle RAM seems to explain the second cluster. The barplots doesn't give us a lot of information so we will check with the Chi-Squared test if the variable and the clusters are dependant. The p-value that the test gives us is significant enough to consider the RAM categorical variable relevant, therefore we will do the test-values to see the relation between proportions of the category in every cluster and the proportion of all the data, and the p-values of each category and group.

We can see that the first group is related with the Low and Middle RAM category. The second group is related with all three categories, especially with the Low Ram, since it has a very low p-value. The third cluster is not related with any RAM category, but the fourth one is also strongly related with the High Ram category. We can visually see this doing a snake plot.

```
paste("Variable", names(dades)[22])
```

```
## [1] "Variable sc_d"
```

#Cross-table

```
table<-table(P,dades[,22])
table
```

```
##
```

```
## P      small   medium   big
## 1      372     129     30
## 2      333     123     31
## 3      341     109     32
## 4      331     130     37
```

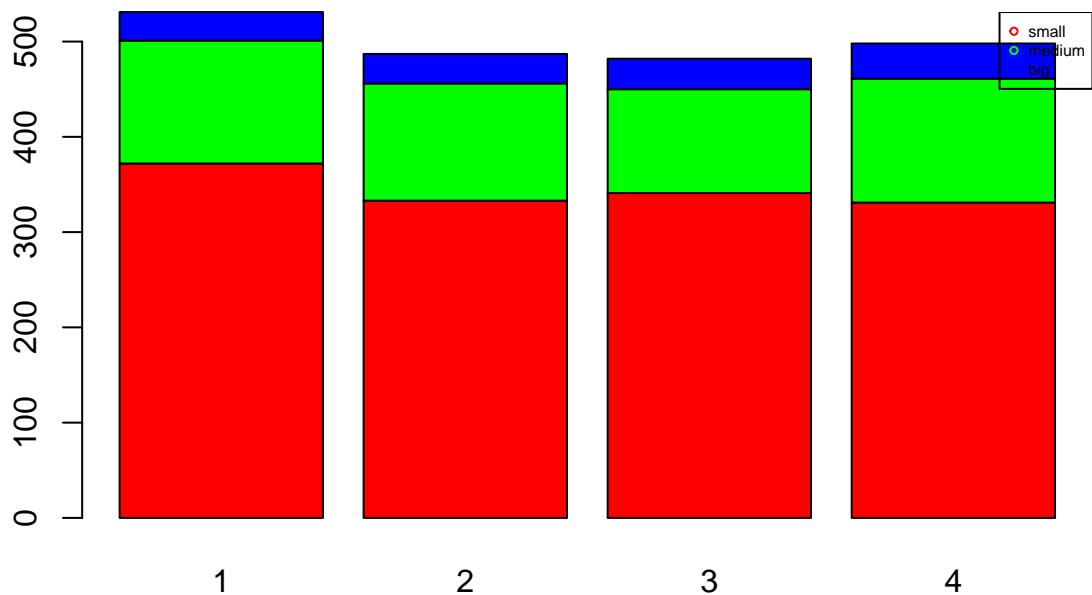
```
rowperc<-prop.table(table,1)
rowperc
```

```
##
```

```
## P          small       medium       big
## 1 0.70056497 0.24293785 0.05649718
## 2 0.68377823 0.25256674 0.06365503
## 3 0.70746888 0.22614108 0.06639004
## 4 0.66465863 0.26104418 0.07429719
```

#diagrammes de barres apilades

```
paleta<-rainbow(length(levels(dades[,22])))
barplot(table(dades[,22], as.factor(P)), beside=FALSE, col=paleta )
legend("topright",levels(as.factor(dades[,22])),pch=1,cex=0.5, col=paleta)
```



```
#Test Chi quadrat:
chisq.test(dades[,22], as.factor(P))

##
## Pearson's Chi-squared test
##
## data: dades[, 22] and as.factor(P)
## X-squared = 3.4056, df = 6, p-value = 0.7565
```

The next variable we will profile is the screen dimension variable. Remember that this variables is a factor variable from two numerical variables that represented the screen height and width. With the contingency table and the proportions by row we check that the first group contains a bigger quantity of small phones, the third group has less medium sized phones and the last group has the most quantity of big phones. Even though the distribution is not the same for all the clusters, it doesn't variate enough to consider this variables as an explanatory, which the Chi-Square test confirms giving us a not enough significant p-value.

```
paste("Variable", names(dades)[23])
```

```
## [1] "Variable px_r"
```

```
#Cross-table
table<-table(P,dades[,23])
table
```

```
##
## P    bad regular good excellent
## 1    91      251   119       70
## 2    86      221   118       62
## 3    85      212   109       76
## 4    97      219   117       65
```

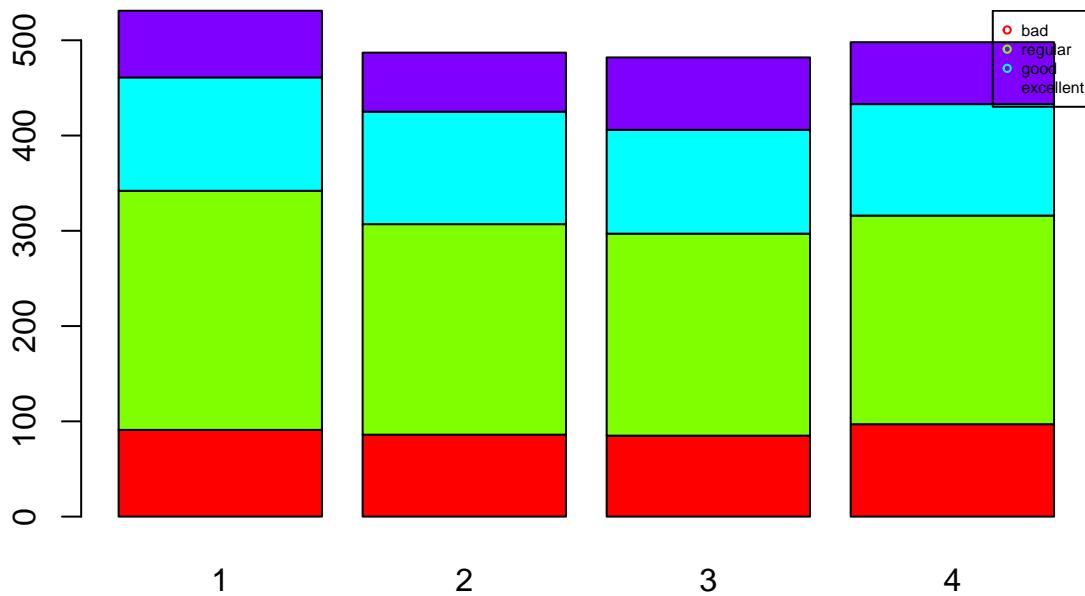
```

rowperc<-prop.table(table,1)
rowperc

## 
## P      bad   regular    good excellent
## 1 0.1713748 0.4726930 0.2241055 0.1318267
## 2 0.1765914 0.4537988 0.2422998 0.1273101
## 3 0.1763485 0.4398340 0.2261411 0.1576763
## 4 0.1947791 0.4397590 0.2349398 0.1305221

#diagrammes de barres apilades
paleta<-rainbow(length(levels(dades[,23])))
barplot(table(dades[,23], as.factor(P)), beside=FALSE, col=paleta )
legend("topright",levels(as.factor(dades[,23])),pch=1,cex=0.5, col=paleta)

```



```

#Test Chi quadrat:
chisq.test(dades[,23], as.factor(P))

```

```

##
## Pearson's Chi-squared test
##
## data: dades[, 23] and as.factor(P)
## X-squared = 4.2738, df = 9, p-value = 0.8925

```

The last variable we profile is the Pixels per Inch (pixel ratio) variable. We can see that the categories are equally distributed between the clusters, although the first and last groups seem to differ a bit with all the others. Doing the Chi-Squared test we can find if this differences are enough to consider this variable related to the clusters. The p-value gives us a high number, so it's not sufficiently significant to consider a relations between the variable and the groups.

7.4 Classes' interpretation

After profiling every category with the clusters we found doing the hierarchical clustering, we can try to interpret the cluster classes with the information we extracted in the previous sections.

The first cluster is characterized by having a very low battery power, much less than the average and the other classes, and contains mobiles without a High Ram capacity. The second cluster has mobiles with very low RAM and more battery power than the first group, but still bellow the average. The third cluster is the one that contains mobiles with very high battery power. Finally, the fourth group is characterized by mobiles with high ram capacity and with the less weight in comparison with the other groups.

8 Global discussions and general conclusions

... {Joan H.}

9 Working plan

9.1 Gantt's diagram

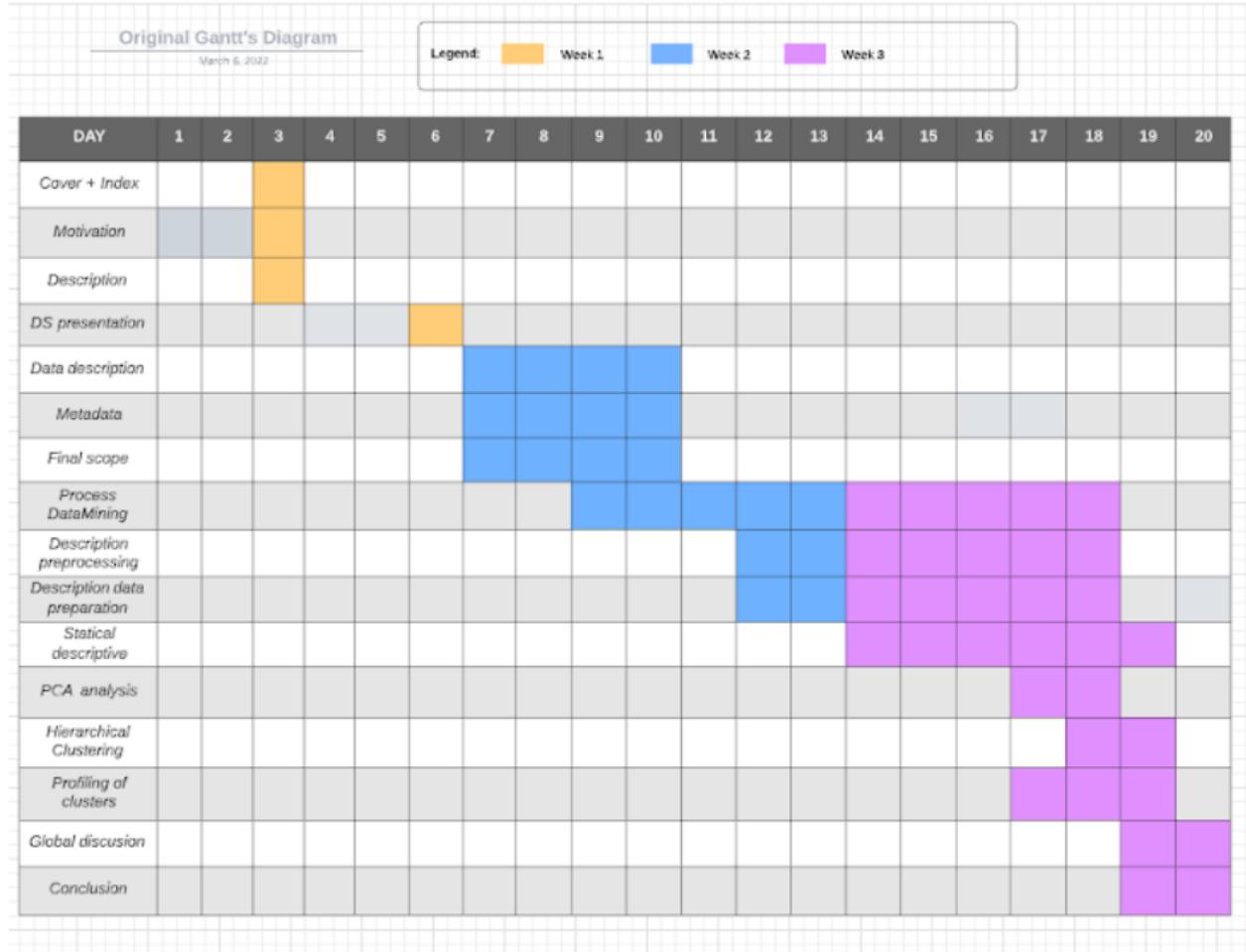


Figure 5: alt text

9.2 Task assignment grid

9.3 Deviances of final scheduling with respect to the original

...

	Joan H.	Joan D.	Xènia	Claudia	Joban	Victor
Gantt diagram			X			
Risk plan	X					
Metadata file describing the selection of variables considered for the analysis					X	
Basic initial univariate descriptive statistics of raw variables				X		
Enumerate which steps of the preprocessing process are used		X				
List and justify all decisions taken for each preprocessing step						X
Cover+Table of contents		X				
Motivation of the work and general description of the problem		X				
Data source presentation	X					
Formal description of data structure and metadata					X	
Complete data mining process performed					X	
Detailed description of preprocessing and data preparation						X
Basic statistical descriptive analysis				X		
PCA analysis for numerical variables	X		X			
Hierarchical Clustering on original data						X
Profiling of clusters		X				
Global discussion and general conclusions of whole work	X					

Figure 6: alt text

9.4 Risk contingency plan

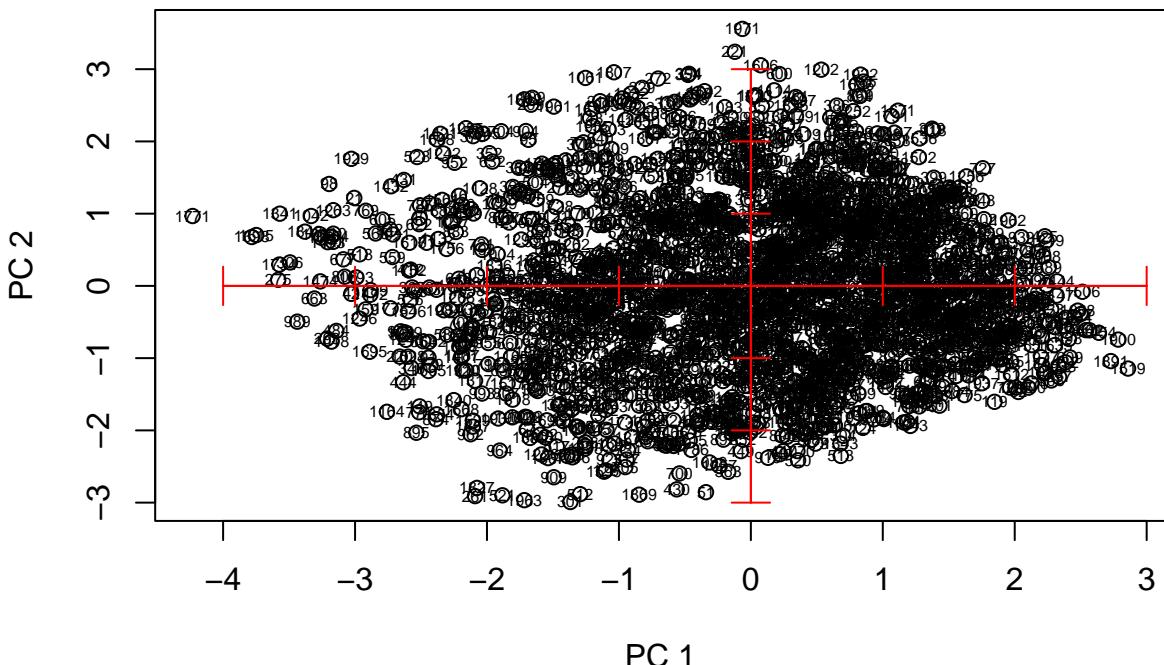
Risk	How to prevent	How to manage
Team member gets Covid-19 or gets sick	Can not really prevent this issue	Affected team member keeps working from home, the rest of the group keeps in touch via Discord and Whatsapp
Work deleted/overwritten	Have cloud backups (github) and personal backups in our PCs	Restore to the last backed up version and keep working as expected
Team member can't work that week because of unexpected personal problems	Can not really prevent this issue	Work assigned to said team member is redistributed over the rest of the group
Team member leaves the course	Tasks have an alternate member assigned	Pending work is reassigned to re-balance efforts

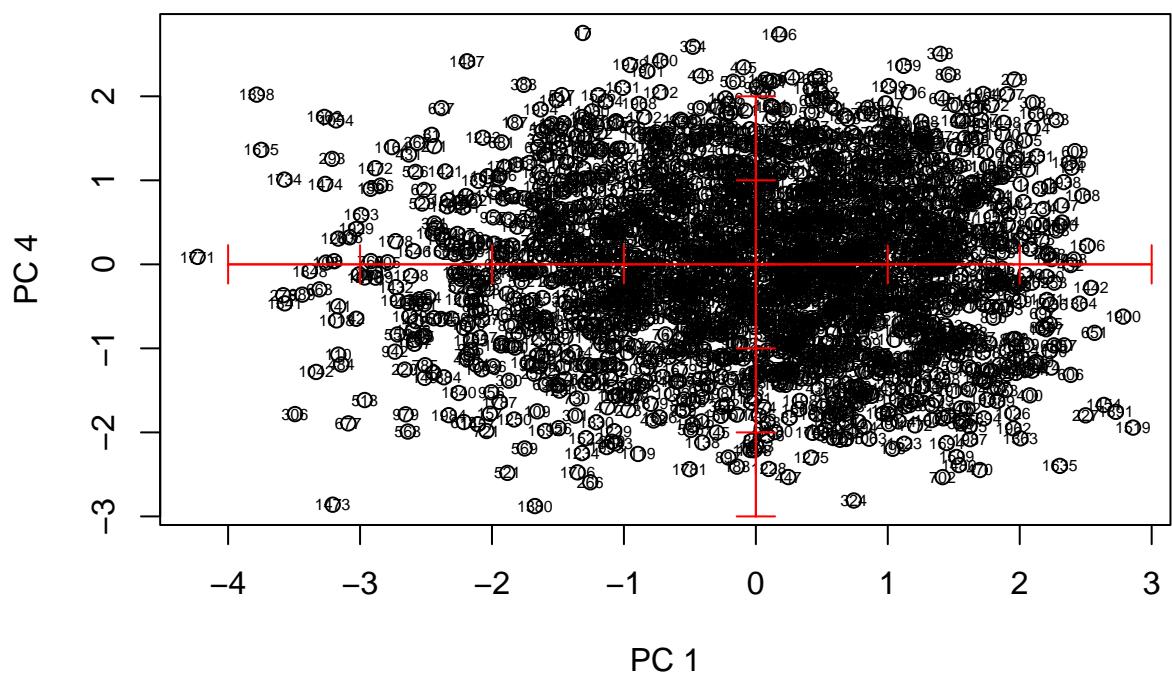
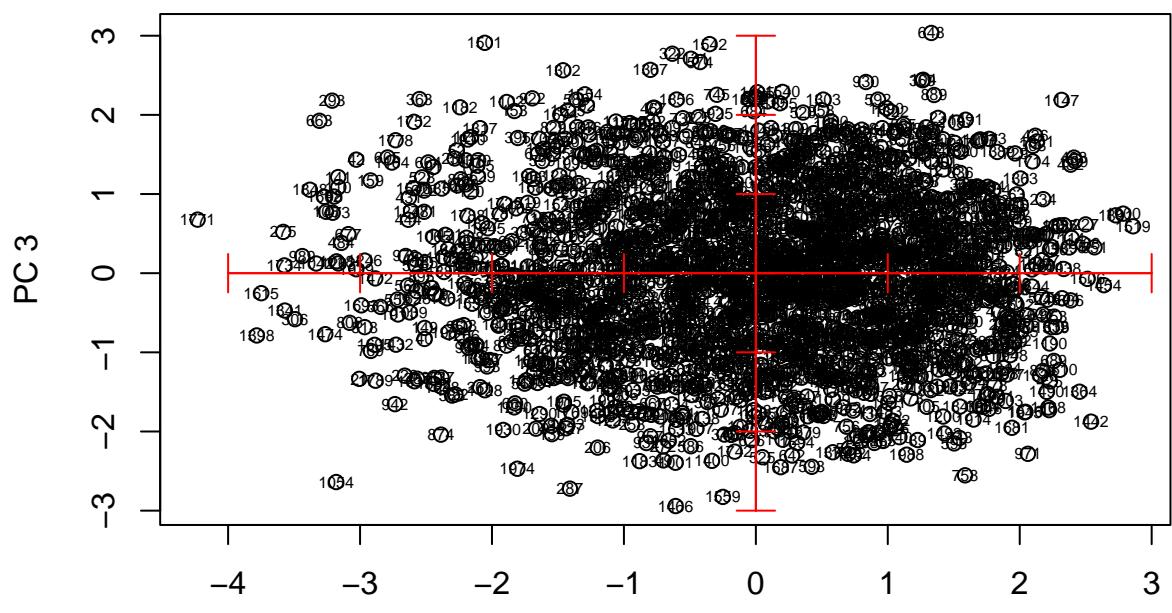
10 Annex

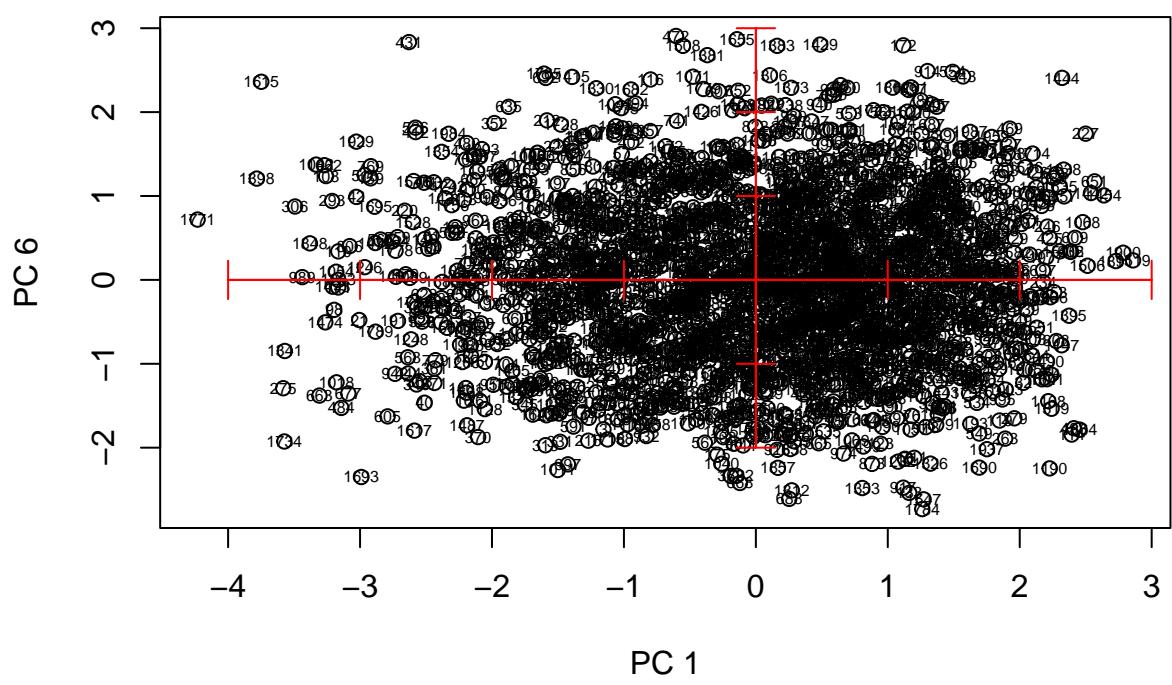
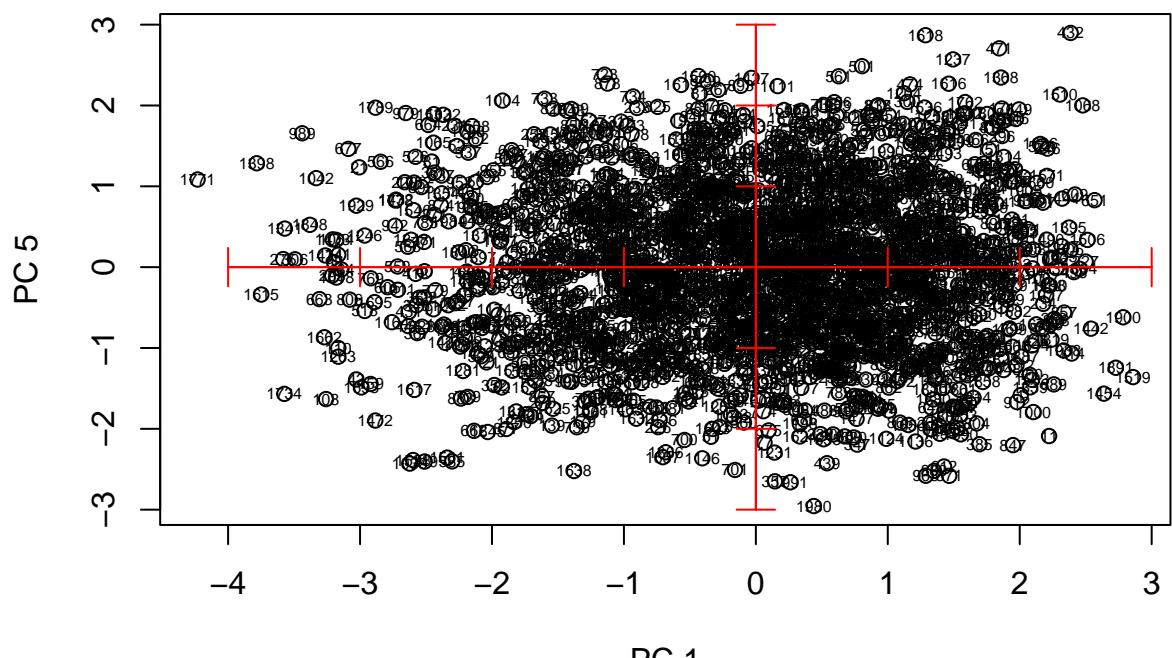
10.1 Individuals projections

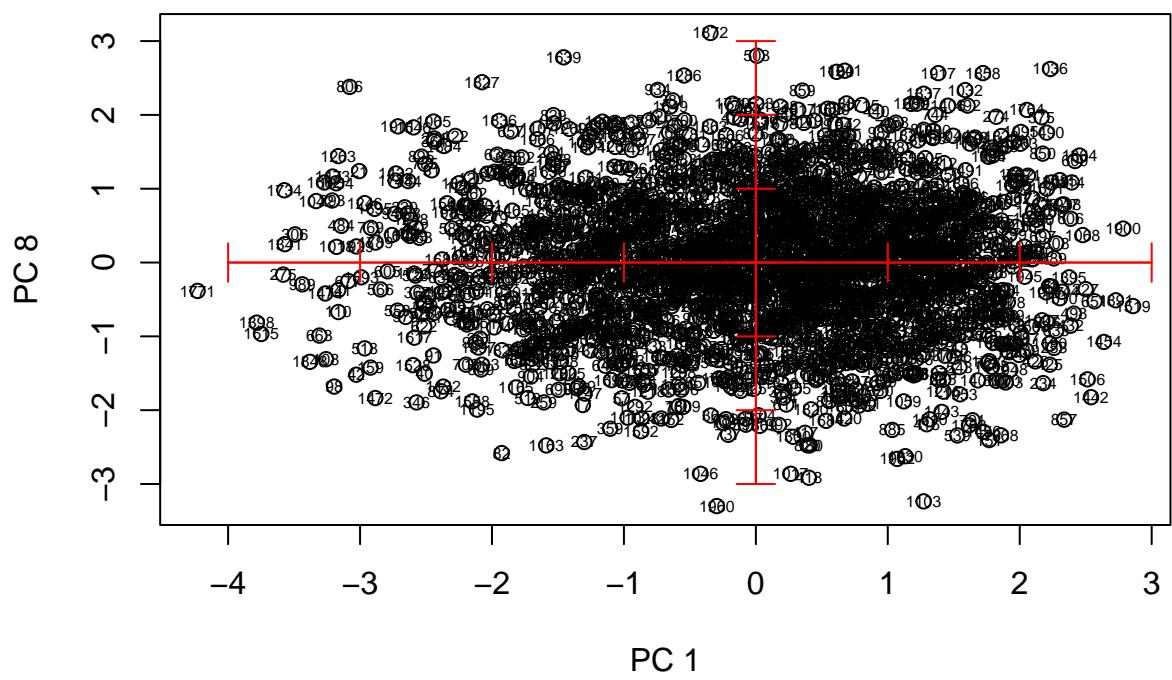
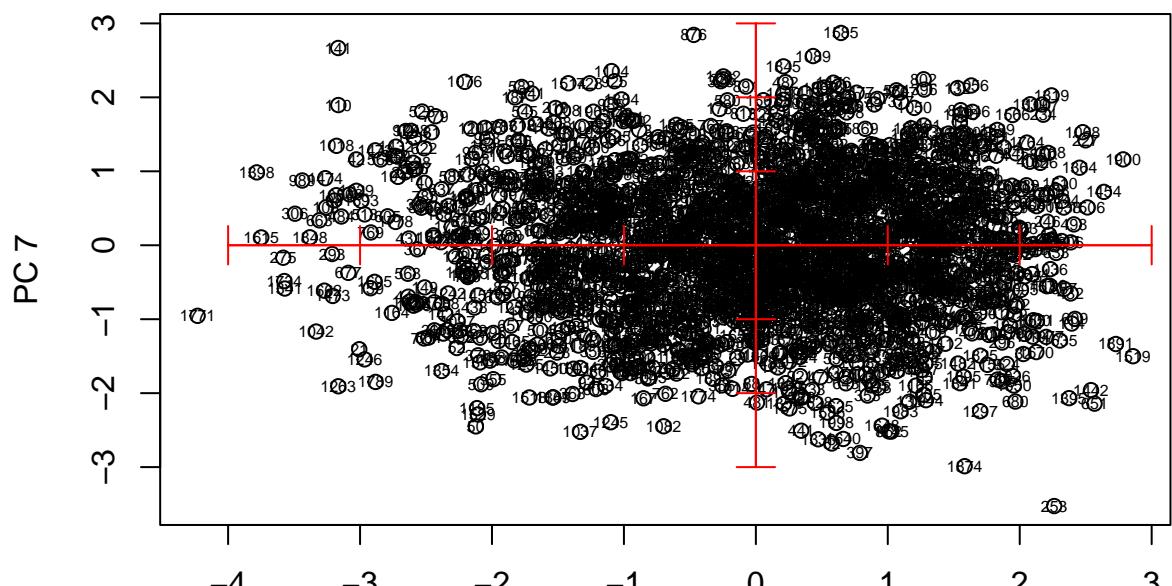
```
# PLOT OF INDIVIDUALS
for(i in 1:7) {
  for (j in (i+1):8) {
    eje1<-i
    eje2<-j

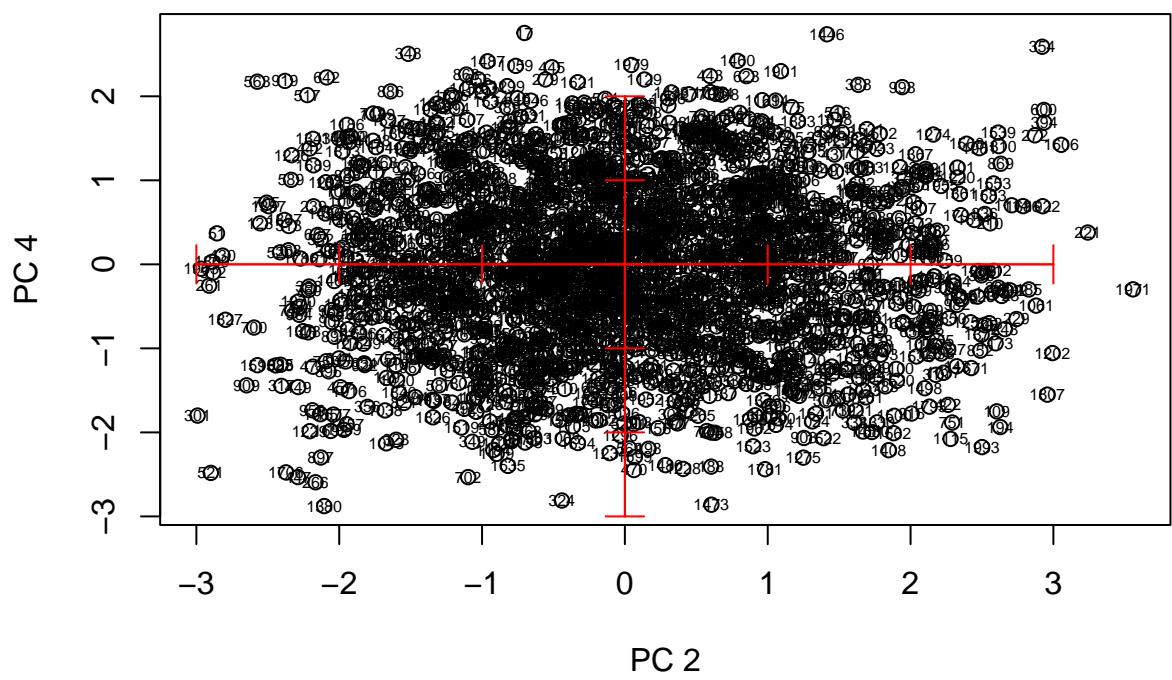
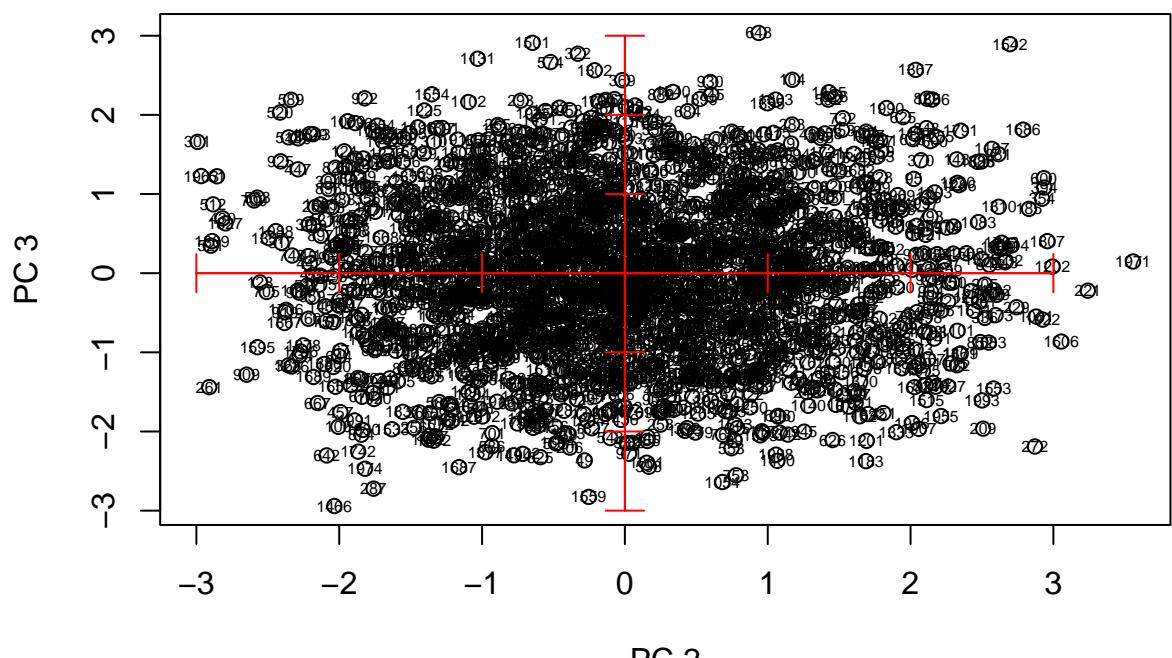
    plot(Psi[,eje1],Psi[,eje2], xlab= paste("PC",toString(i)) , ylab= paste("PC", toString(j)))
    text(Psi[,eje1],Psi[,eje2],labels=iden, cex=0.5)
    axis(side=1, pos= 0, labels = F, col="red")
    axis(side=3, pos= 0, labels = F, col="red")
    axis(side=2, pos= 0, labels = F, col="red")
    axis(side=4, pos= 0, labels = F, col="red")
  }
}
```

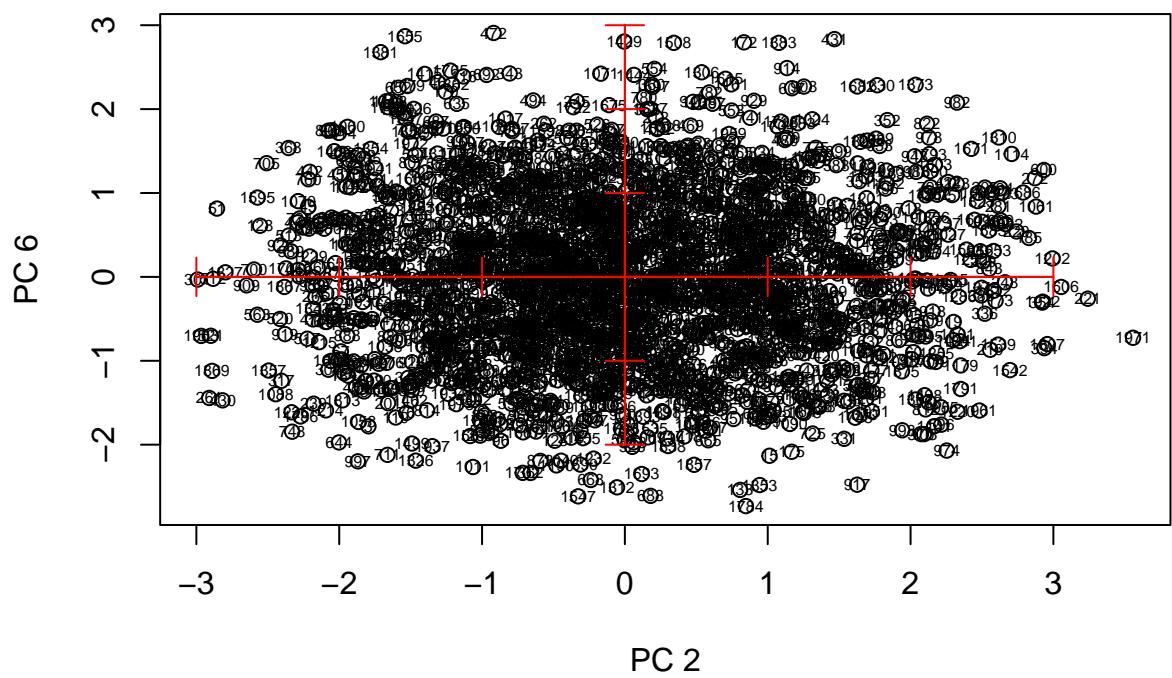
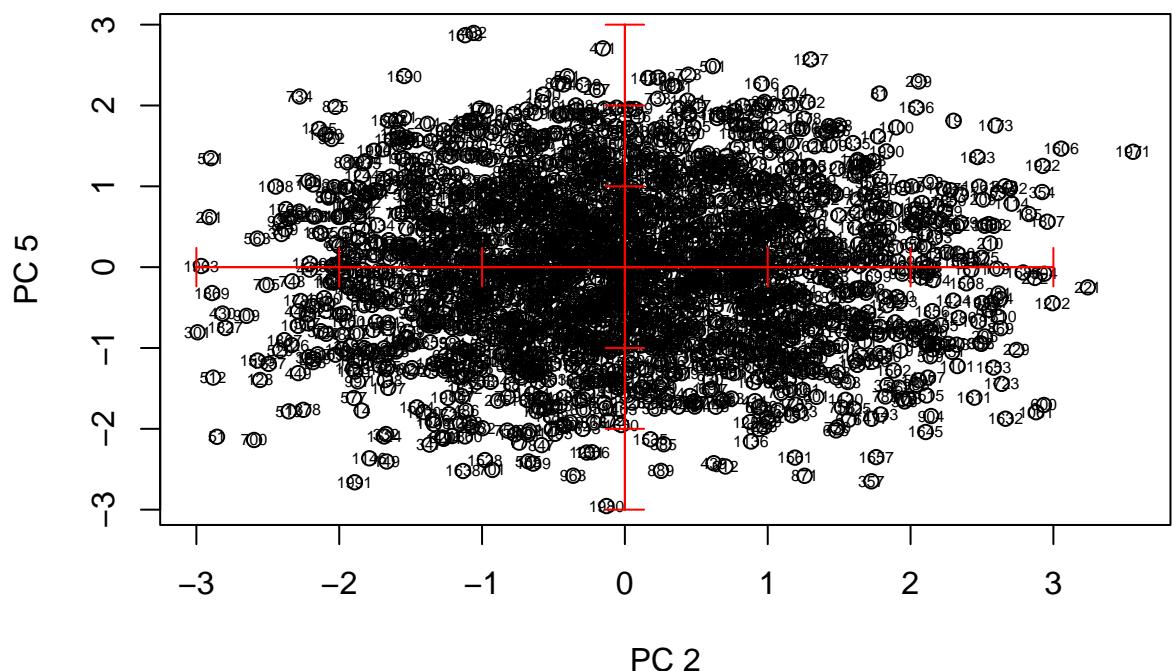


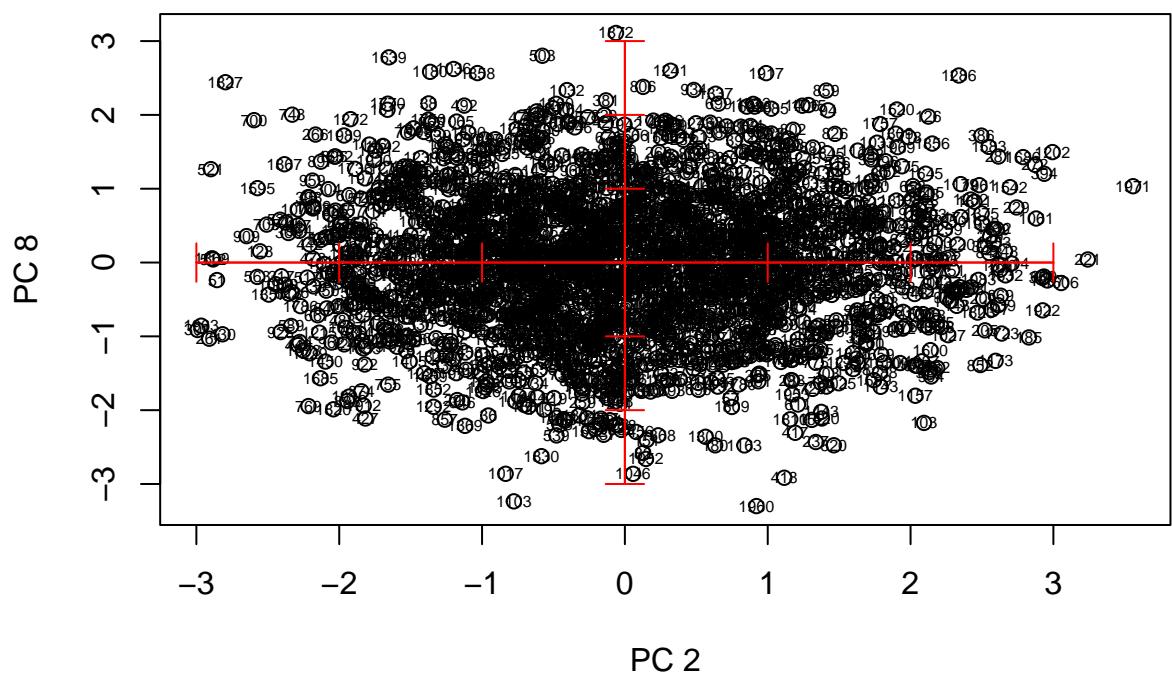
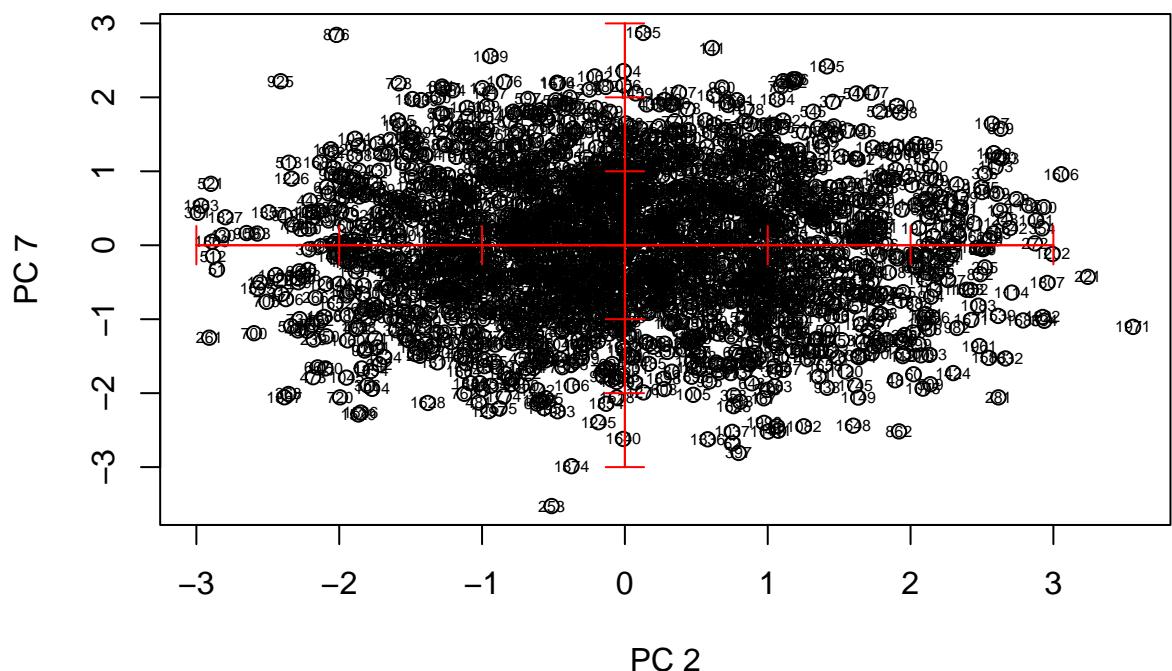


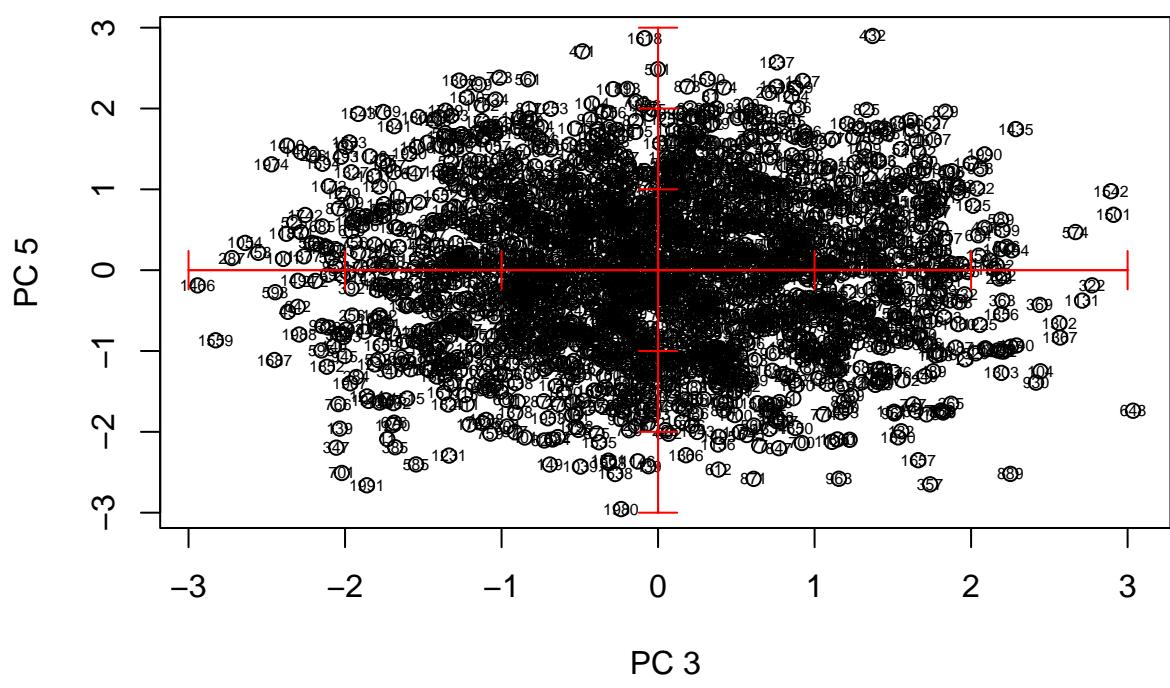
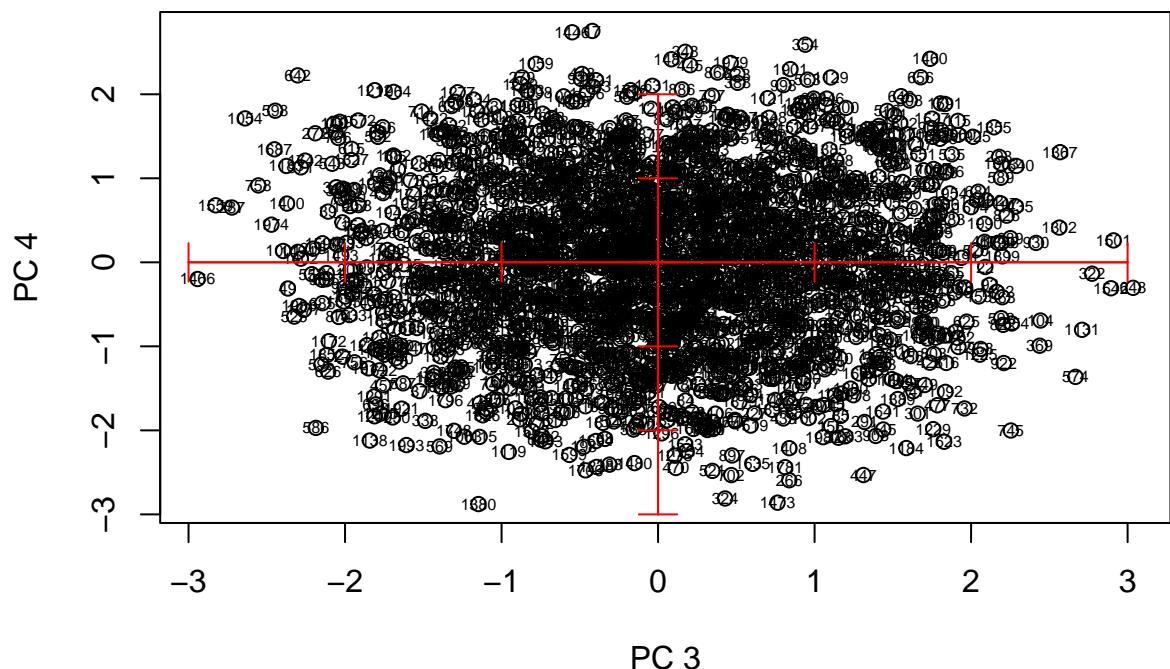


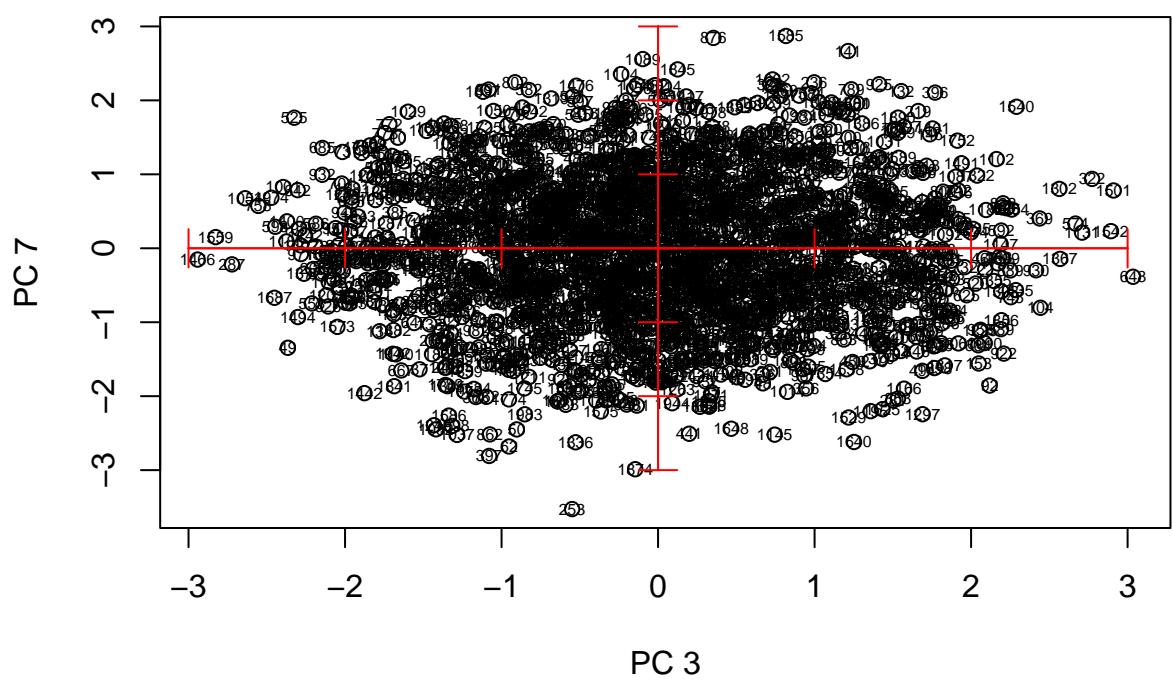
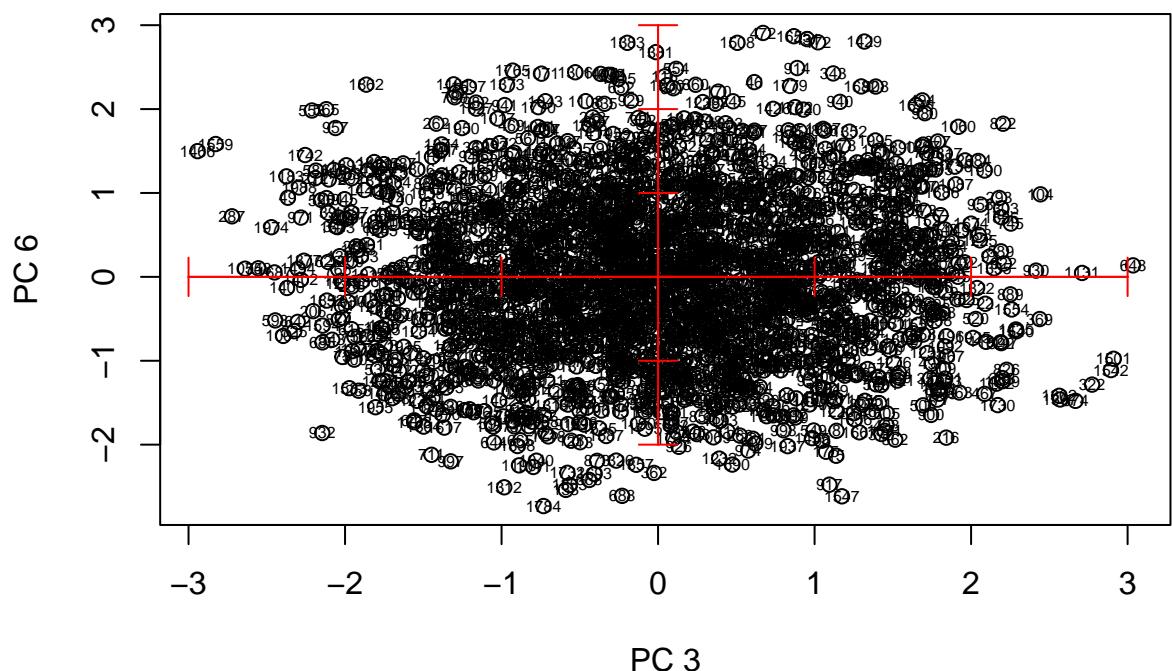


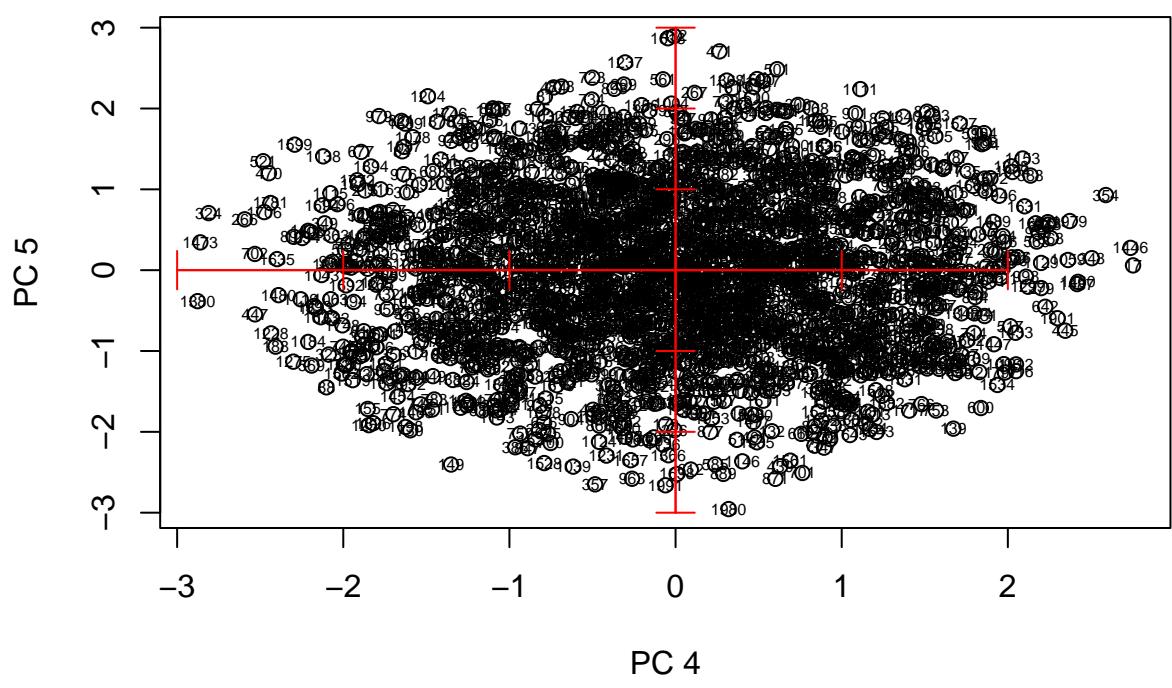
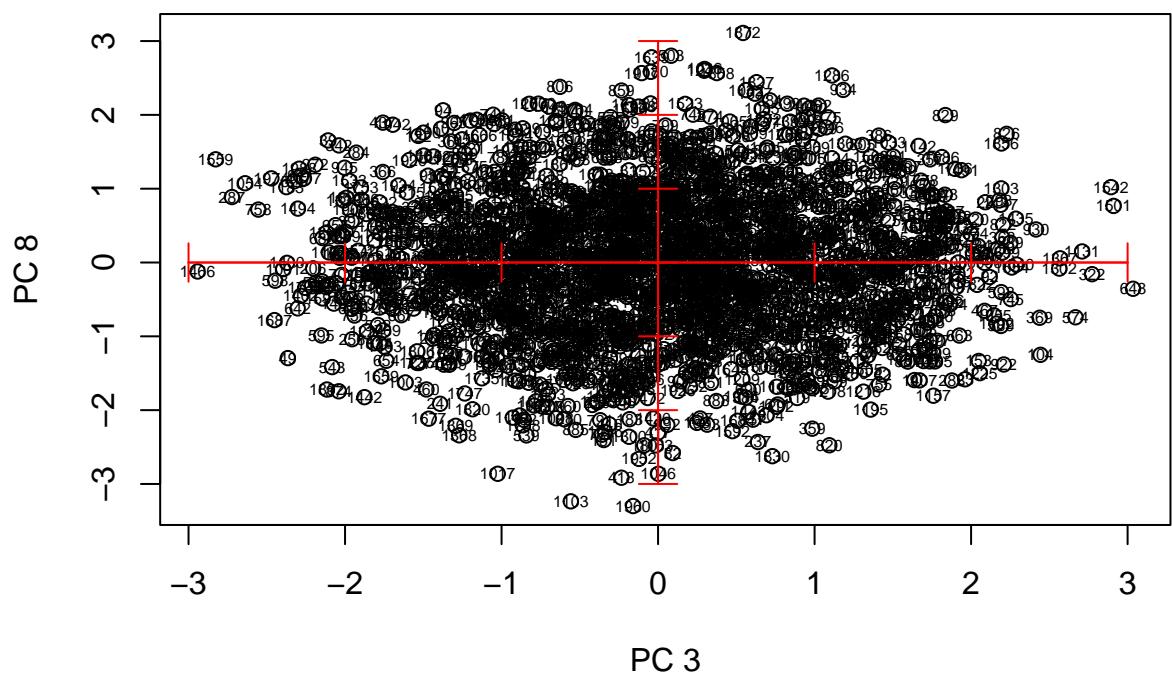


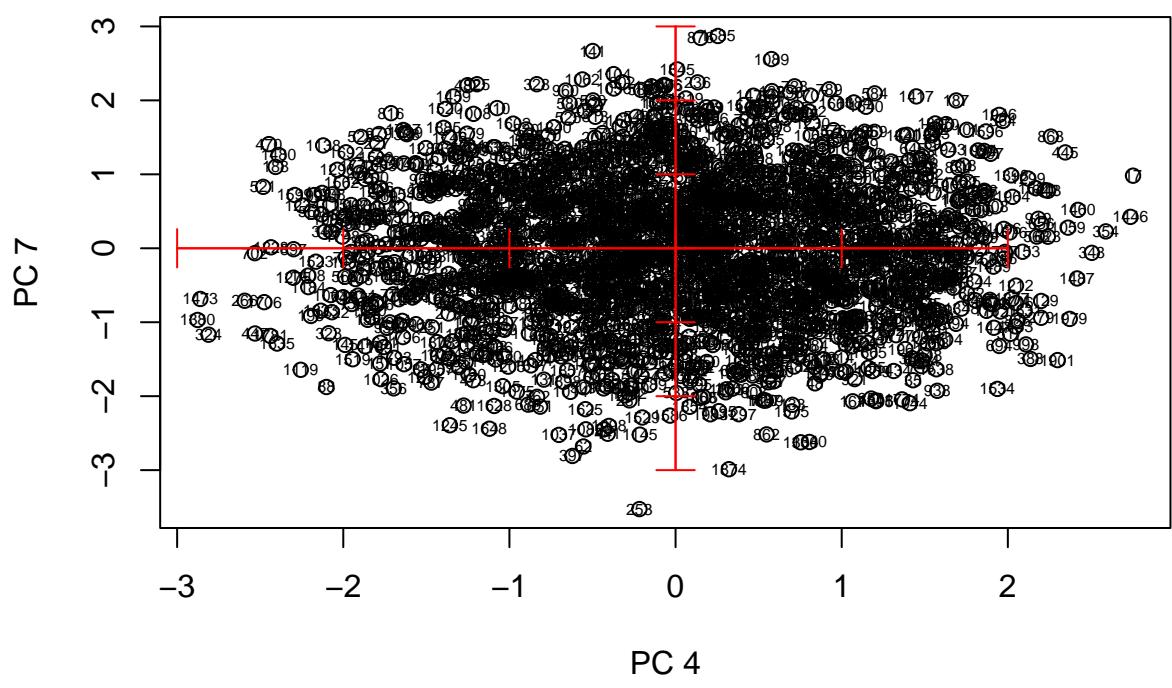
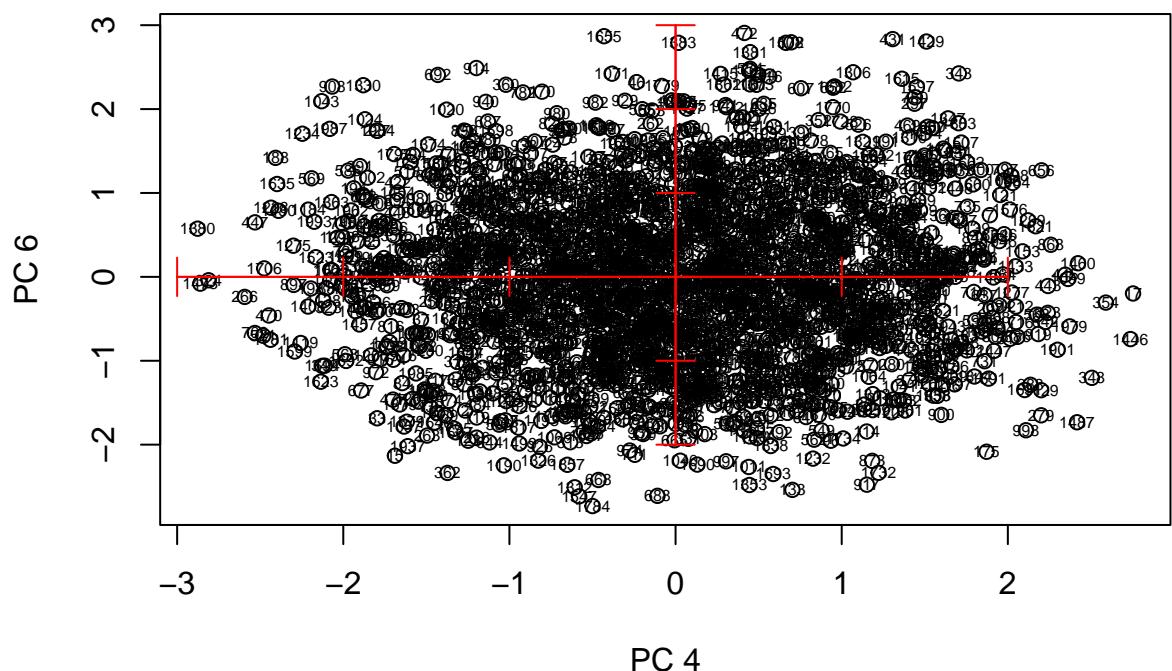


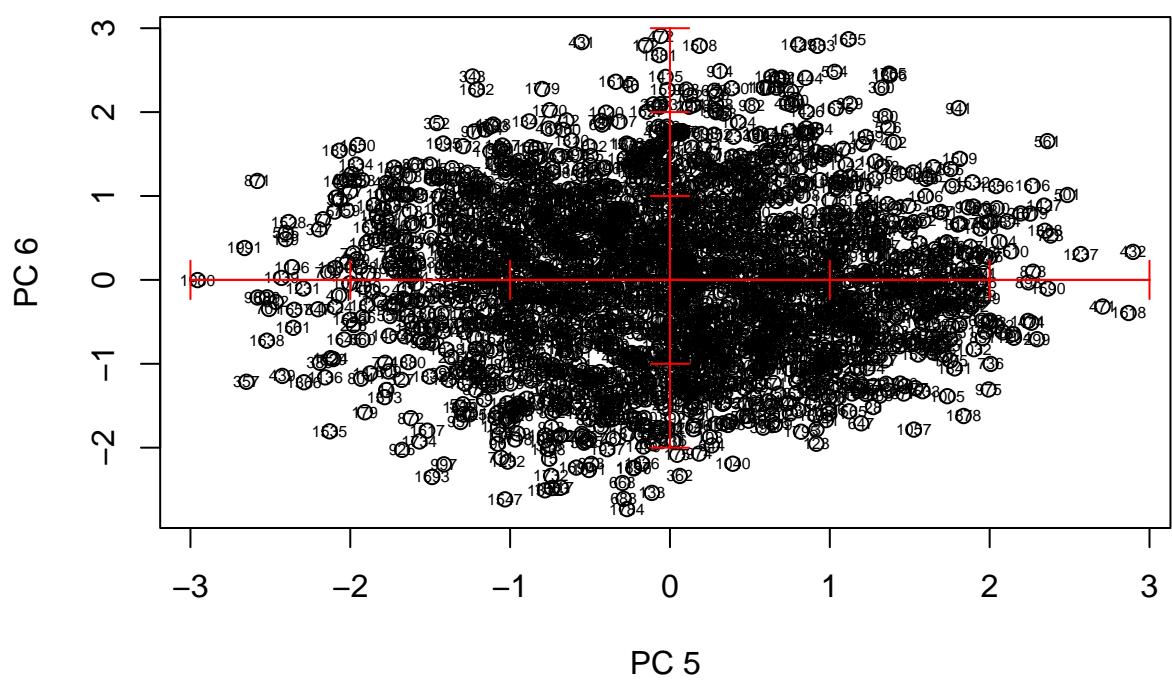
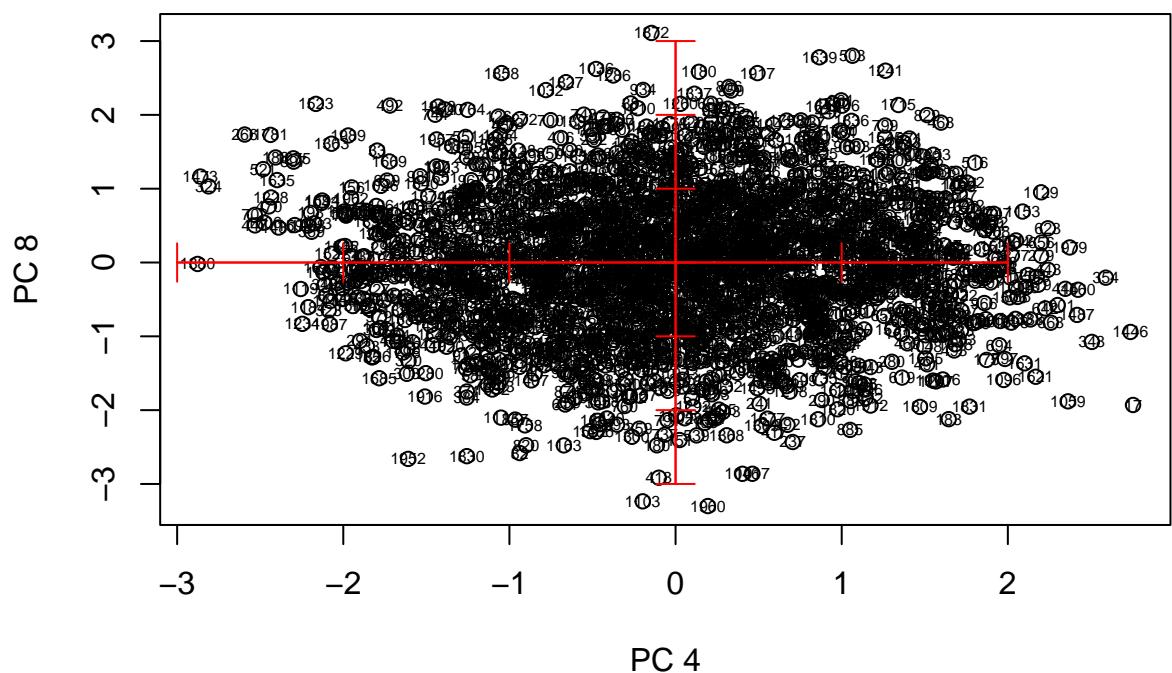


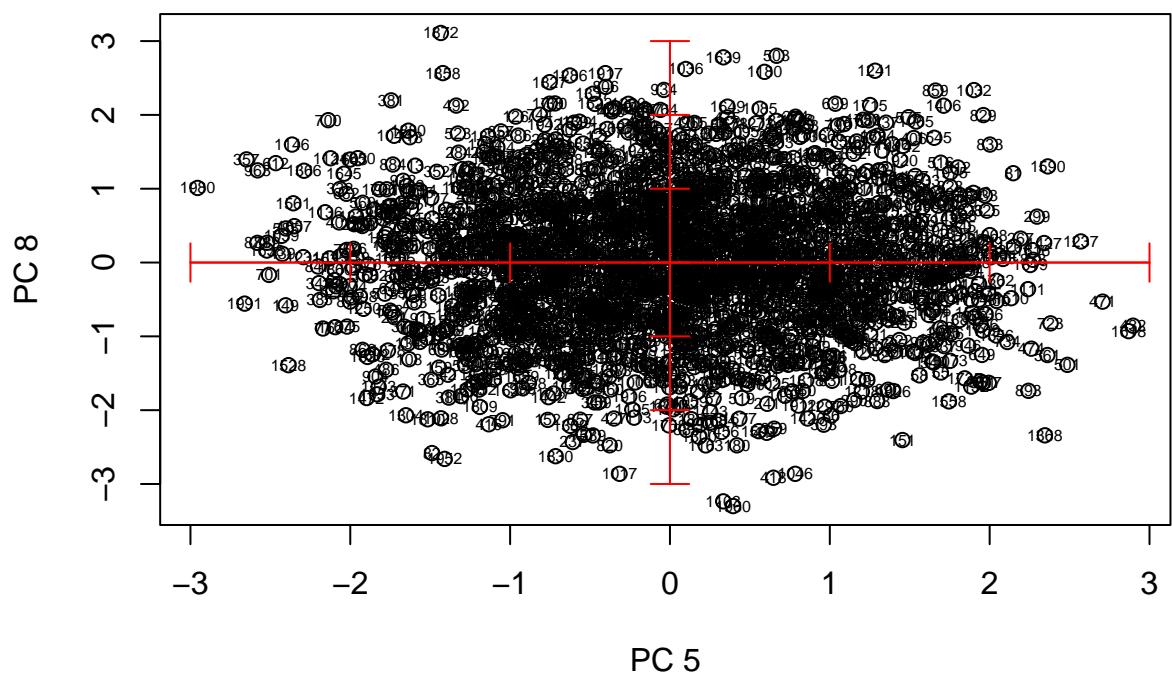
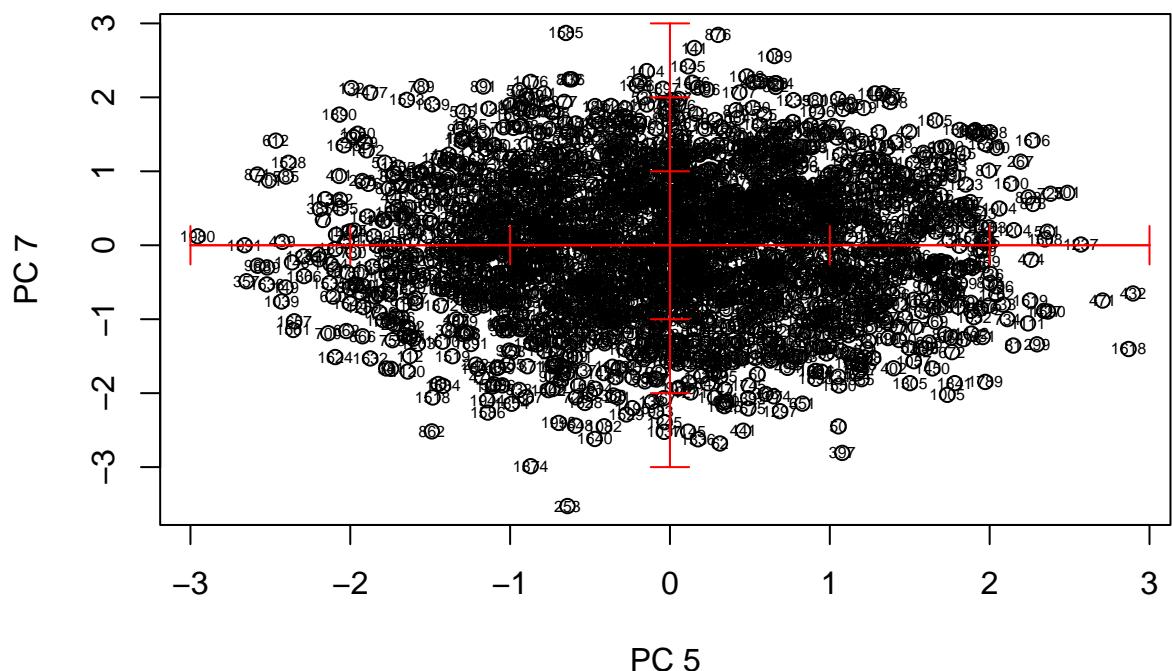


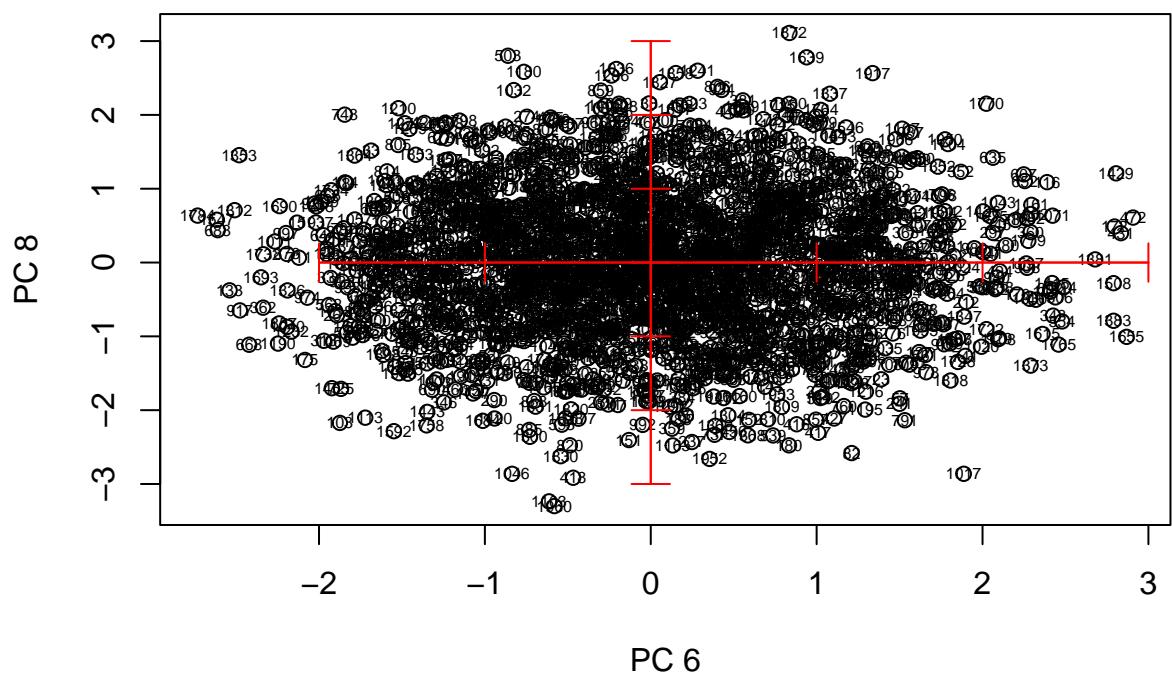
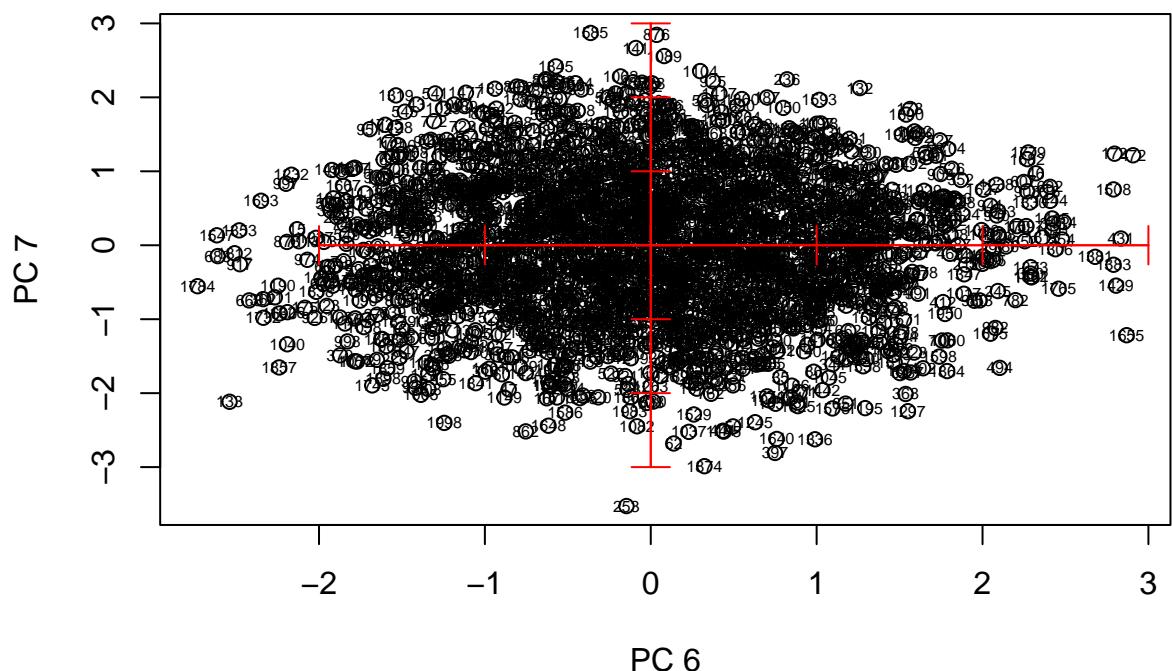


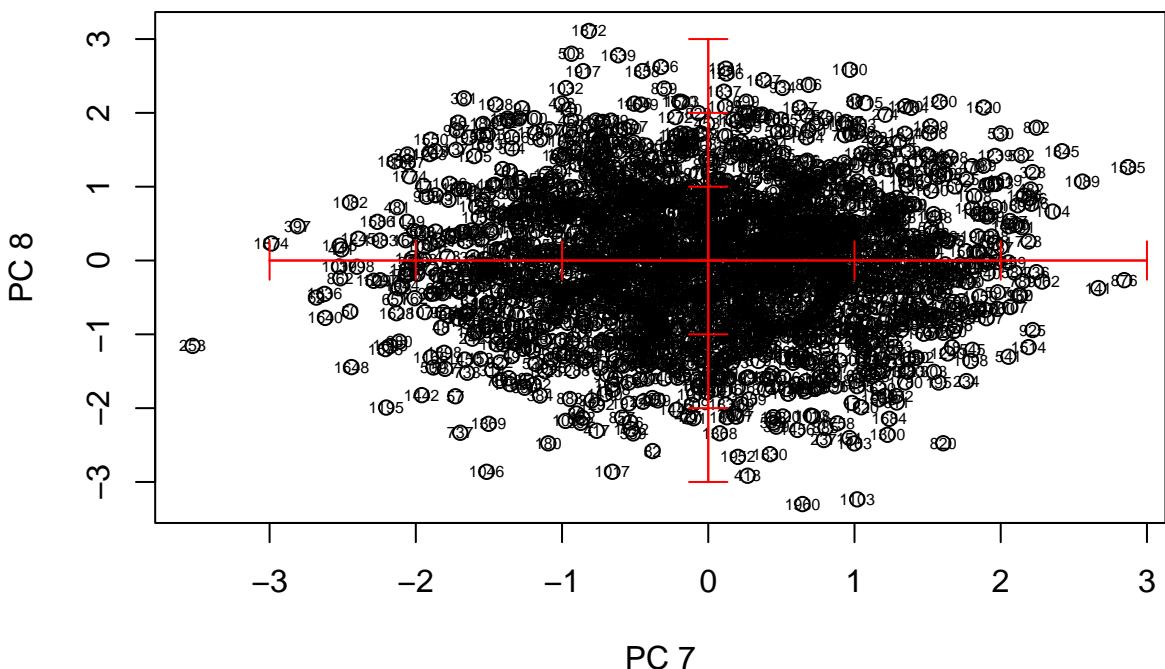












10.2 Common projection of numerical variables

```
#Projection of variables
```

```
Phi = cor(dcon,Psi)
```

```
Phi
```

```
##          PC1        PC2        PC3        PC4        PC5
## battery_power 0.01640159 -0.087044811 0.675136424 -0.10410063 0.003205500
## clock_speed   0.04591799 -0.003938079 -0.007911906 -0.17223617 -0.525493569
## int_memory    -0.02697513  0.075856925 -0.010832110 -0.69758774 0.304625000
## m_dep         -0.03399688 -0.155436228  0.439120225 -0.03600170 0.096994693
## mobile_wt      0.03875342 -0.120922209  0.193645404  0.23286370 -0.658168589
## n_cores        -0.04767184  0.041176868 -0.195343835  0.61940177 0.341168203
## px_height     -0.78516477 -0.366849948 -0.029020262 -0.05435268 -0.017230914
## px_width       -0.77823632 -0.382218433 -0.052099998  0.01958075 -0.004728977
## sc_h           -0.40628630  0.724315129  0.069085441 -0.02298042 -0.037258178
## sc_w           -0.39921454  0.715434035  0.141401023  0.07618390 -0.102055153
## talk_time      0.01466959 -0.039769657  0.565503284  0.25346171 0.291316252
##          PC6        PC7        PC8
## battery_power 0.24623920 0.20603299 0.3899900078
## clock_speed   0.71110404 -0.36680154 0.0272808139
## int_memory    0.04073068 -0.28985750 -0.4677308225
## m_dep         -0.39787907 -0.68374334 0.2639948034
## mobile_wt     -0.34535702 -0.09999104 -0.4656108471
## n_cores        0.28263686 -0.45576165 -0.0433676358
## px_height     0.02519647  0.04809261 -0.0044741718
## px_width       0.04852283  0.02090369 -0.0276701074
## sc_h           -0.09389957 -0.00530107 0.0009137262
## sc_w           0.01653851 -0.03507744 0.0300431490
## talk_time      0.25470237  0.16489173 -0.5451135382
```

```
#select your axis
```

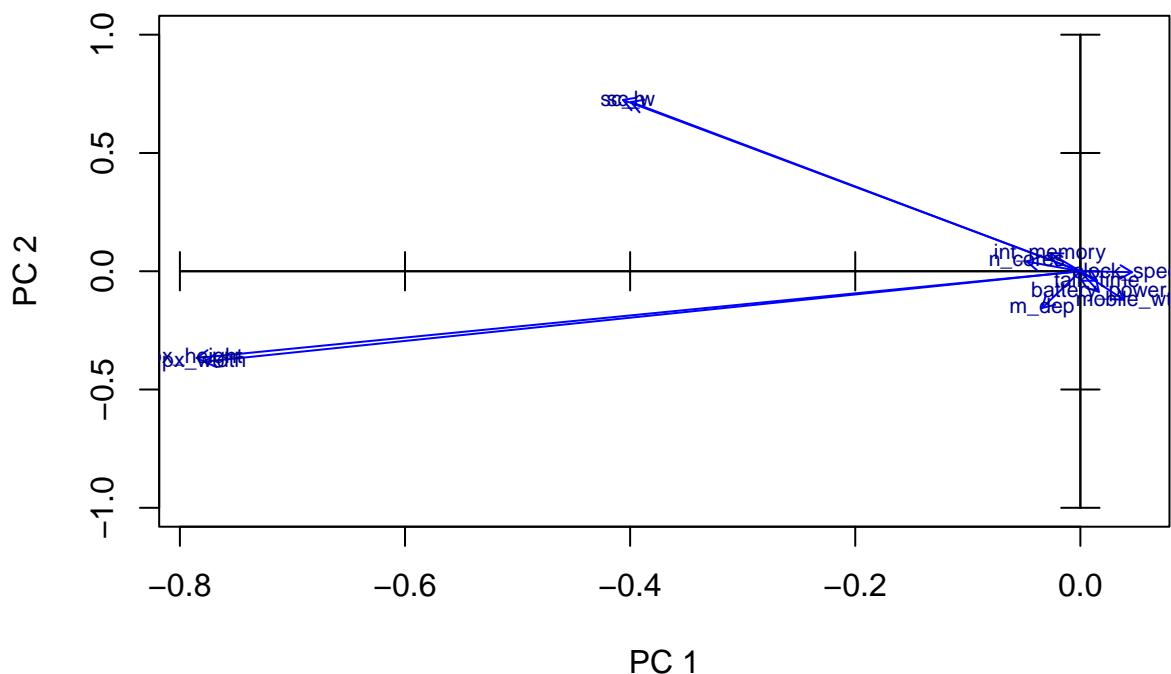
```
for(i in 1:7) {
```

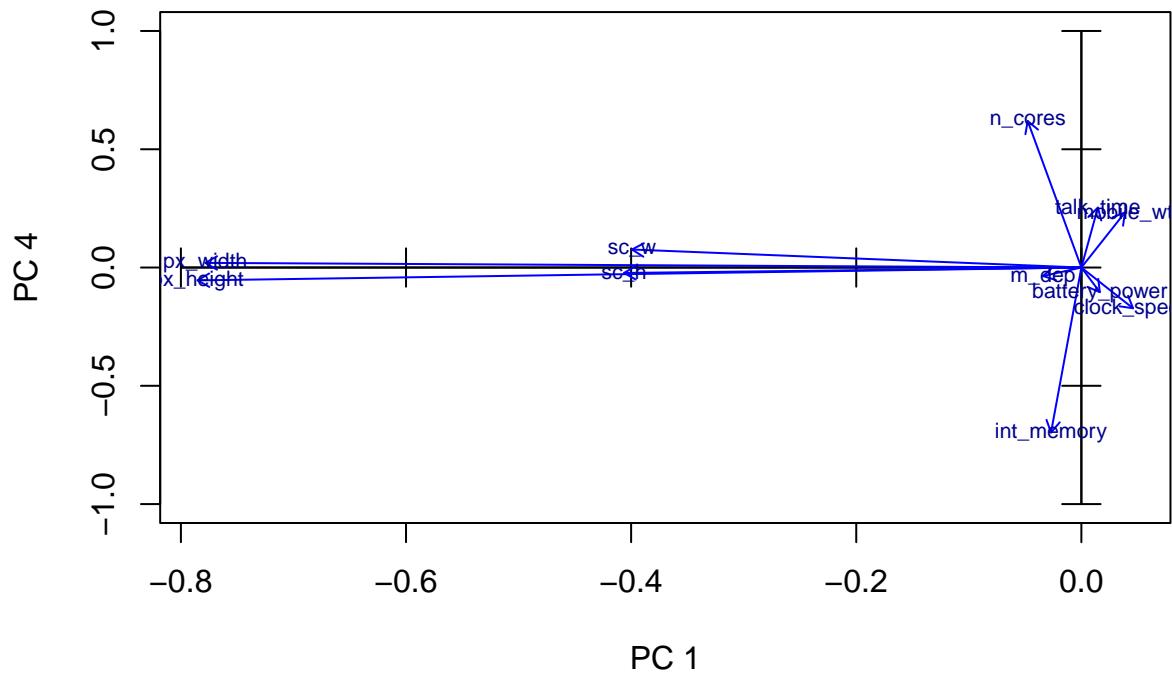
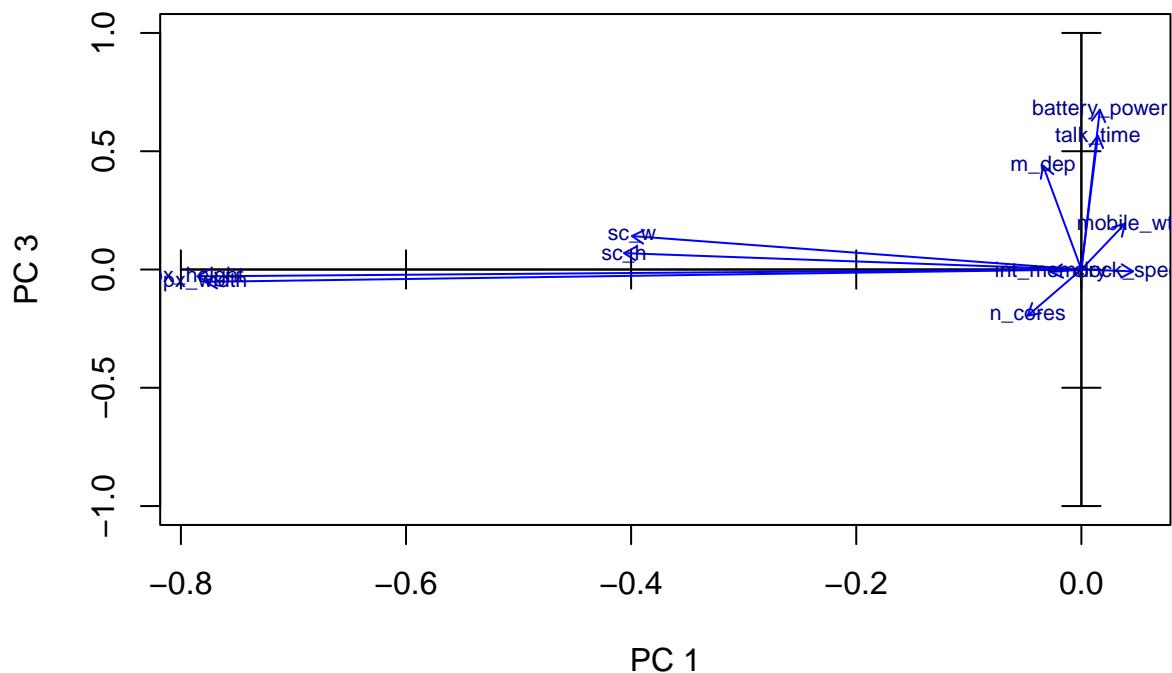
```

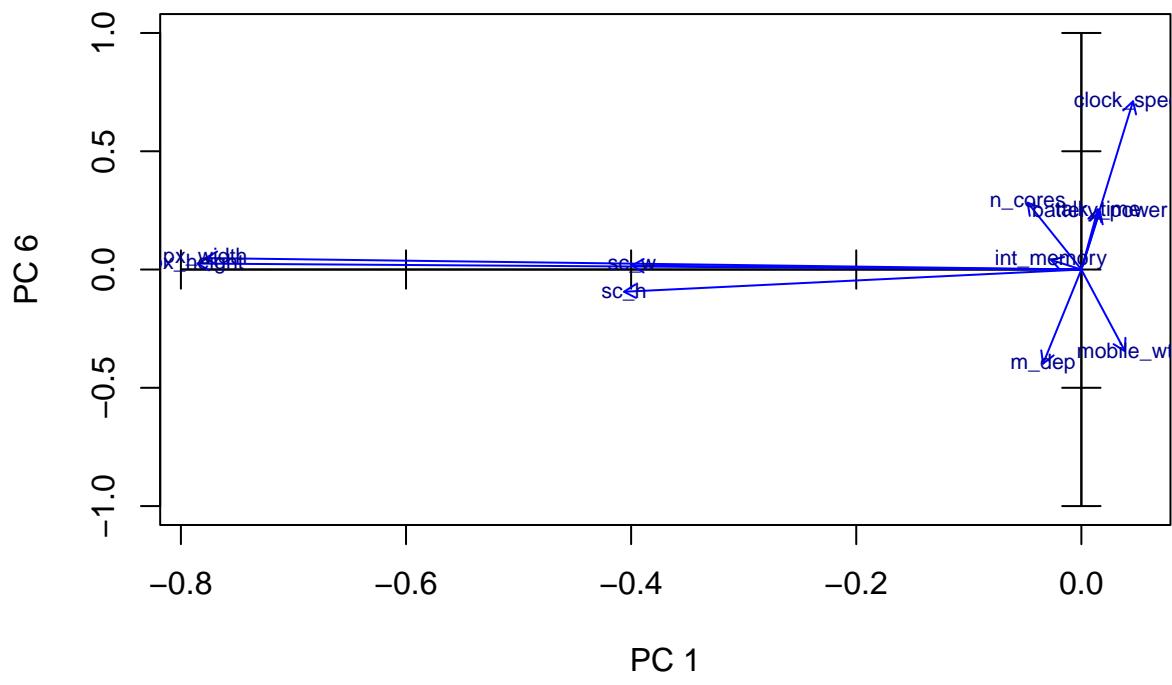
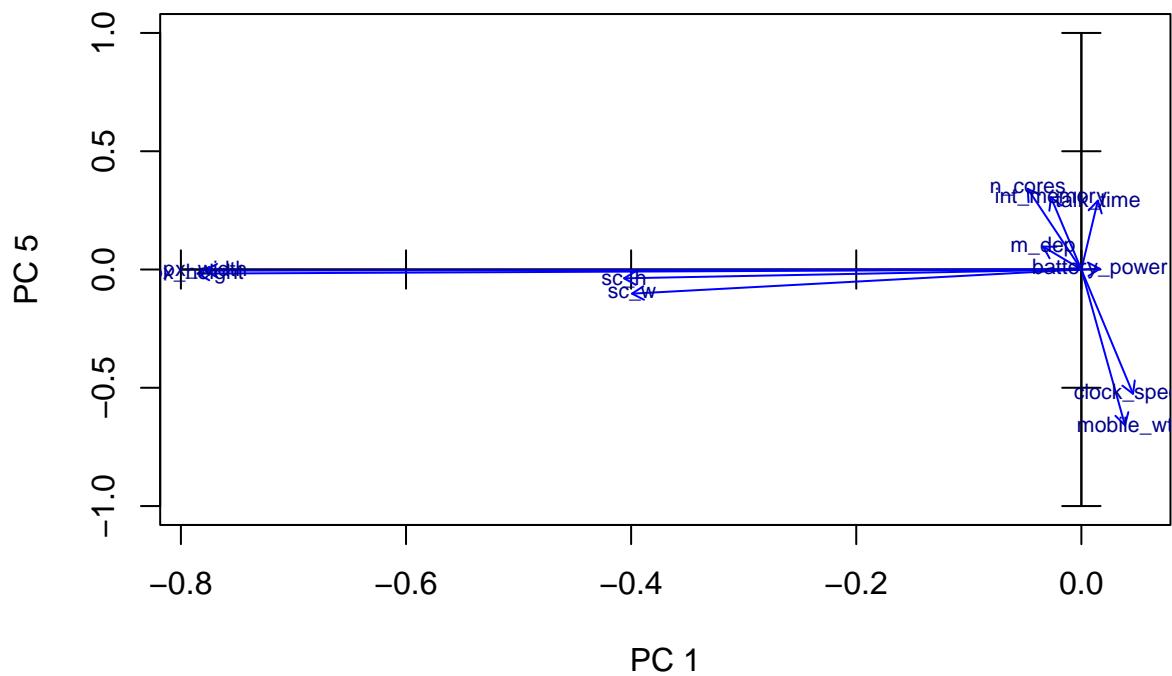
for (j in (i+1):8) {
  X<-Phi[,i]
  Y<-Phi[,j]

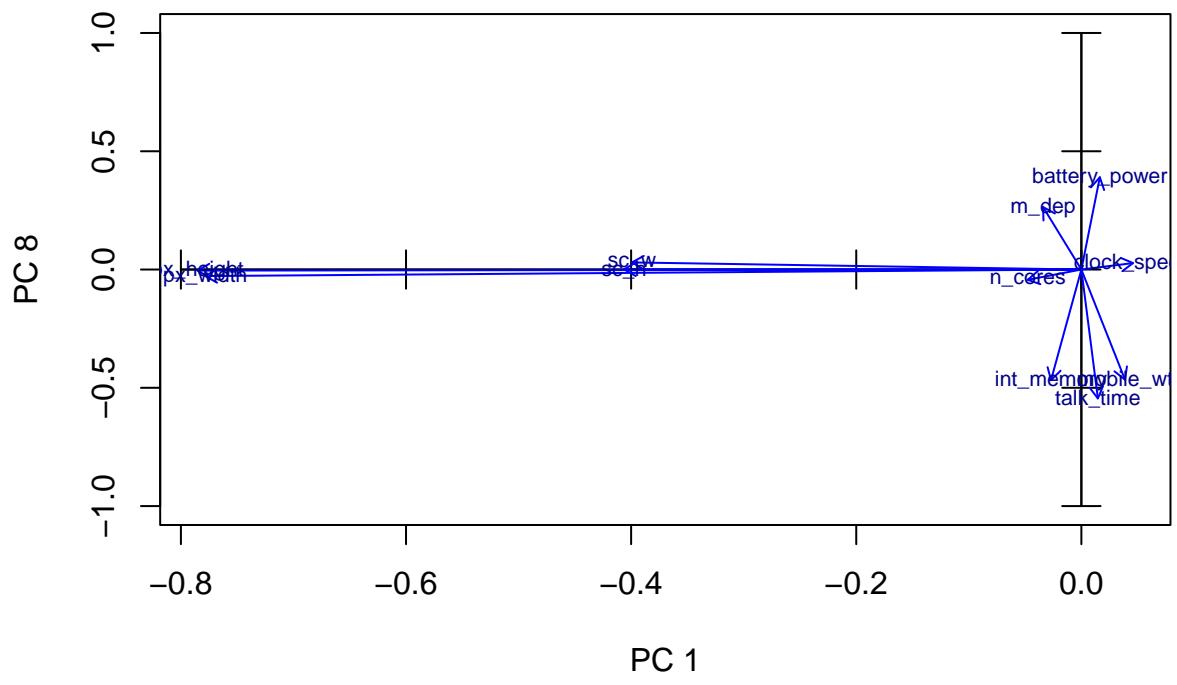
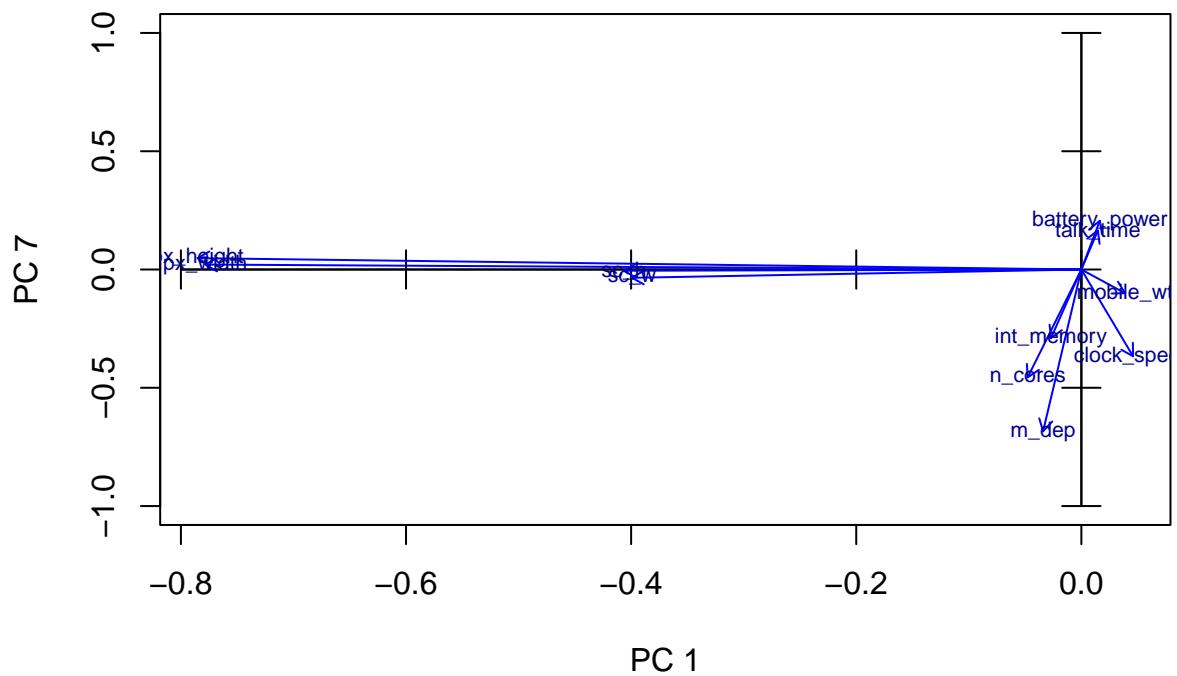
  #zoom
  plot(X, Y, xlab=paste("PC",toString(i)) , ylab=paste("PC" , toString(j)), type="n" , xlim=c(min(X,0),max(X,0)), ylim=c(min(Y,0),max(Y,0)))
  axis(side=1, pos= 0, labels = F)
  axis(side=3, pos= 0, labels = F)
  axis(side=2, pos= 0, labels = F)
  axis(side=4, pos= 0, labels = F)
  arrows(ze, ze, X, Y, length = 0.07,col="blue")
  text(X,Y,labels=etiq,col="darkblue", cex=0.7)
}
}

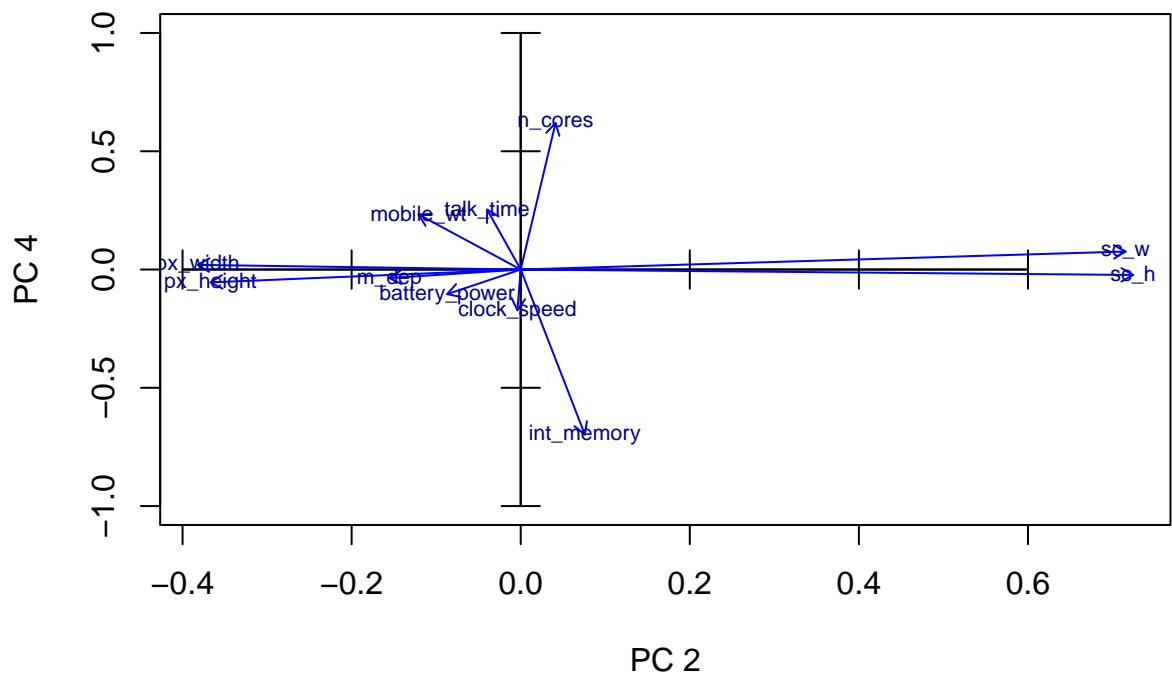
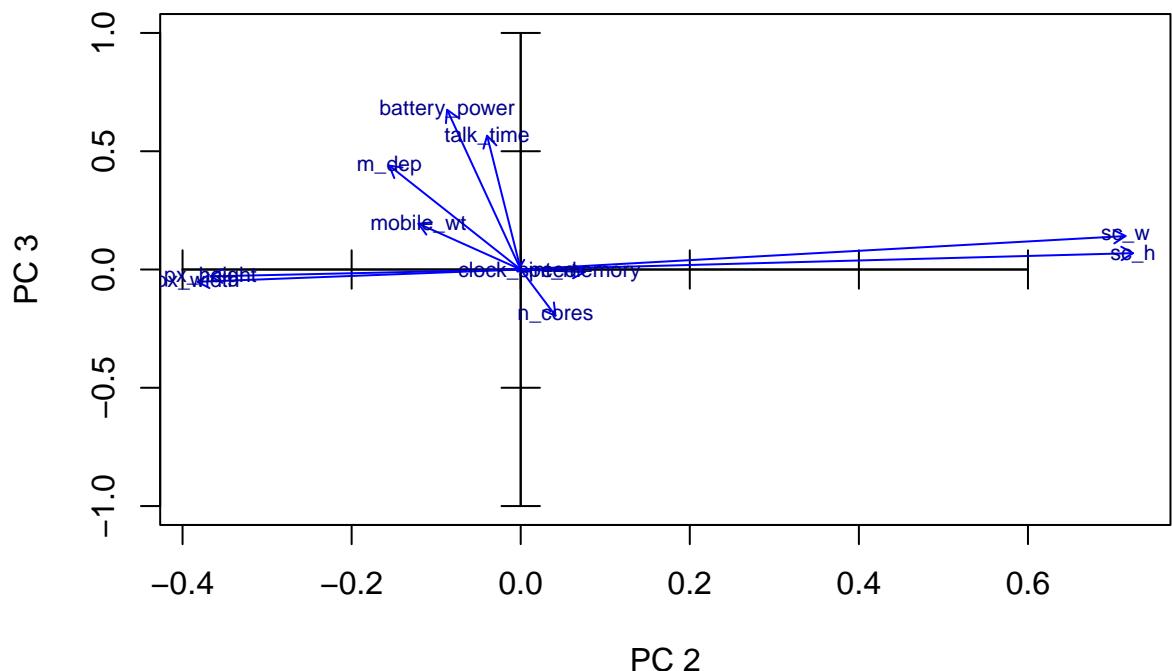
```

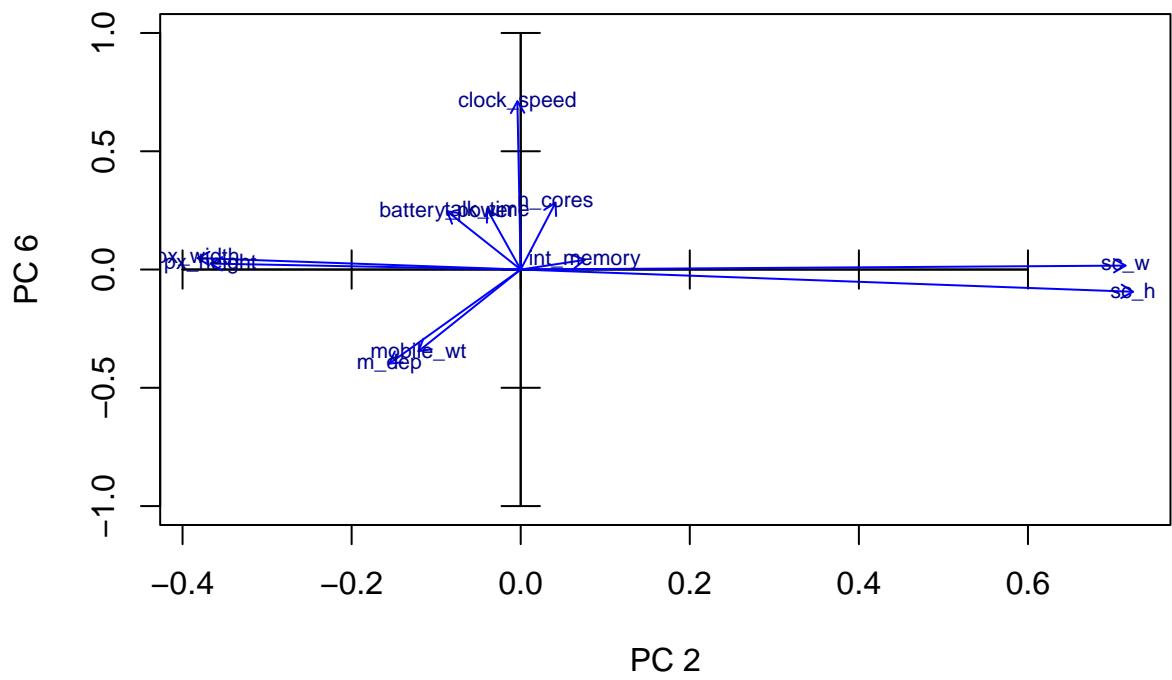
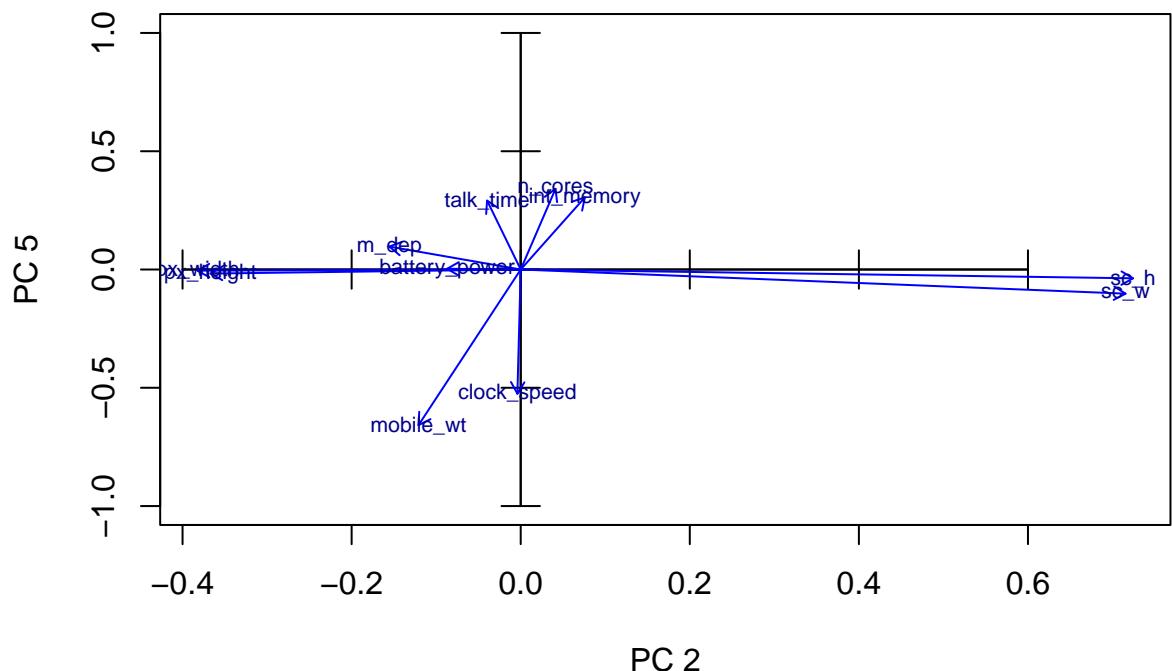


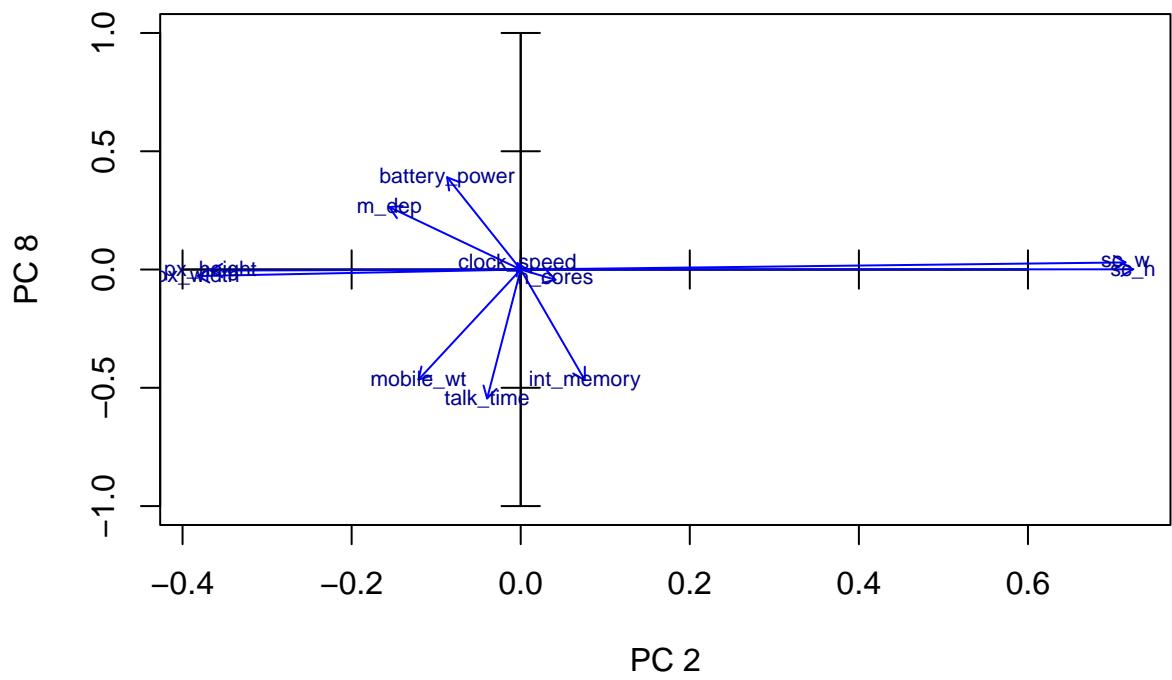
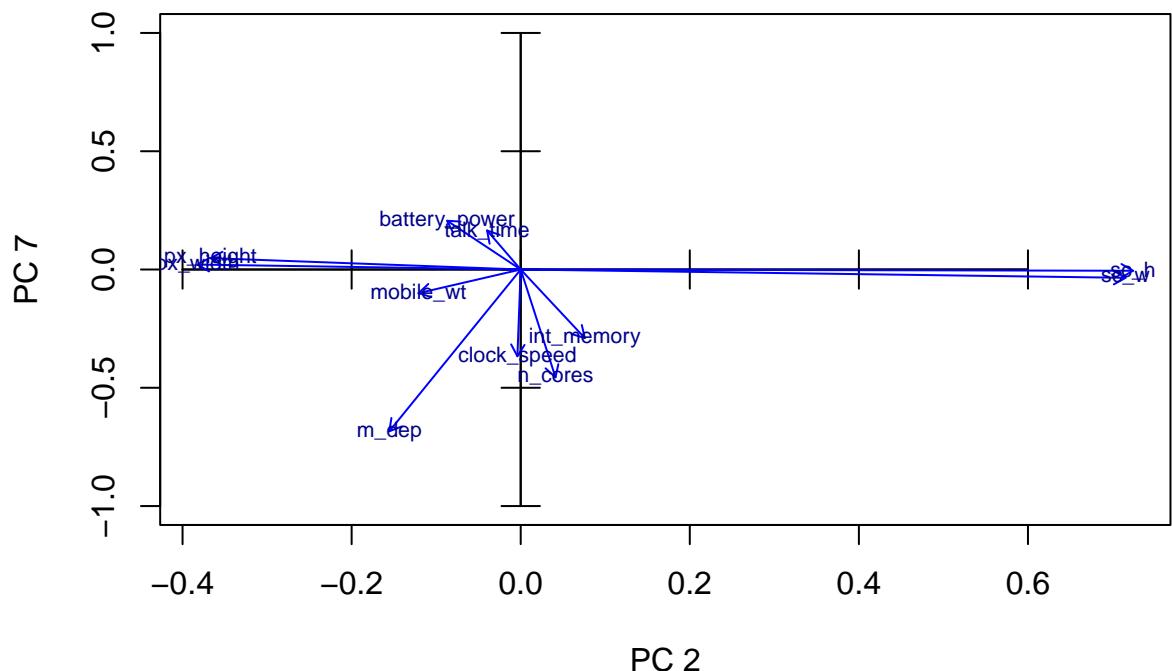


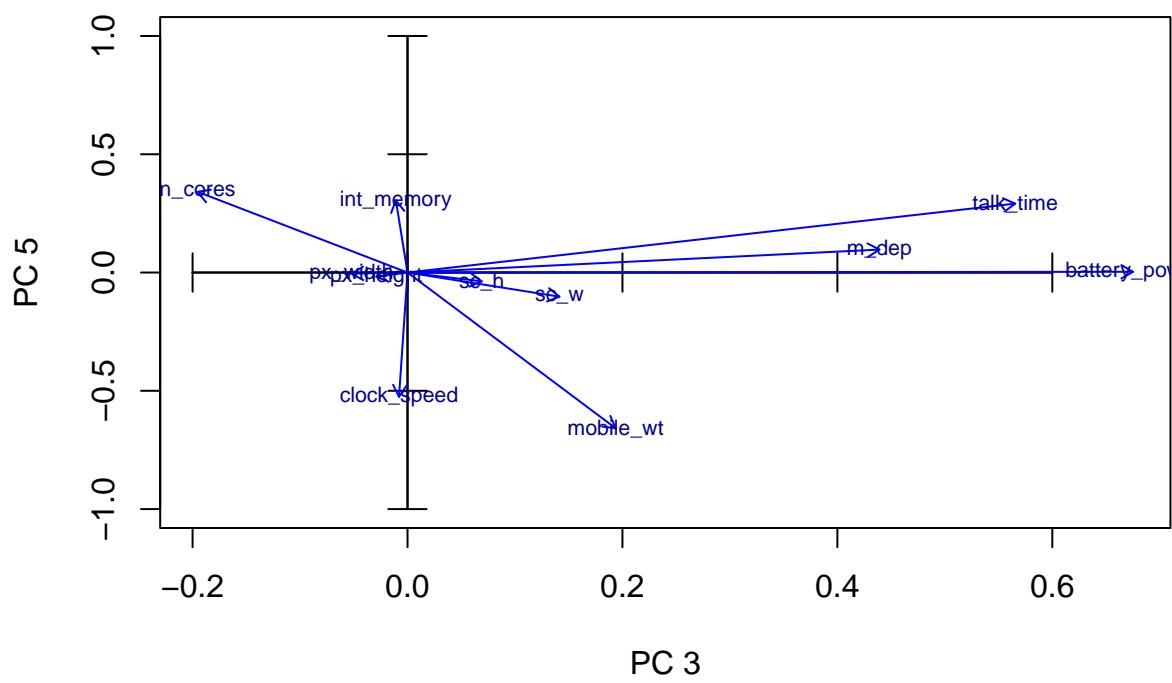
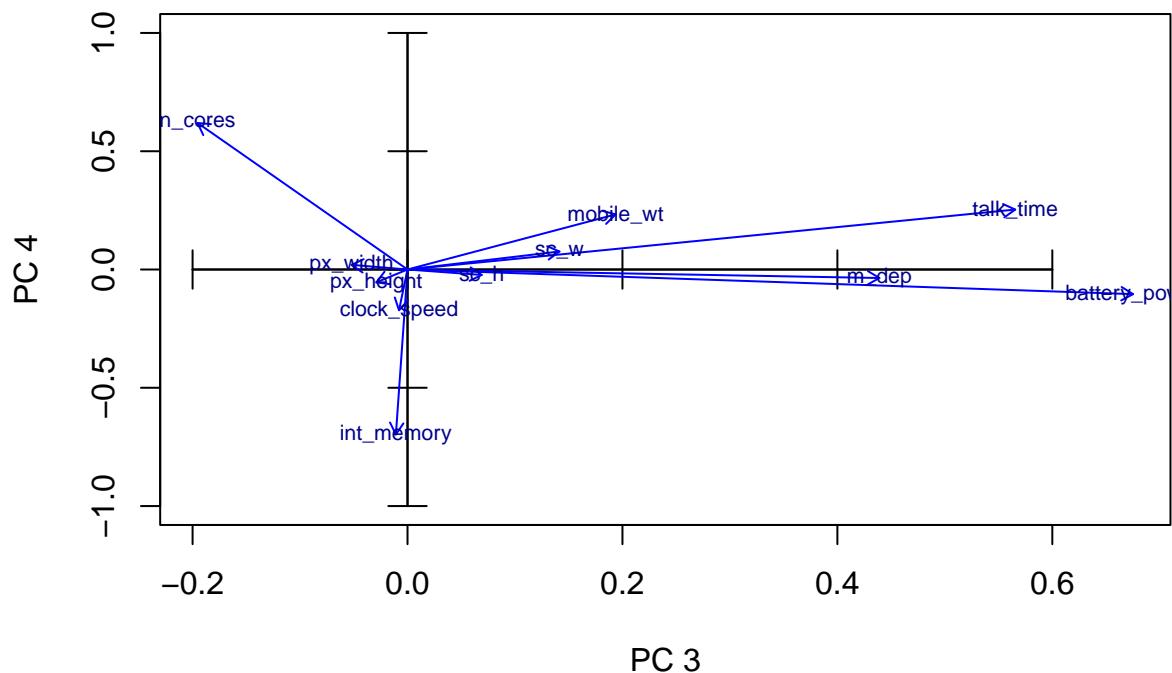


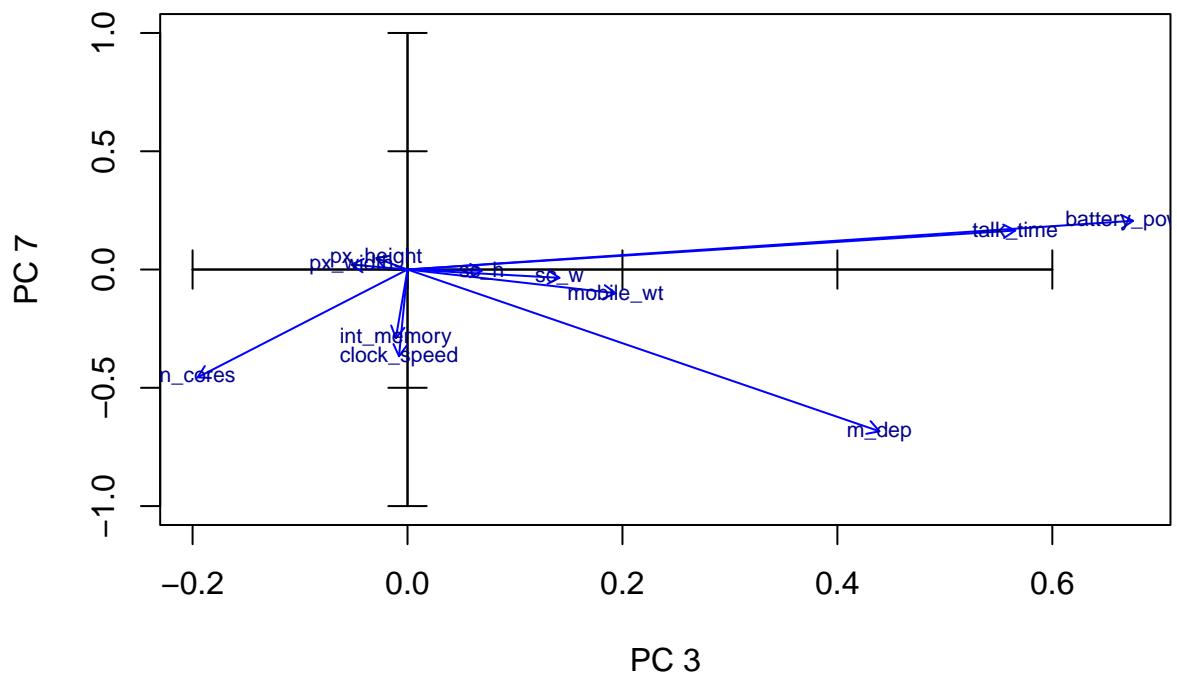
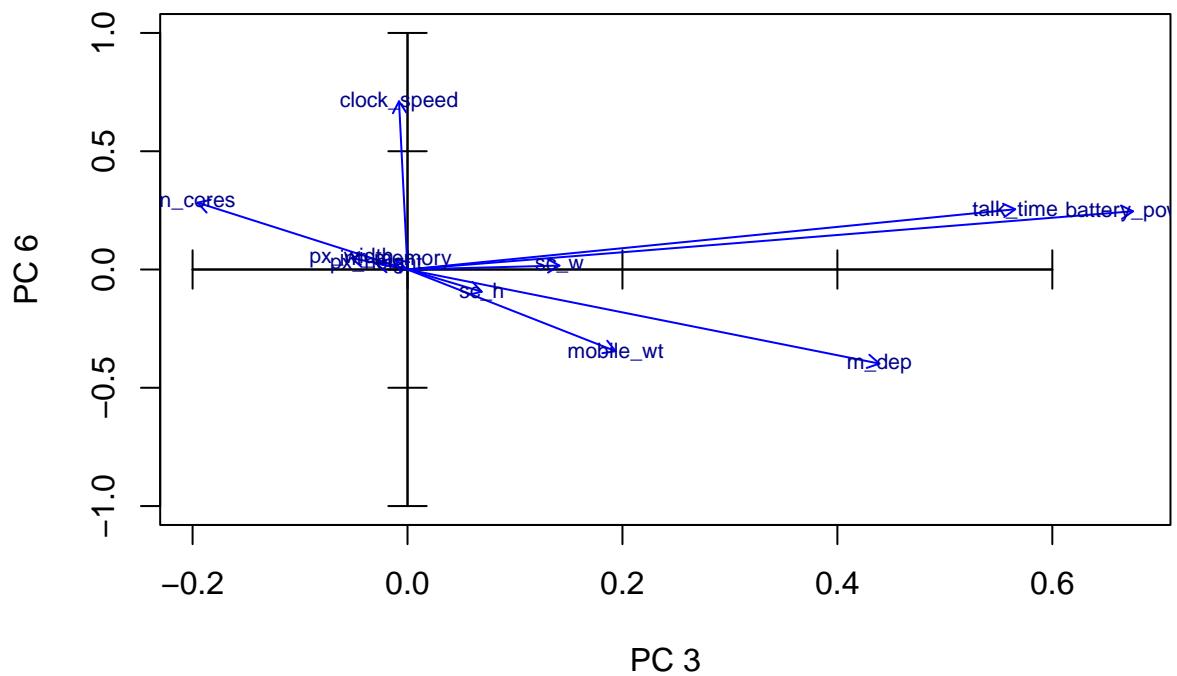


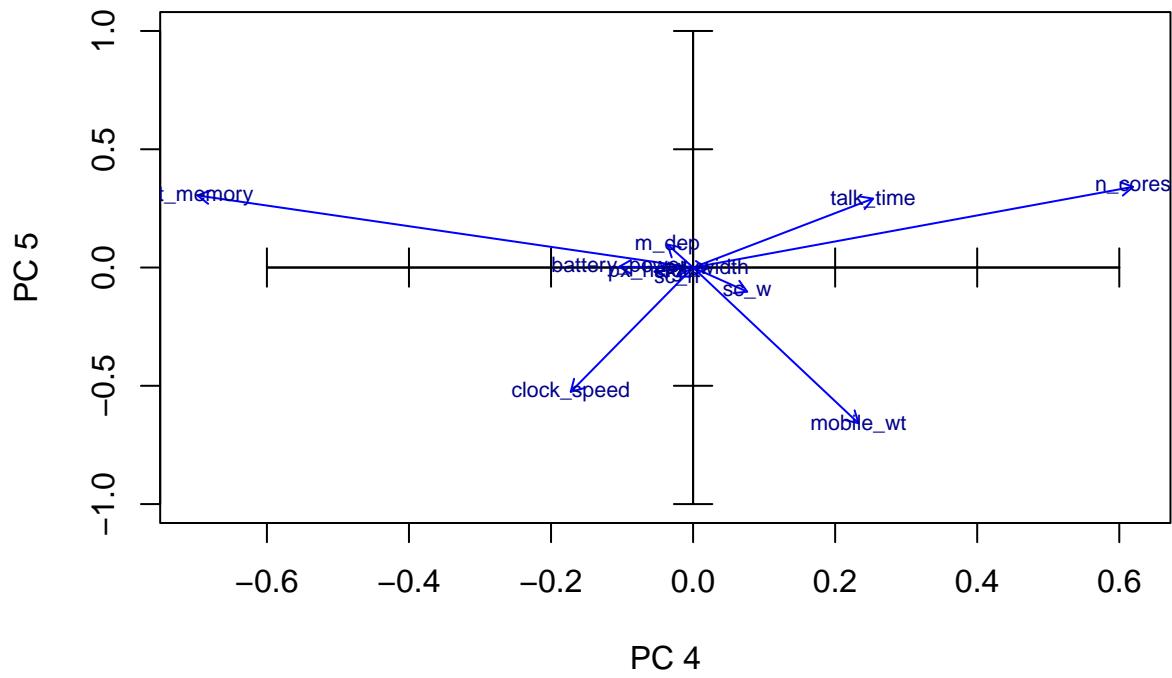
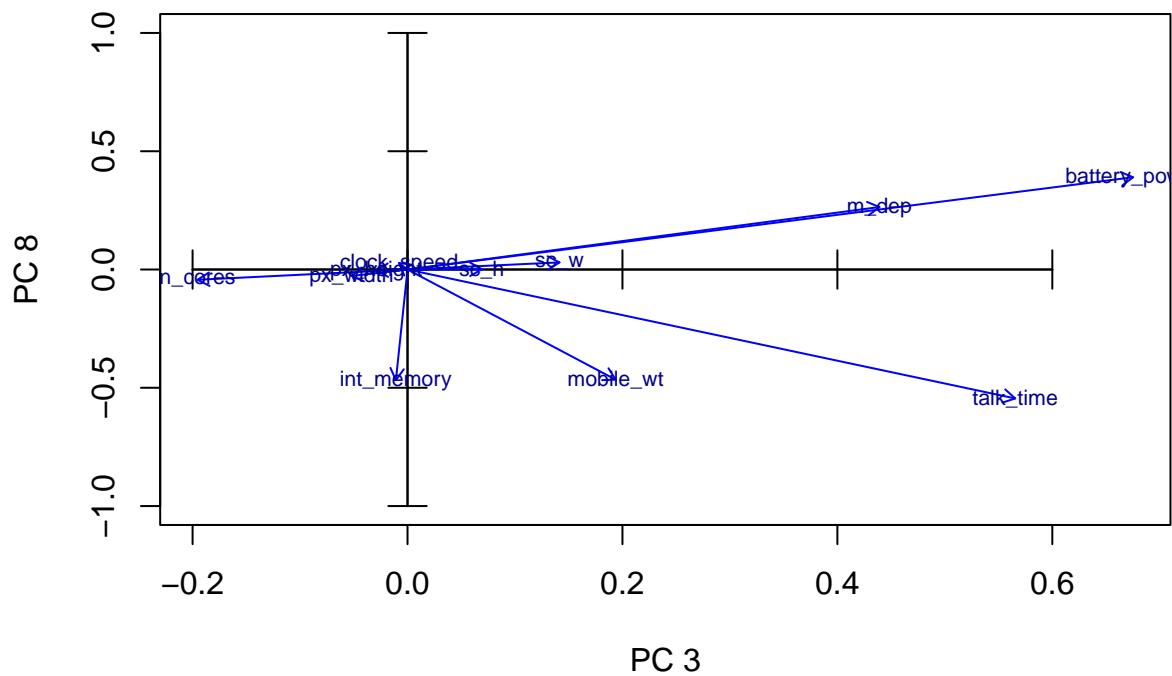


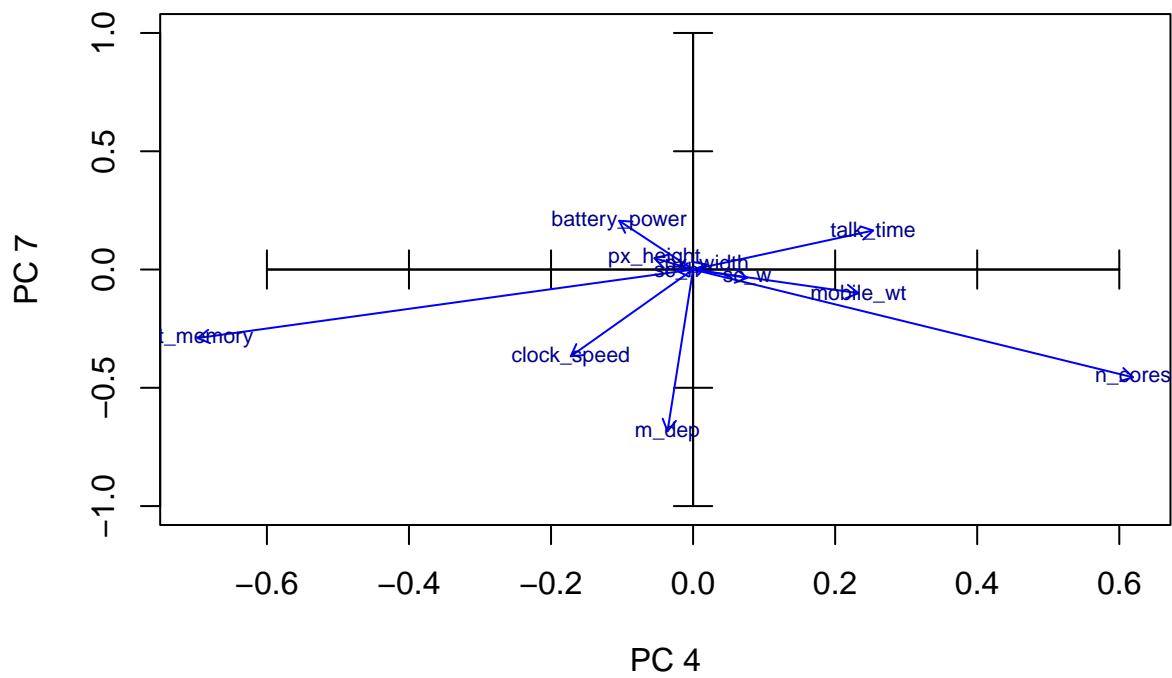
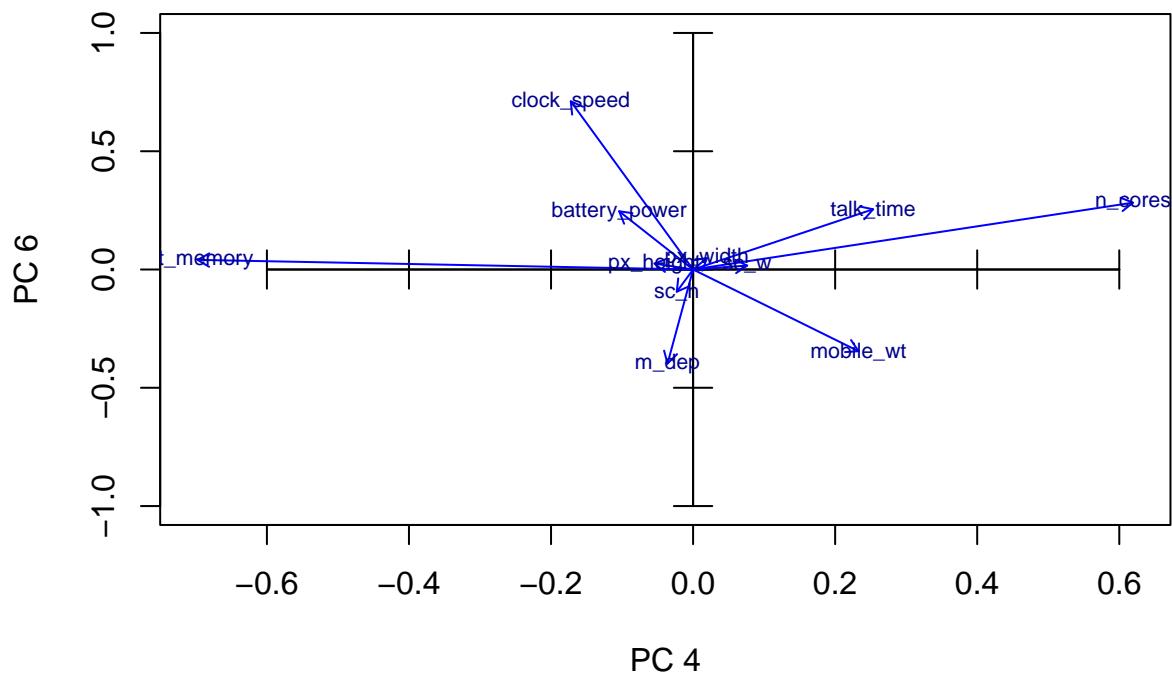


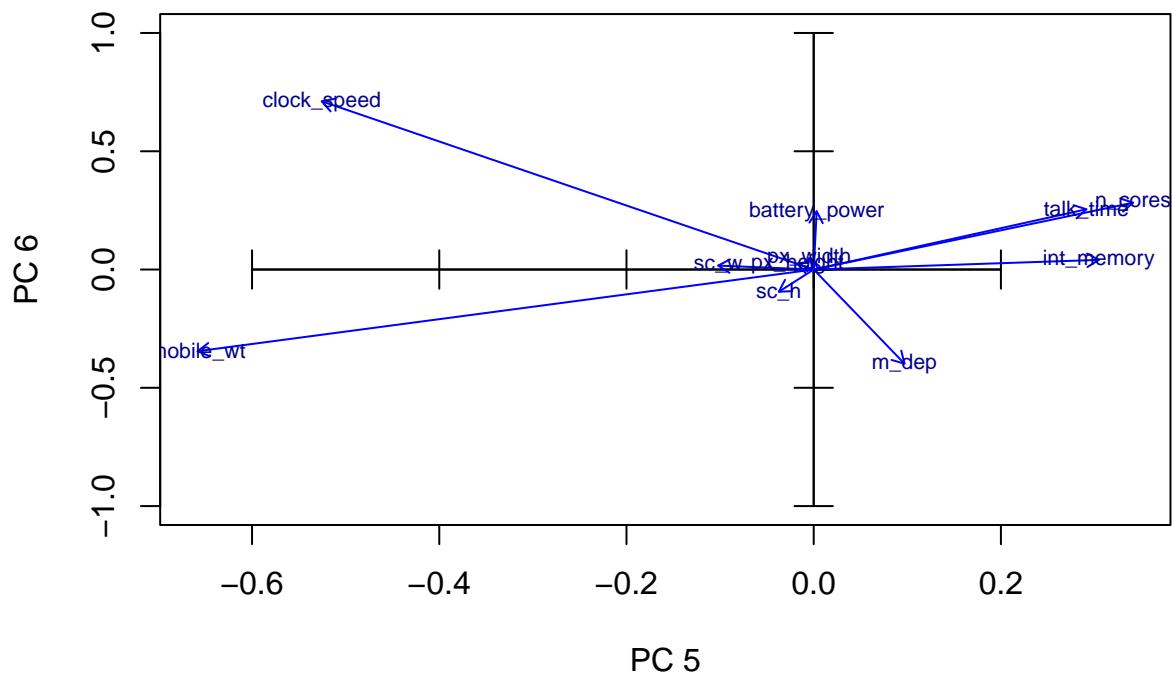
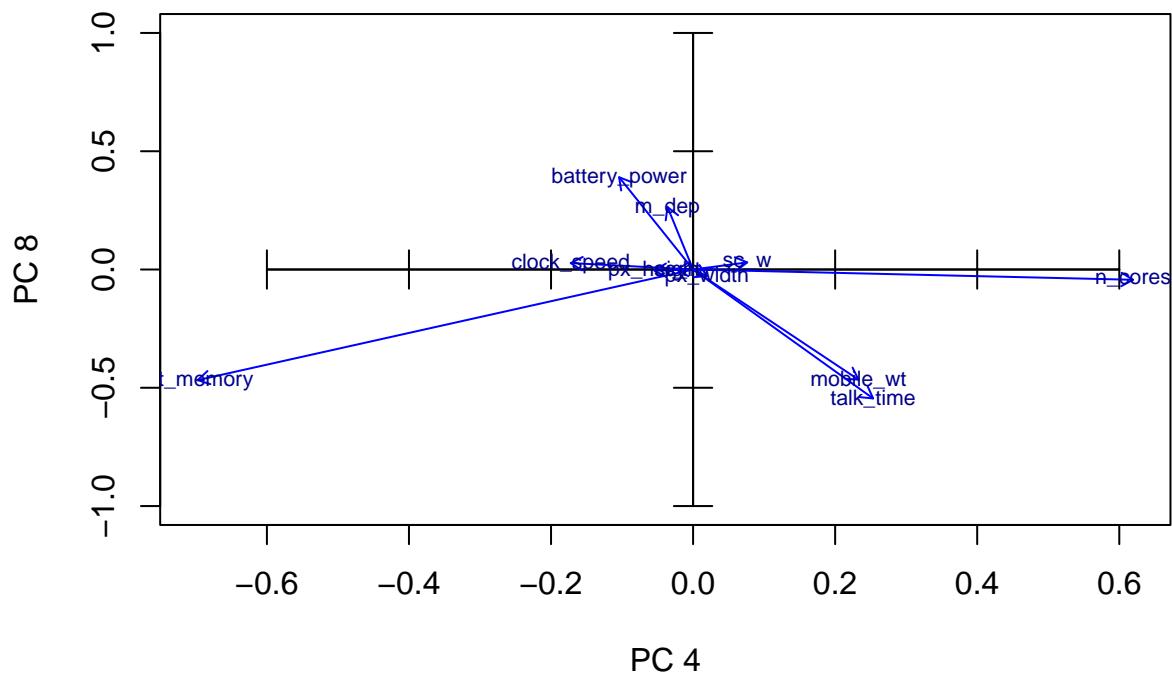


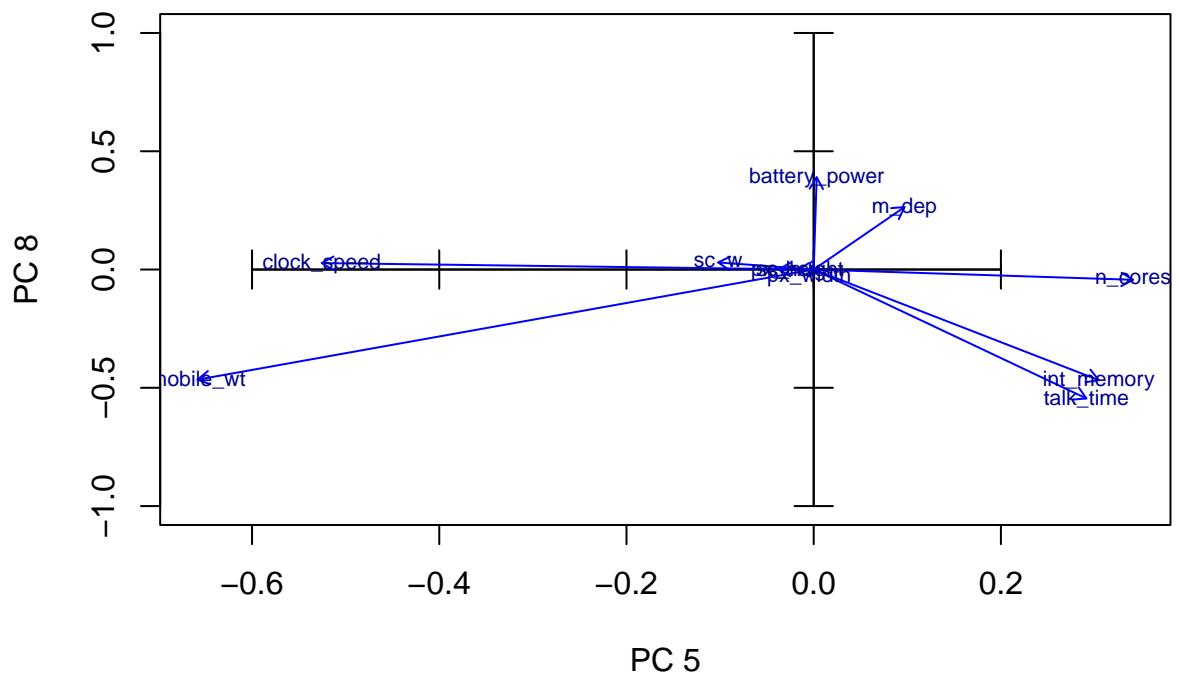
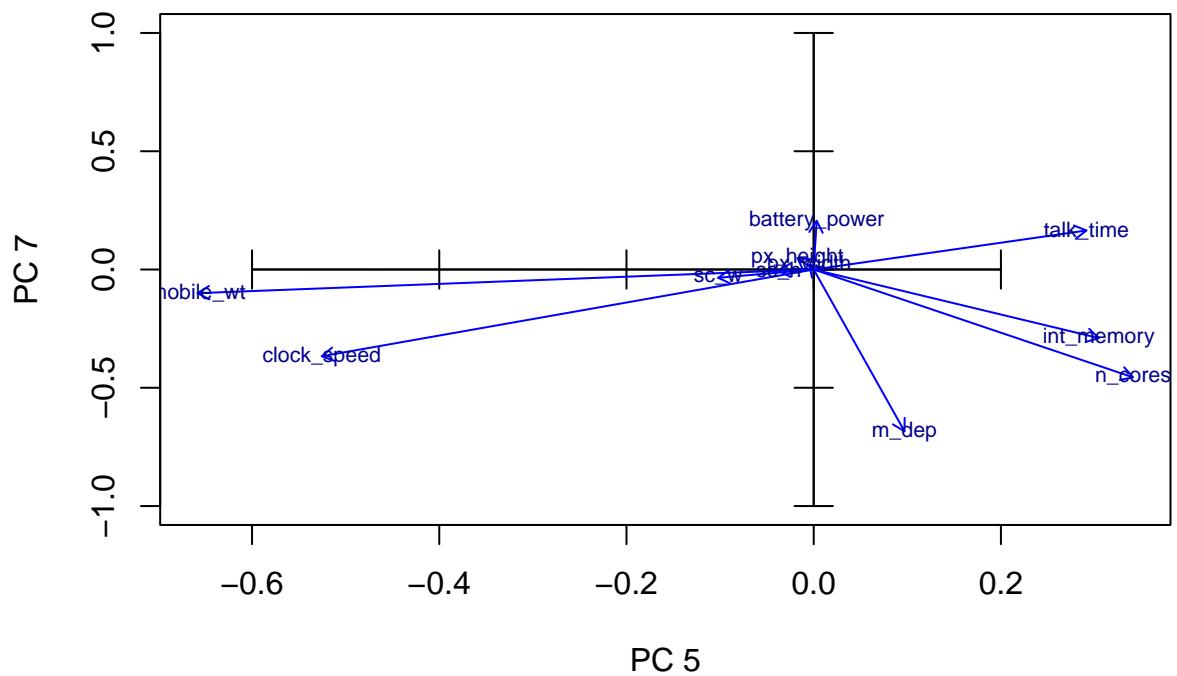


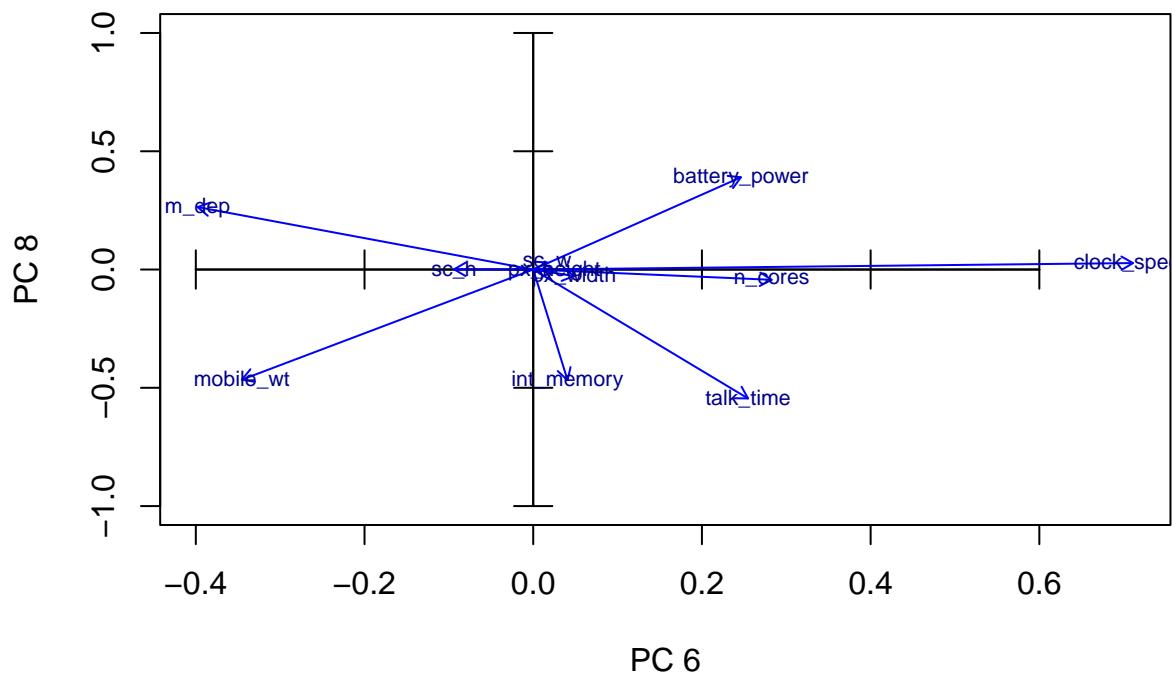
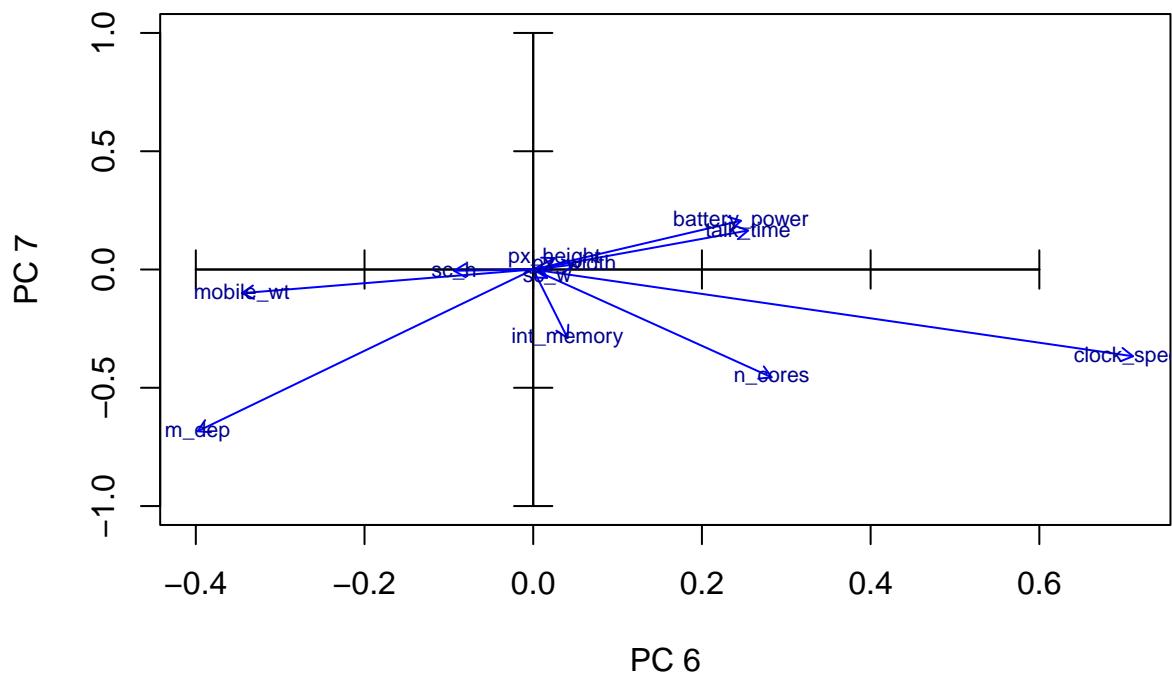


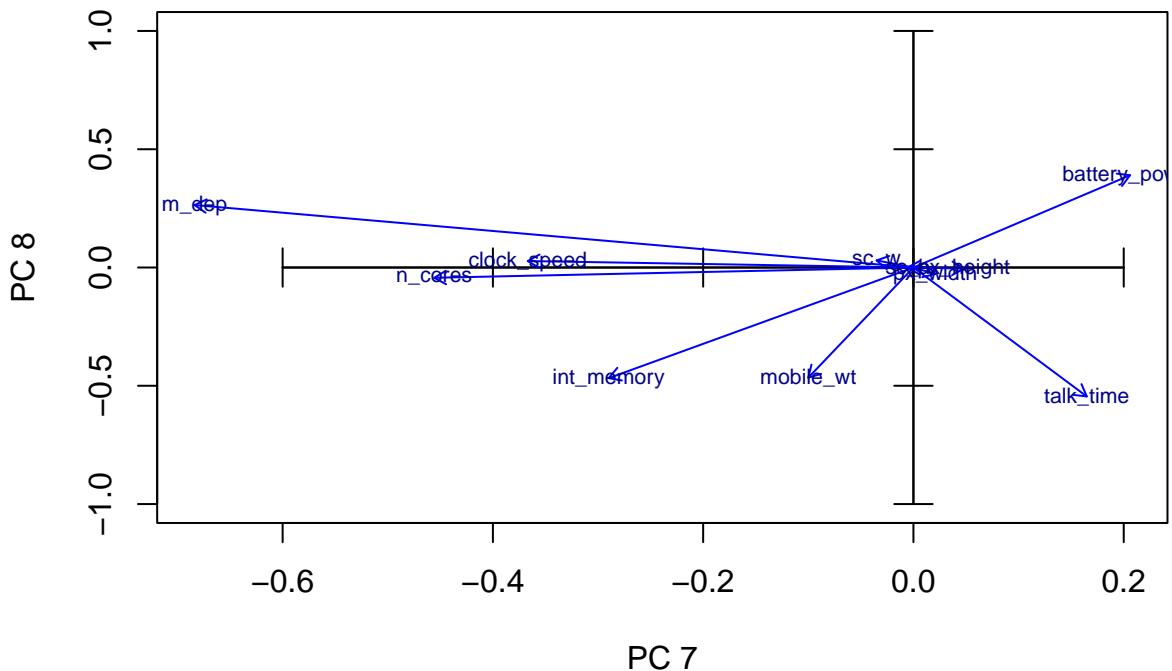












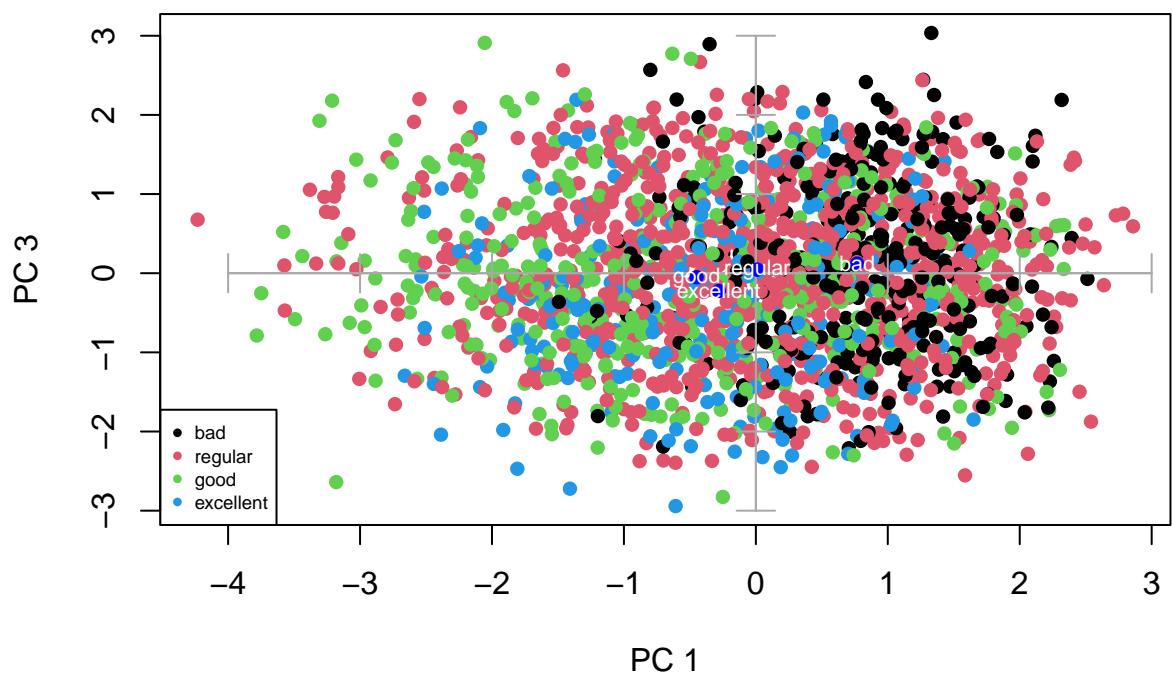
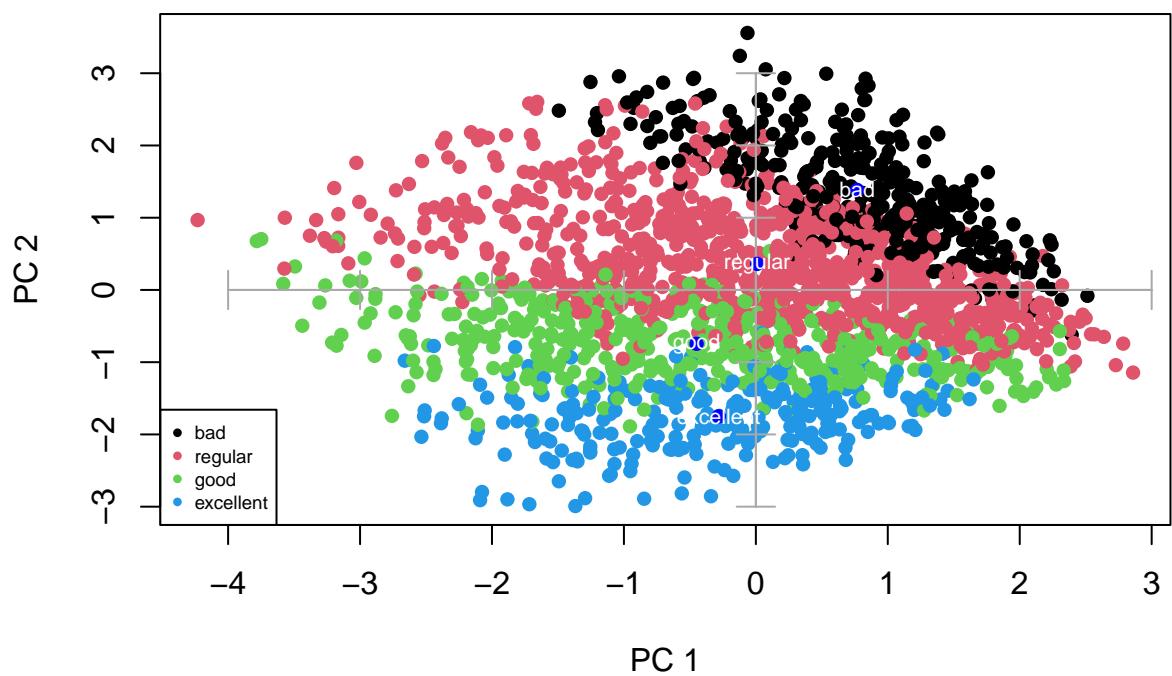
10.3 Categorical variables projection

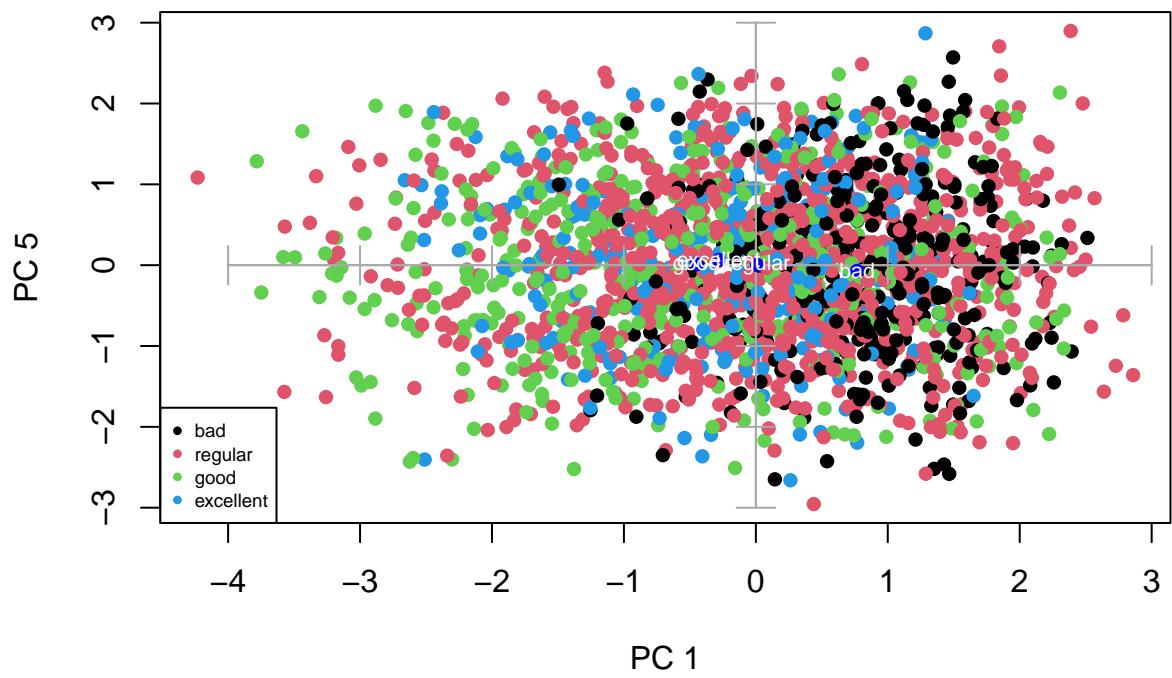
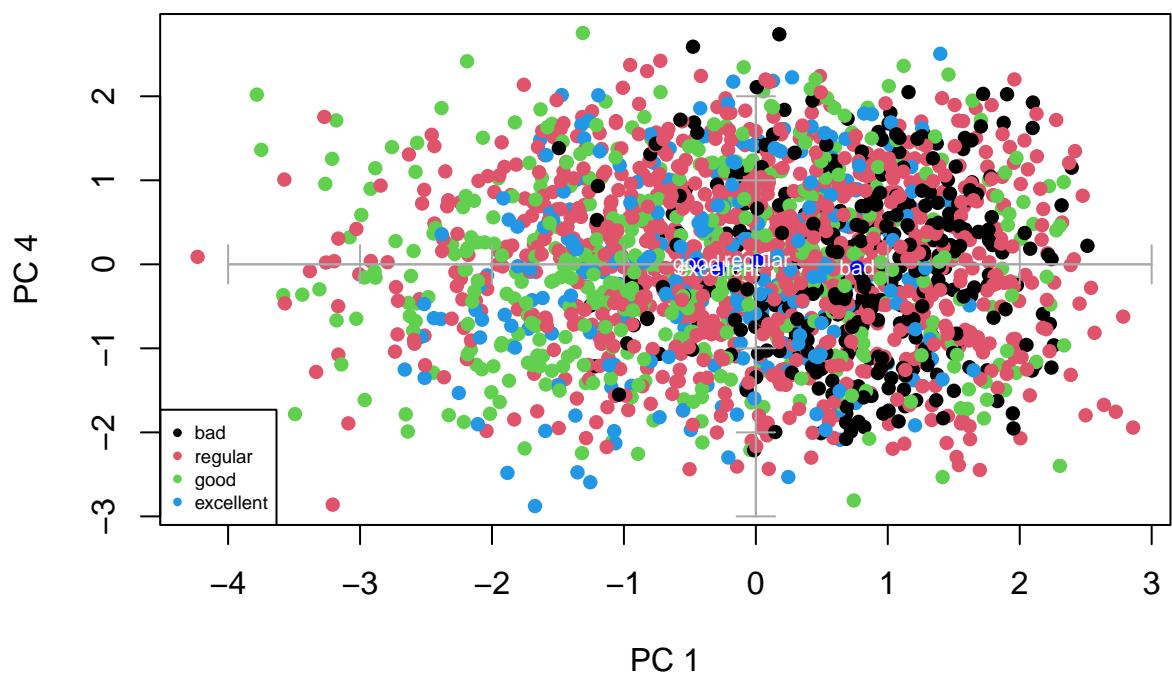
10.4 Porjection of illustrative qualitative variables on individuals' map

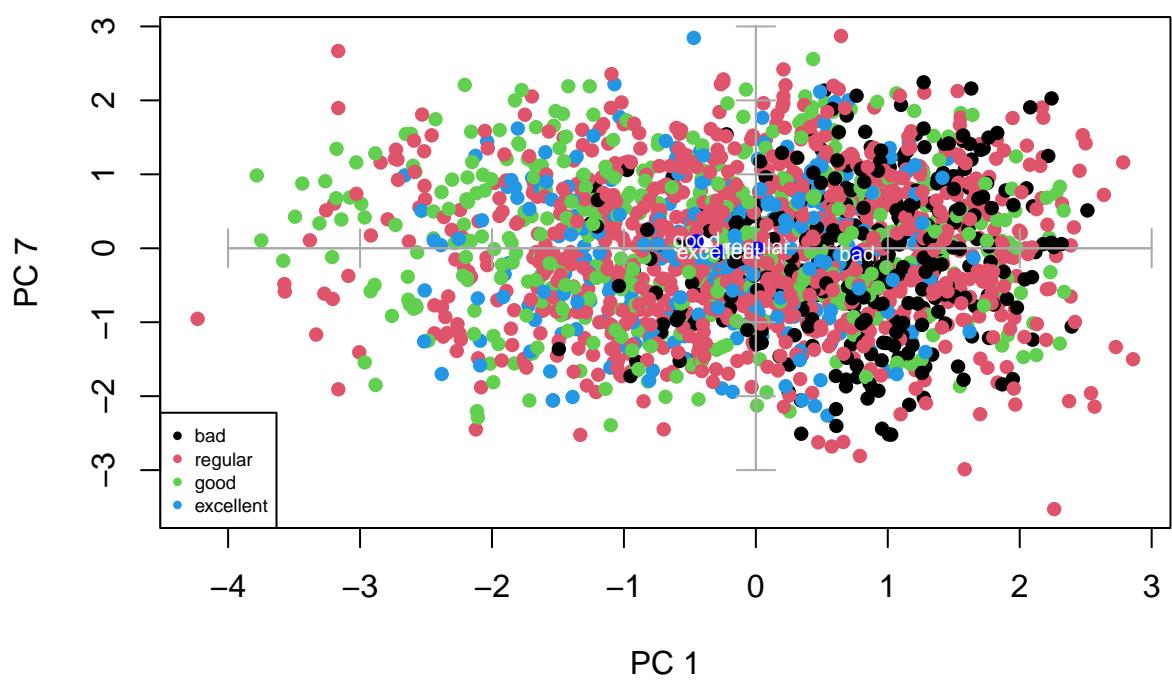
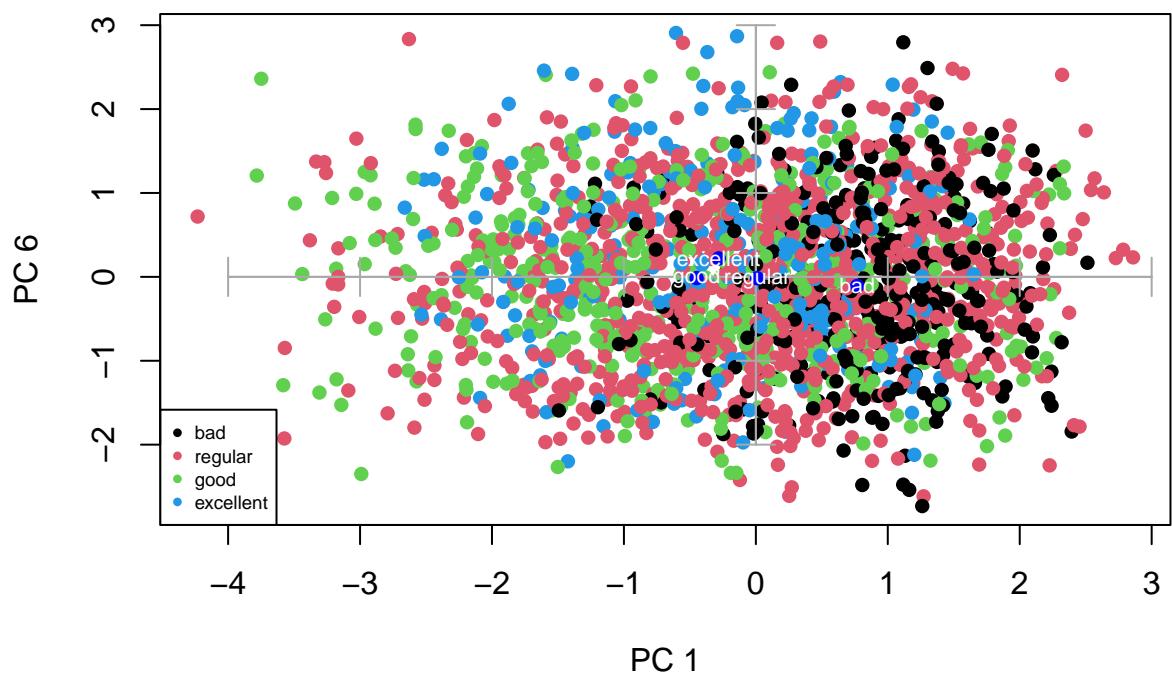
```

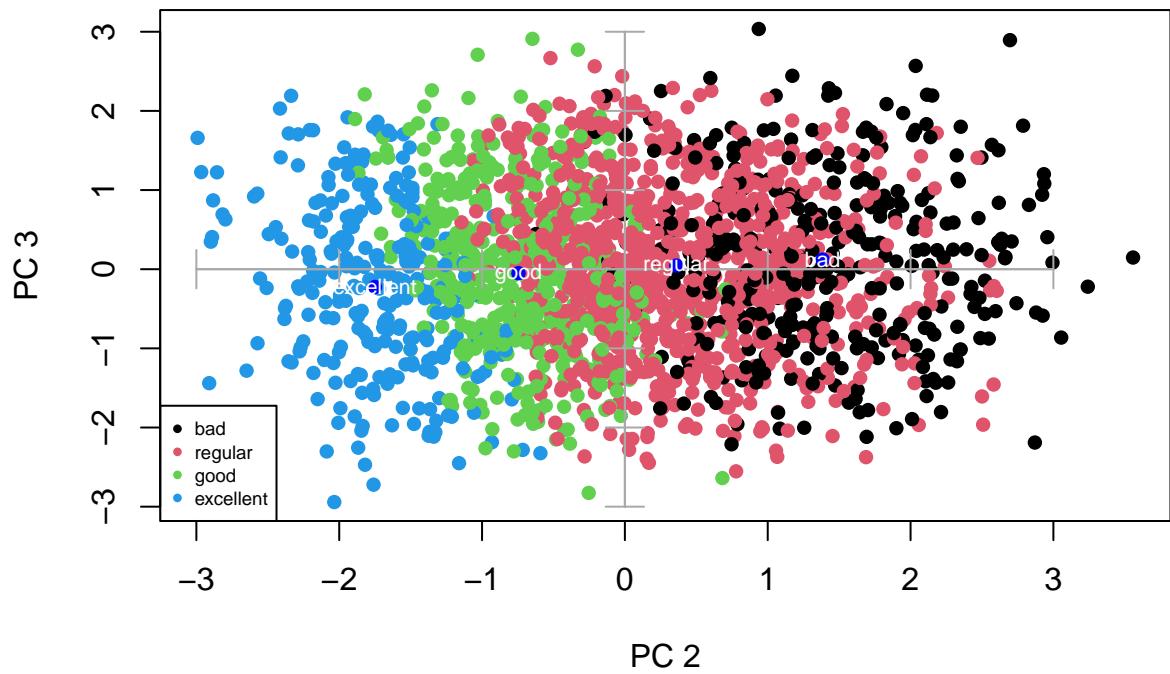
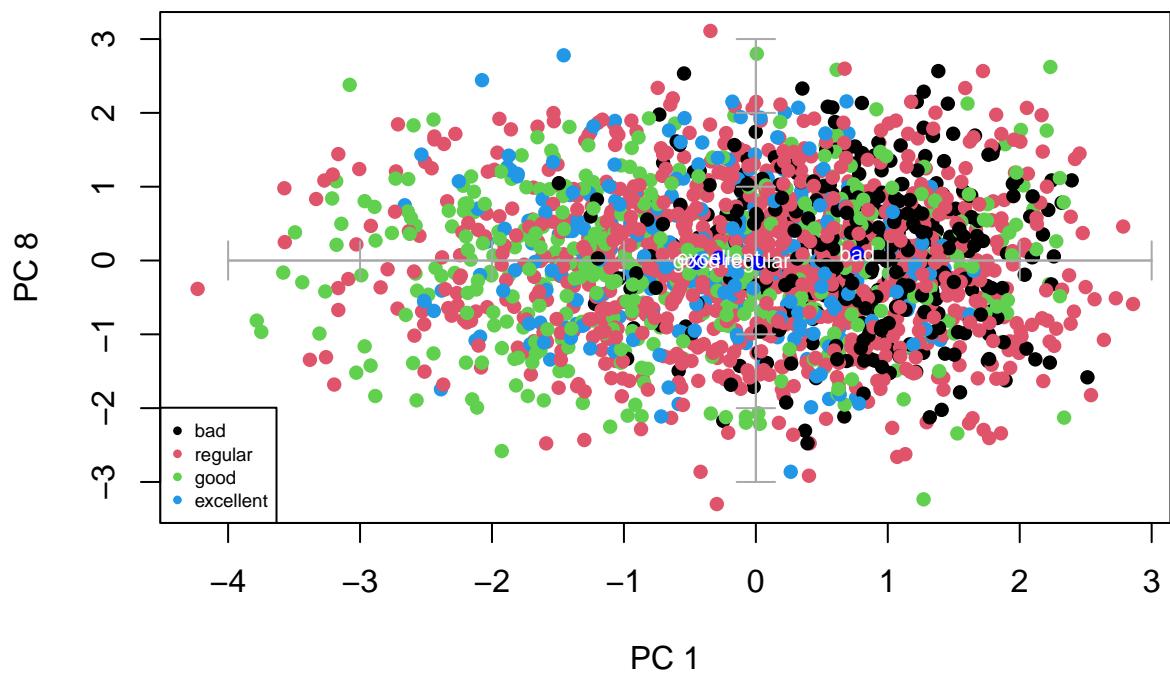
for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$px_r
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)), cex=0.6)
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
    legend("bottomleft",levels(varcat),pch=16,col=c(1:4), cex=0.6)

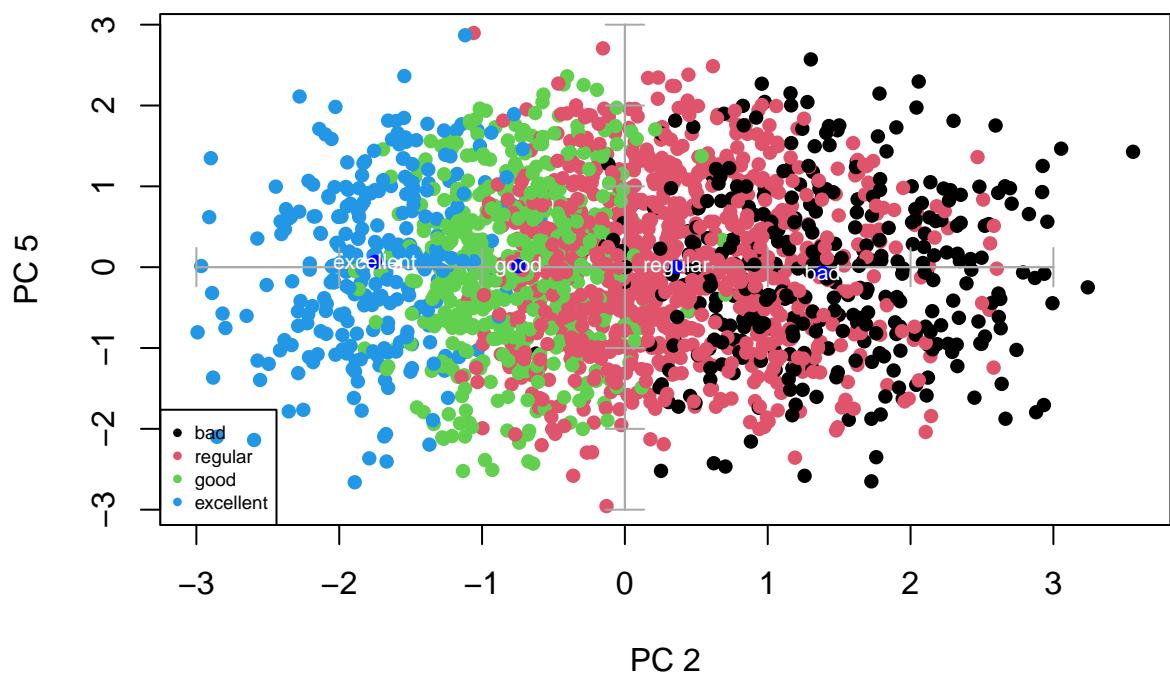
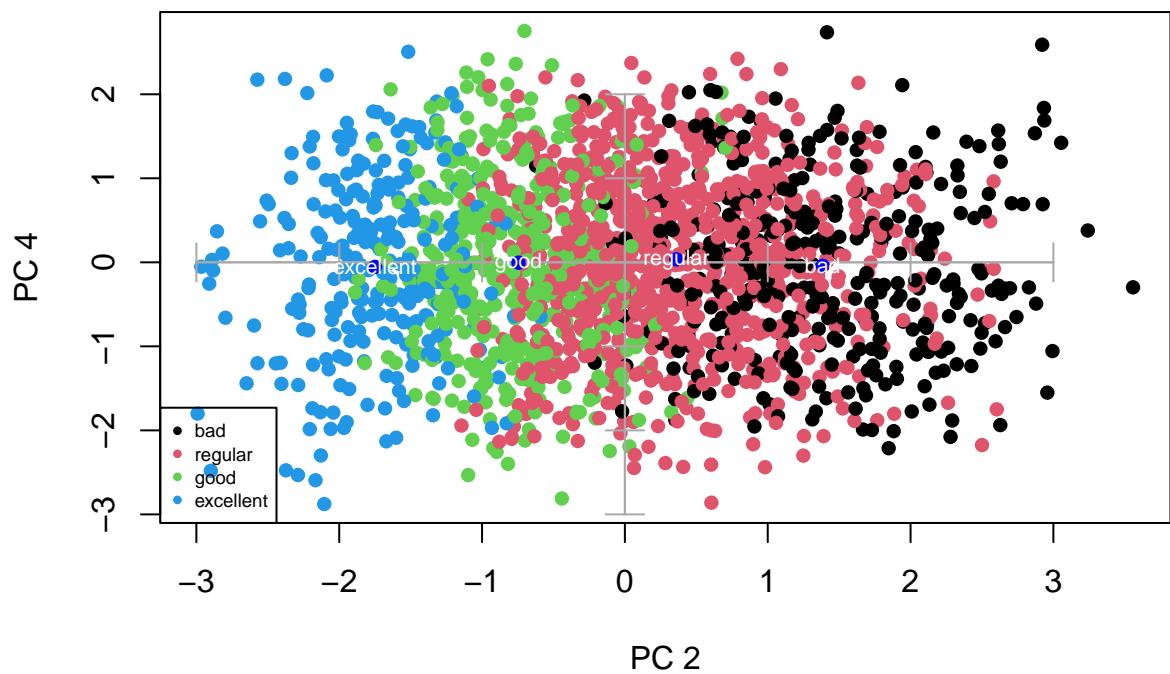
    #select your qualitative variable
    fdic1 = tapply(Psi[,i],varcat,mean)
    fdic2 = tapply(Psi[,j],varcat,mean)
    points(fdic1,fdic2,pch=16,col="blue")
    text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
  }
}
  
```

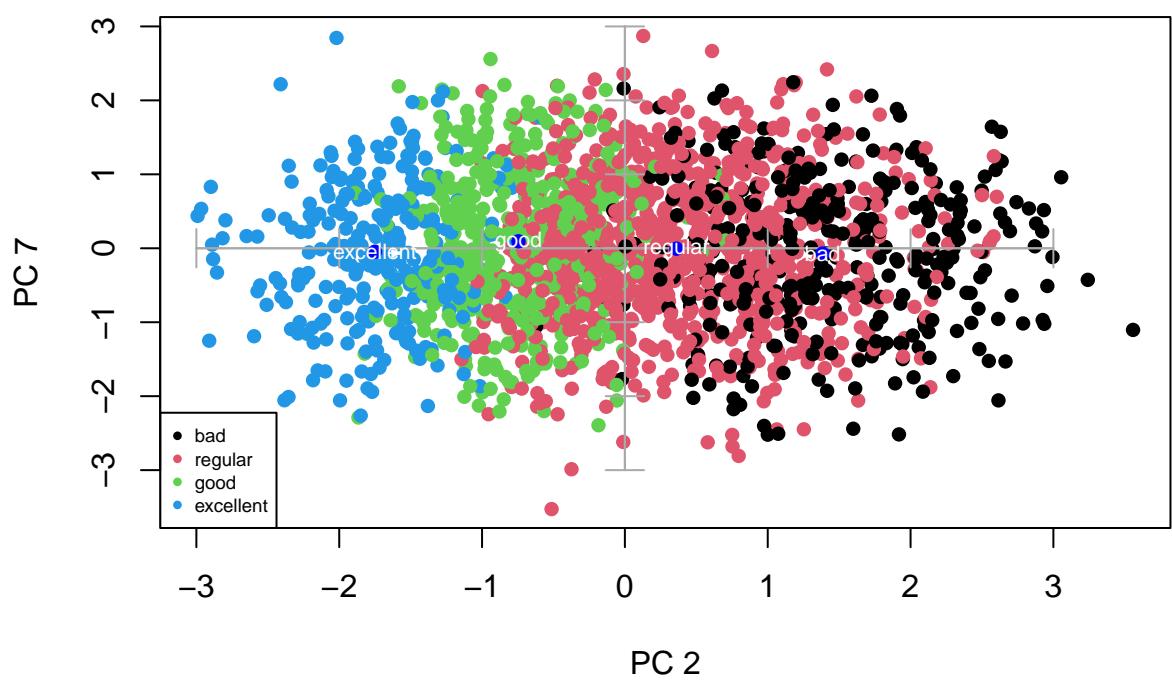
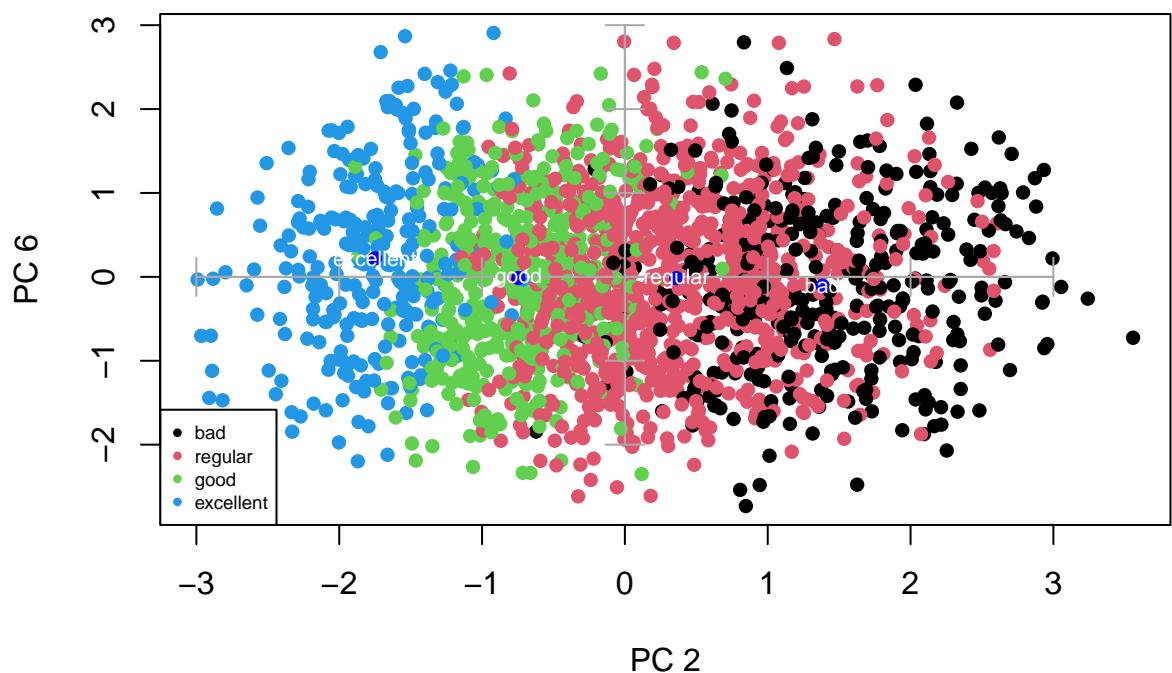


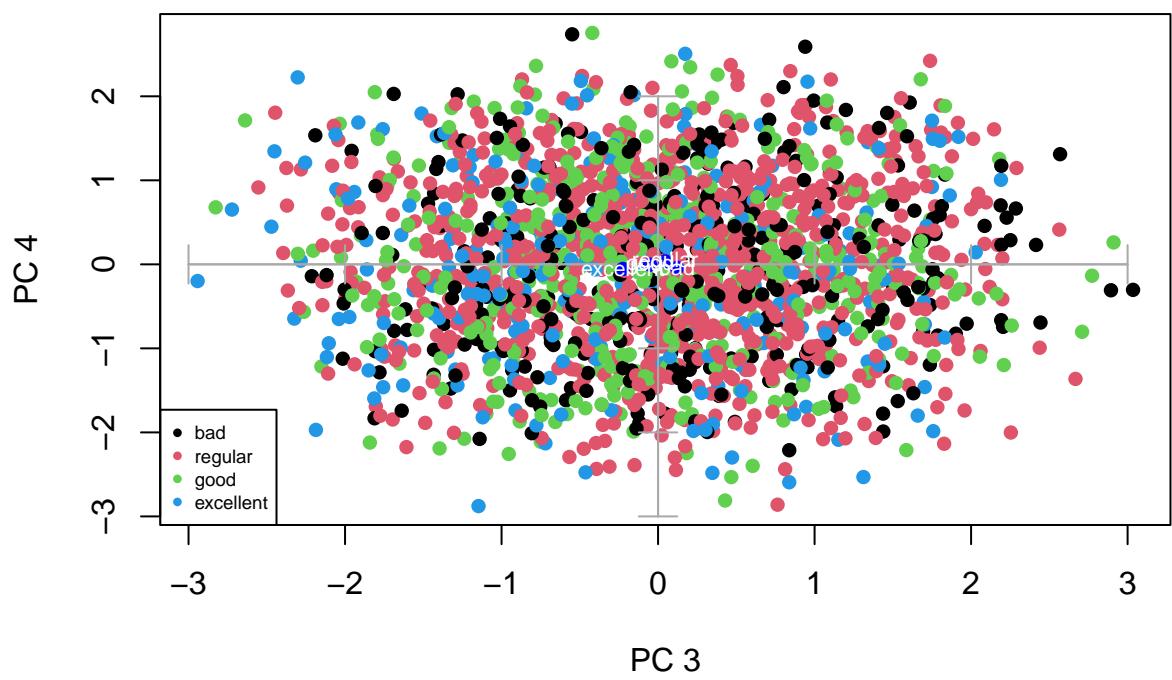
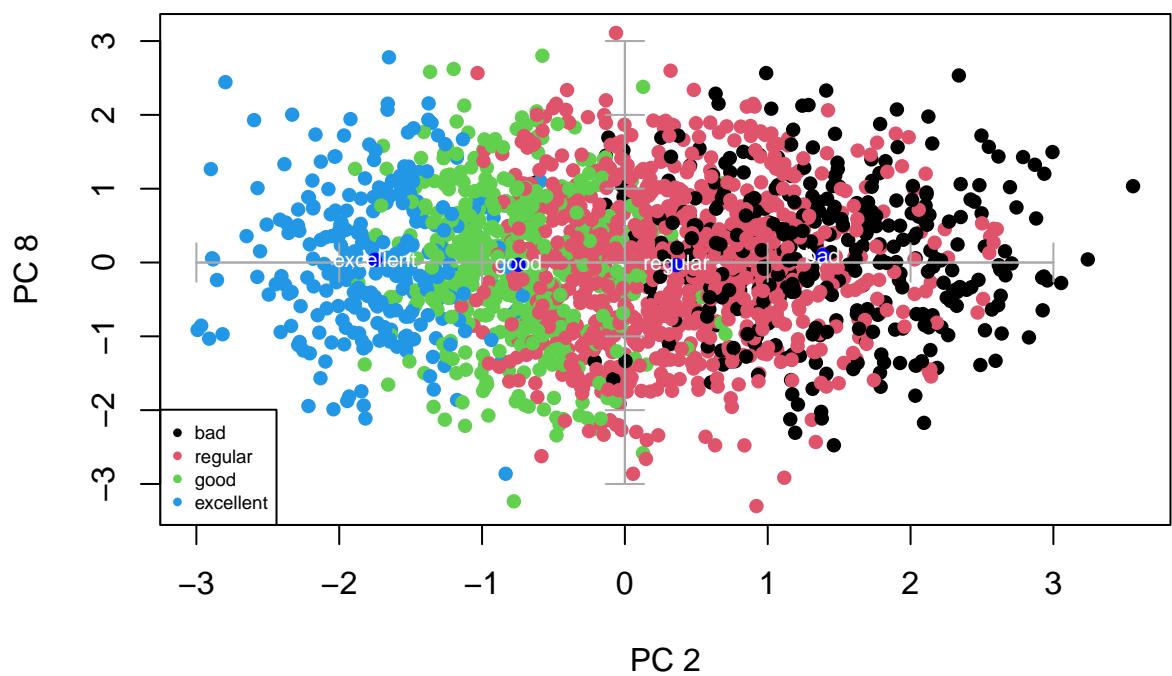


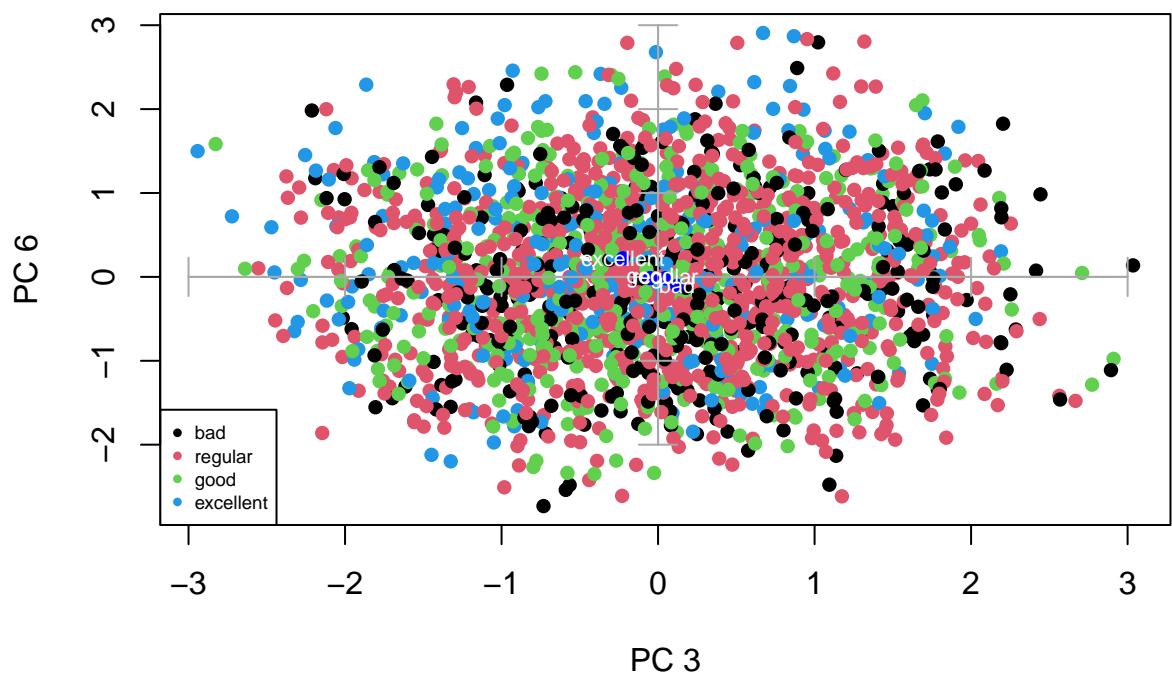
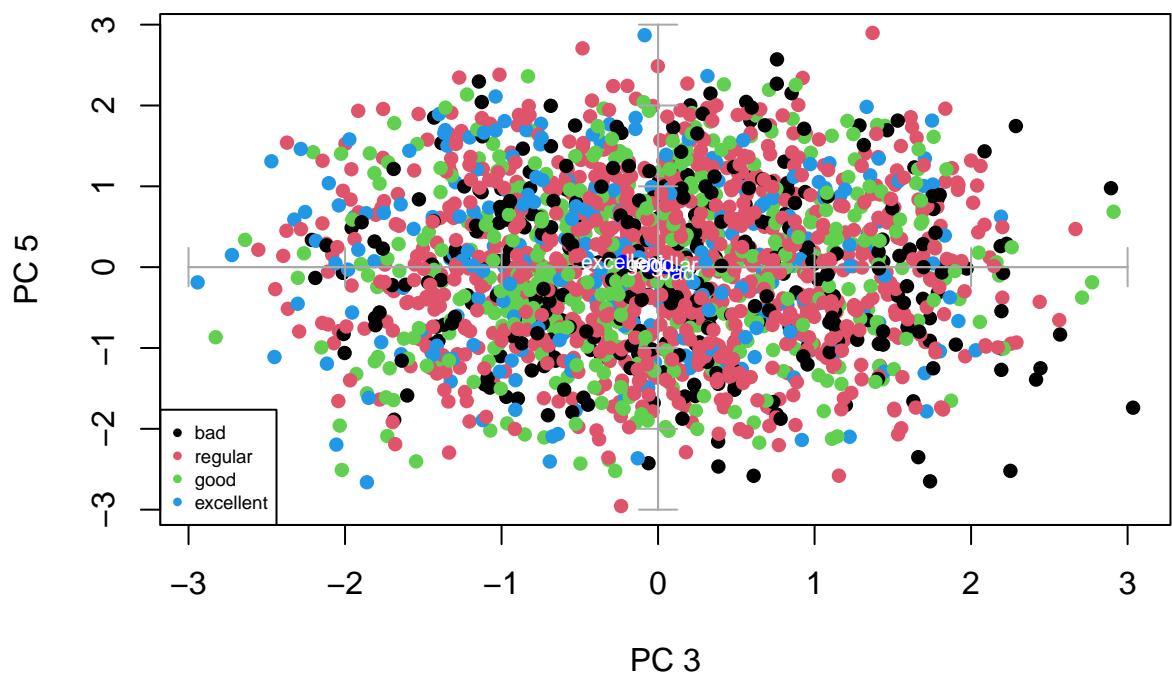


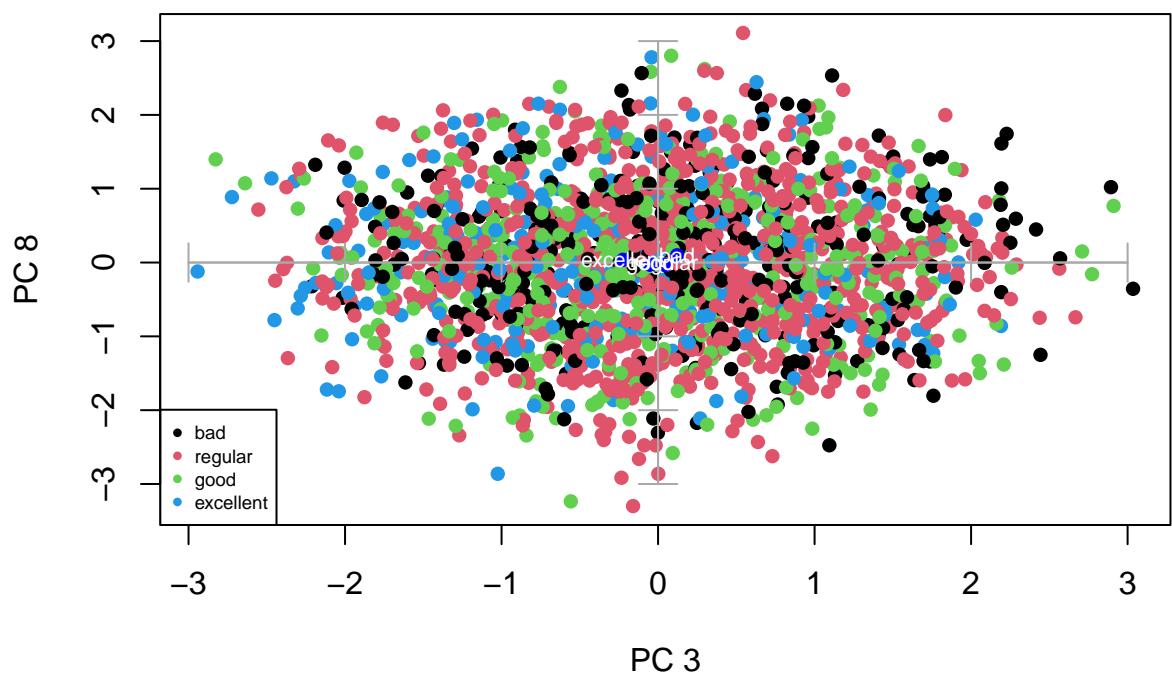
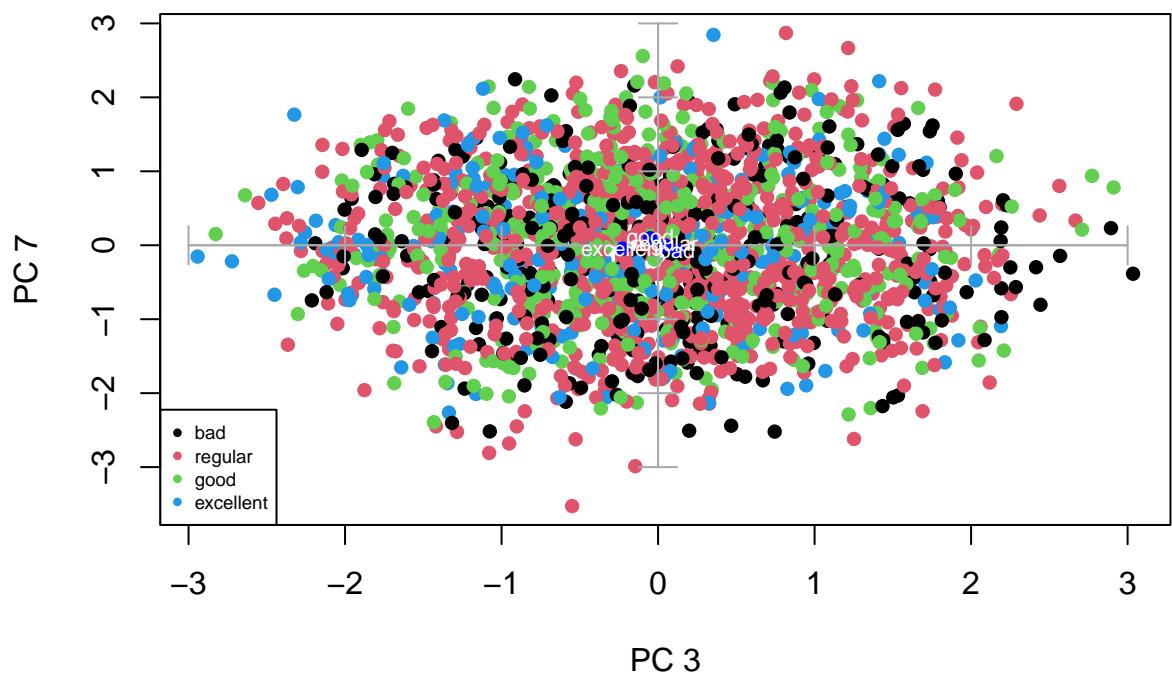


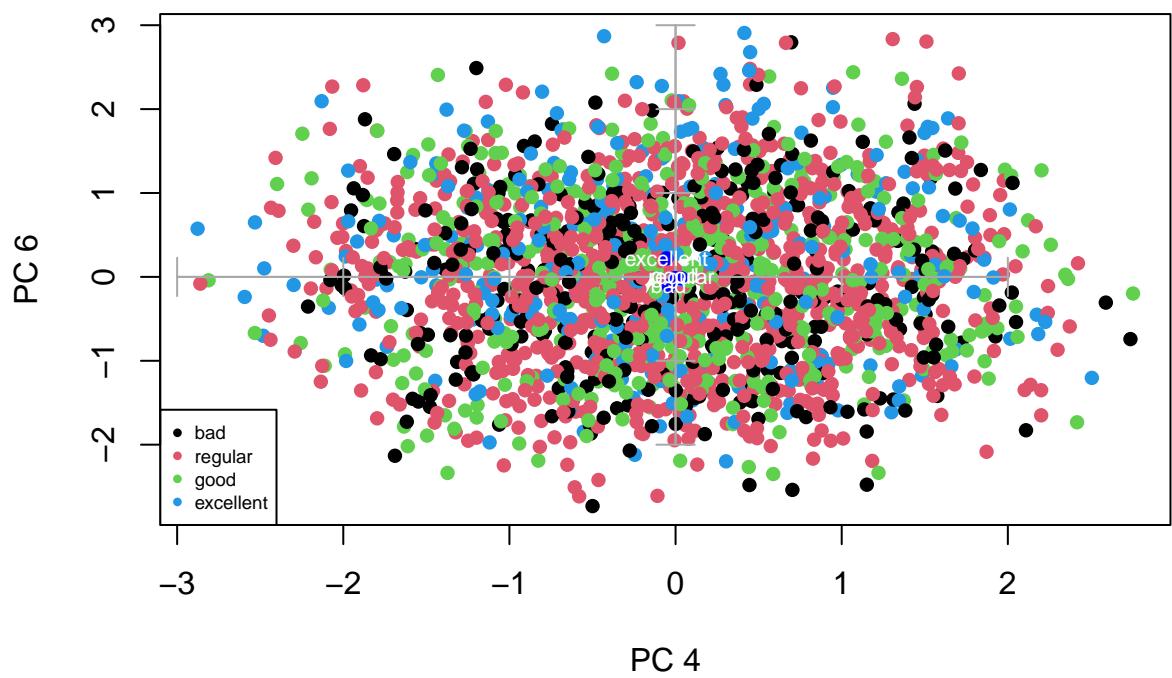
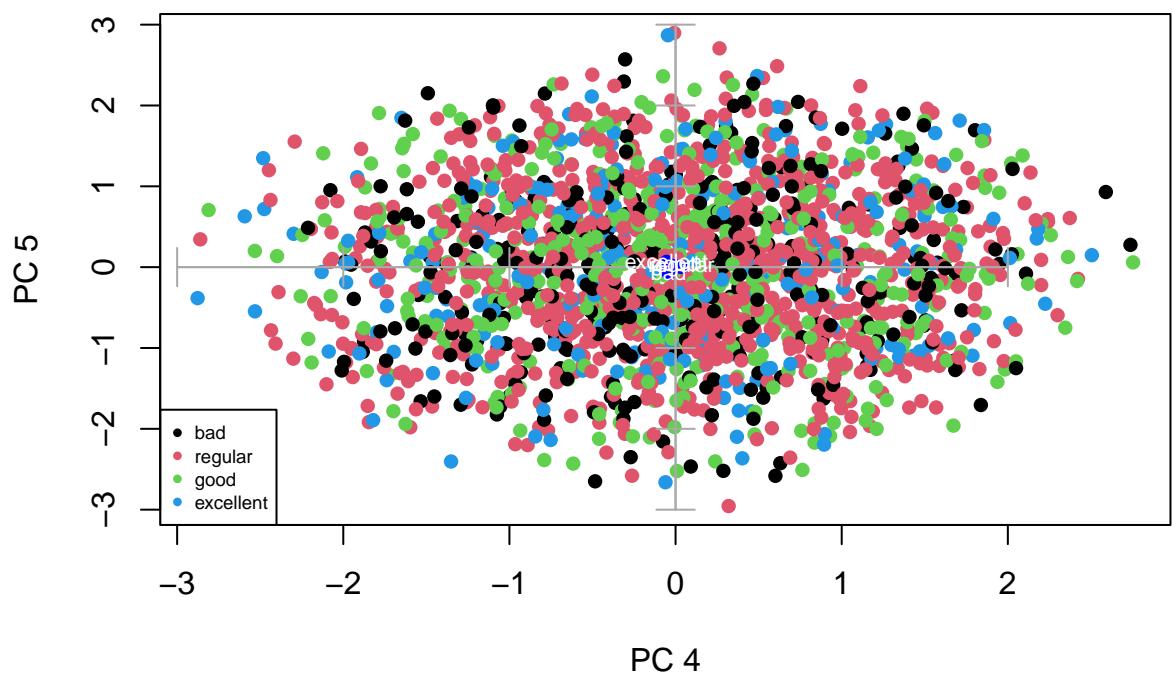


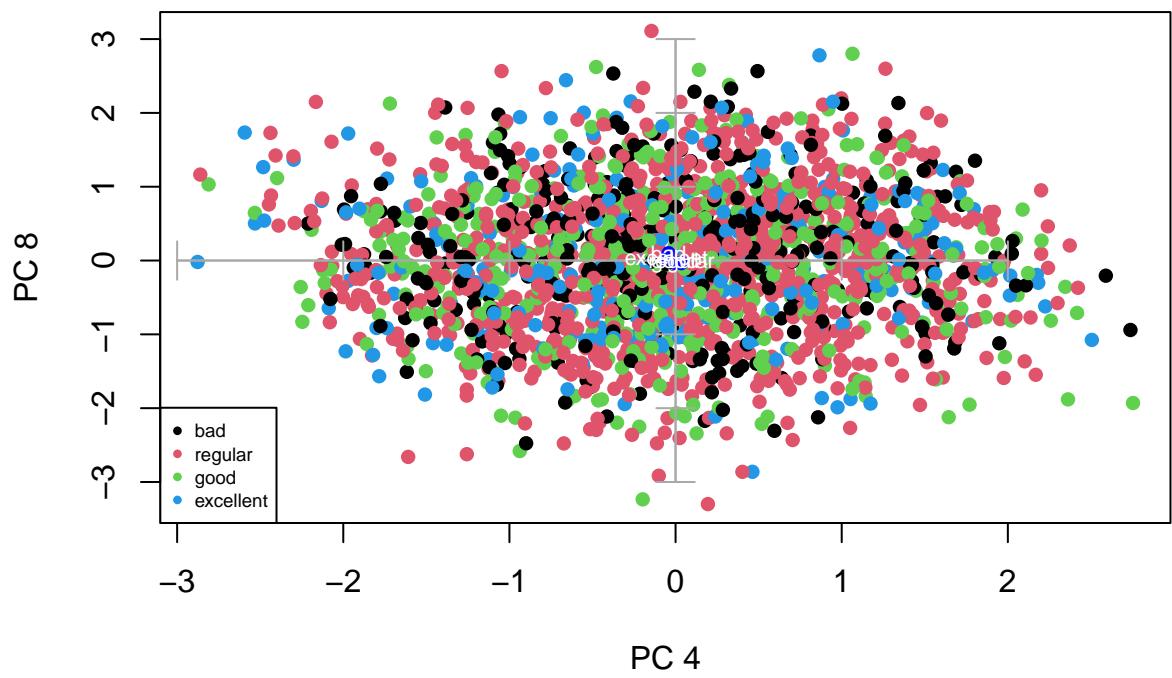
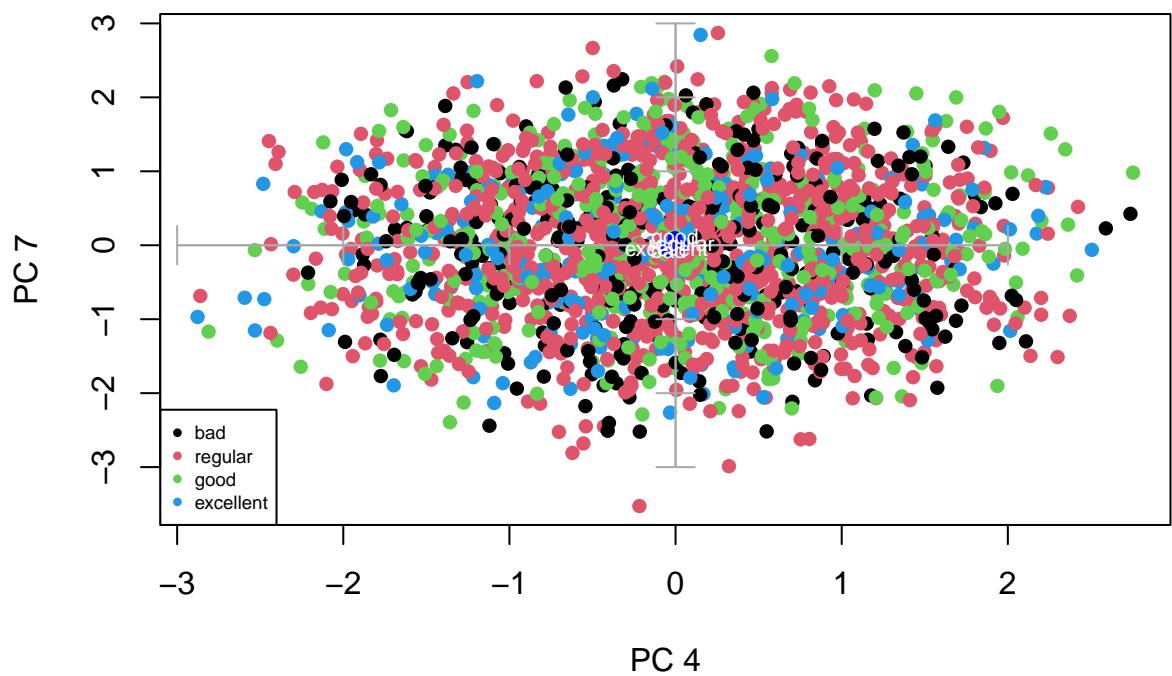


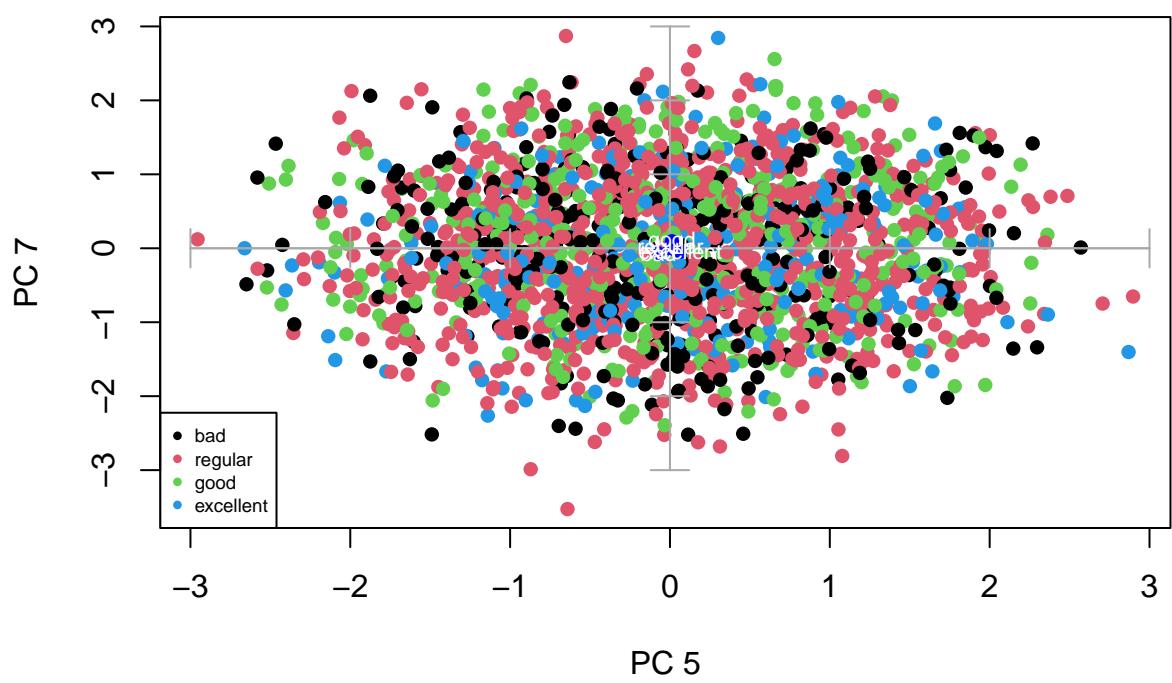
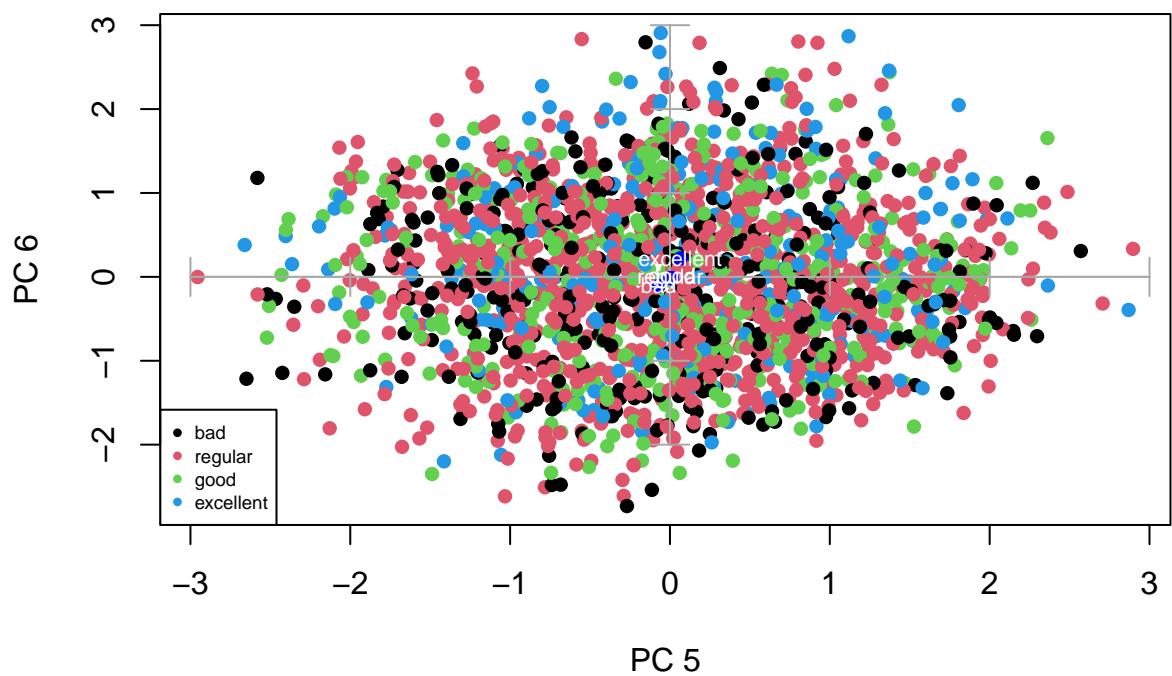


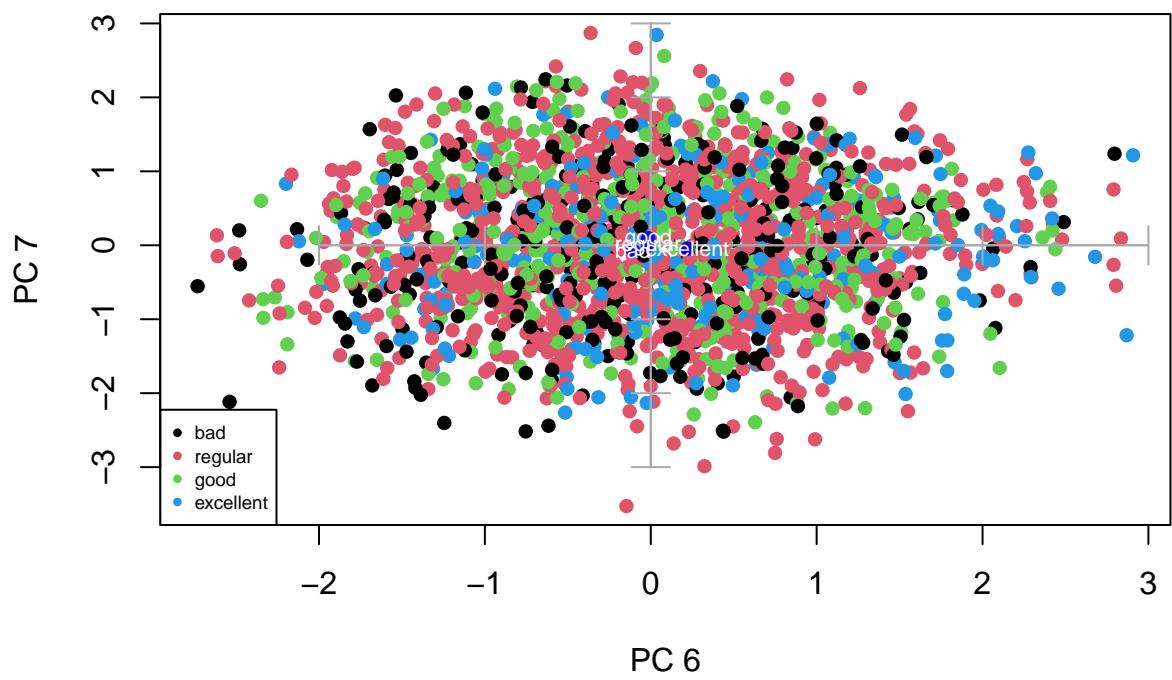
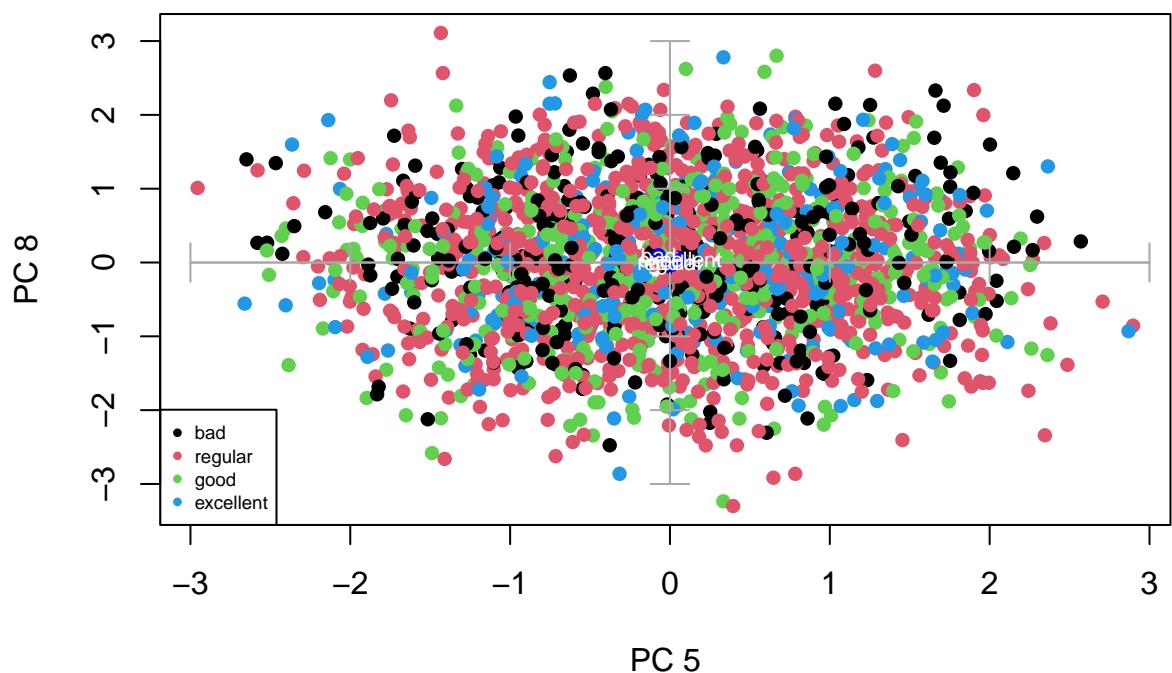


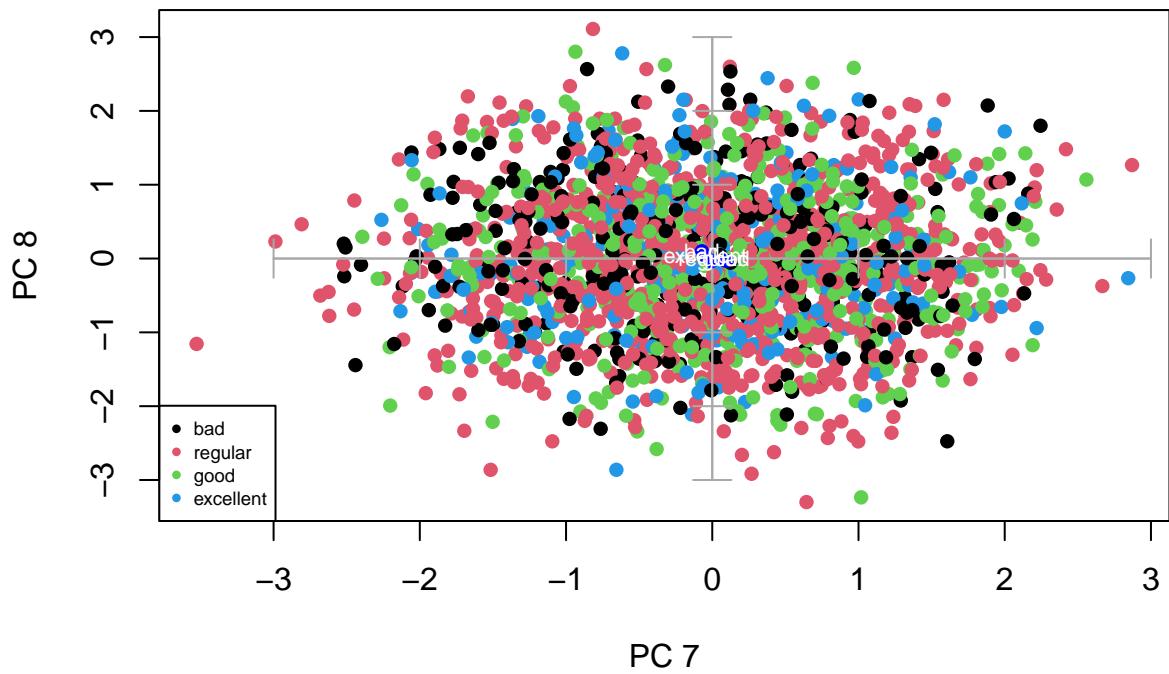
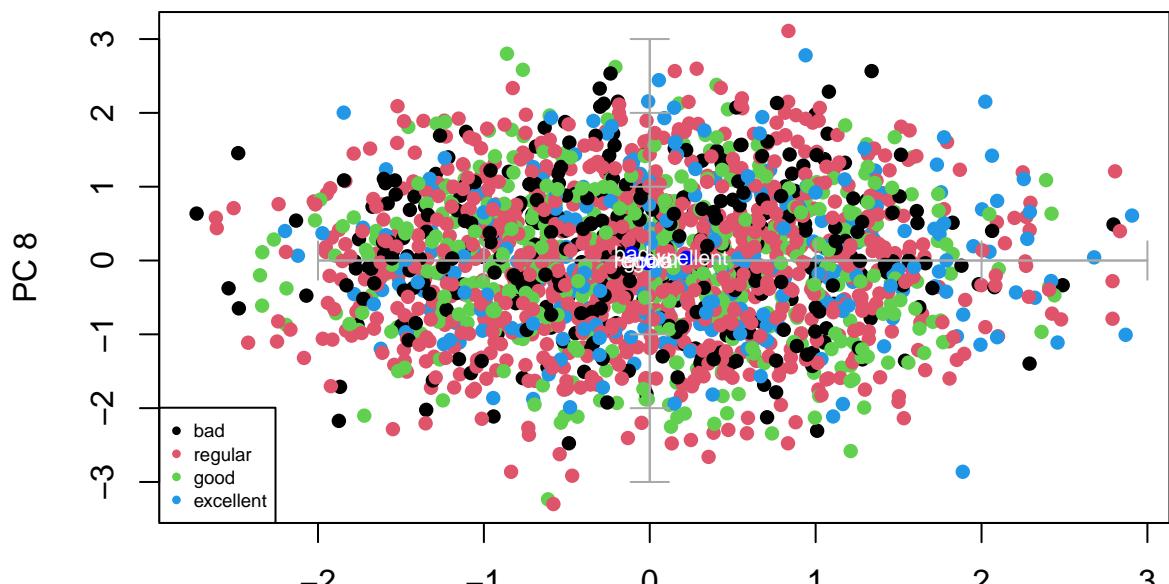












```

for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$sc_d
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
  }
}
  
```

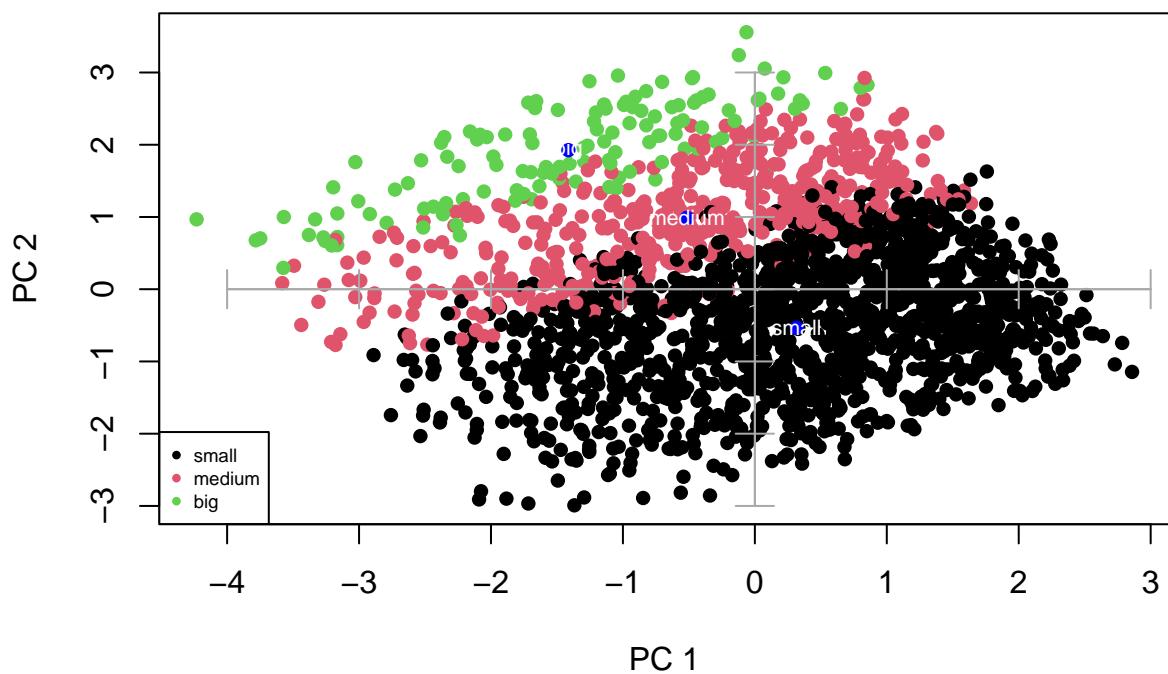
```

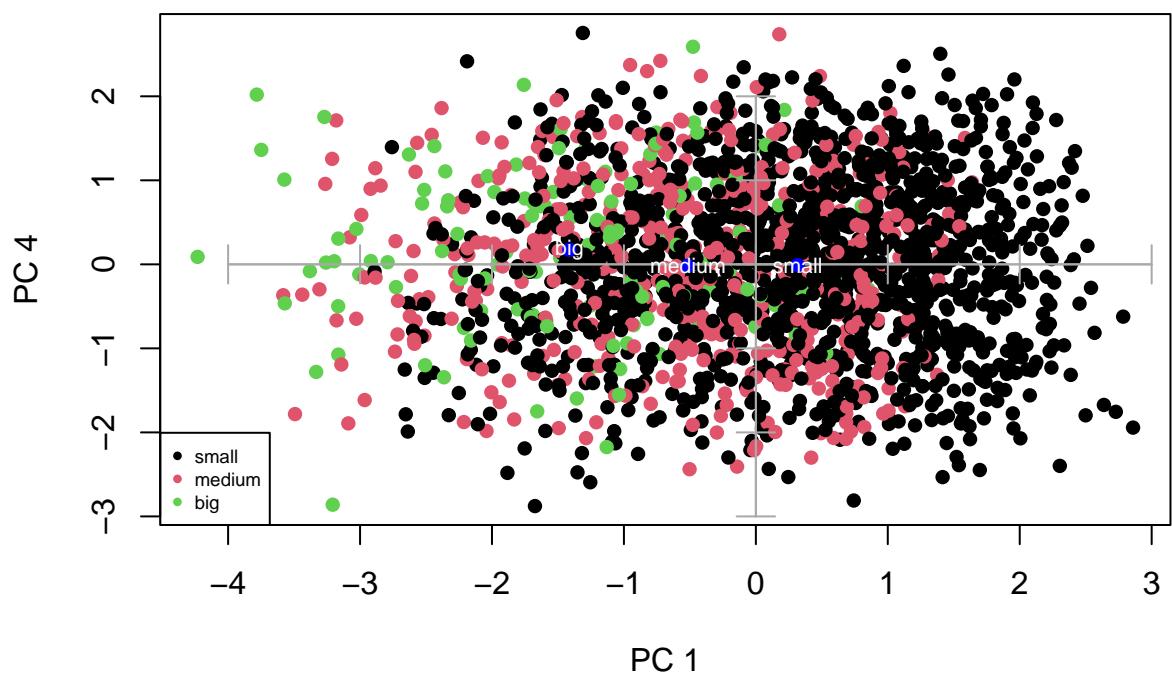
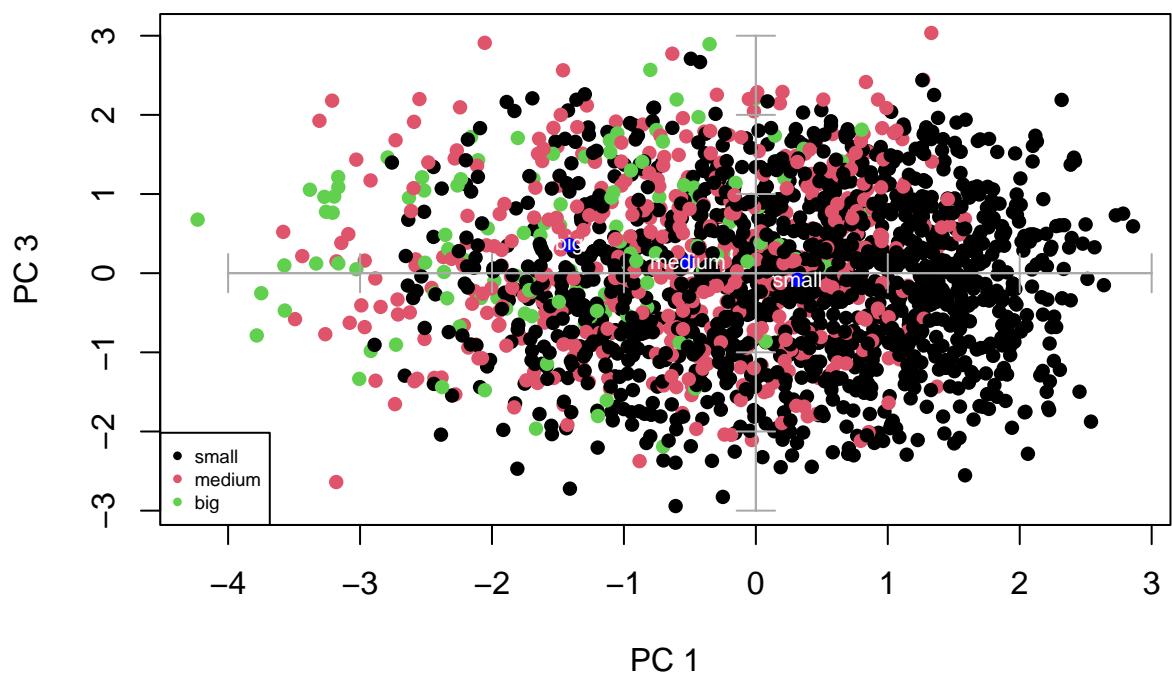
legend("bottomleft",levels(varcat),pch=16,col=c(1:3), cex=0.6)

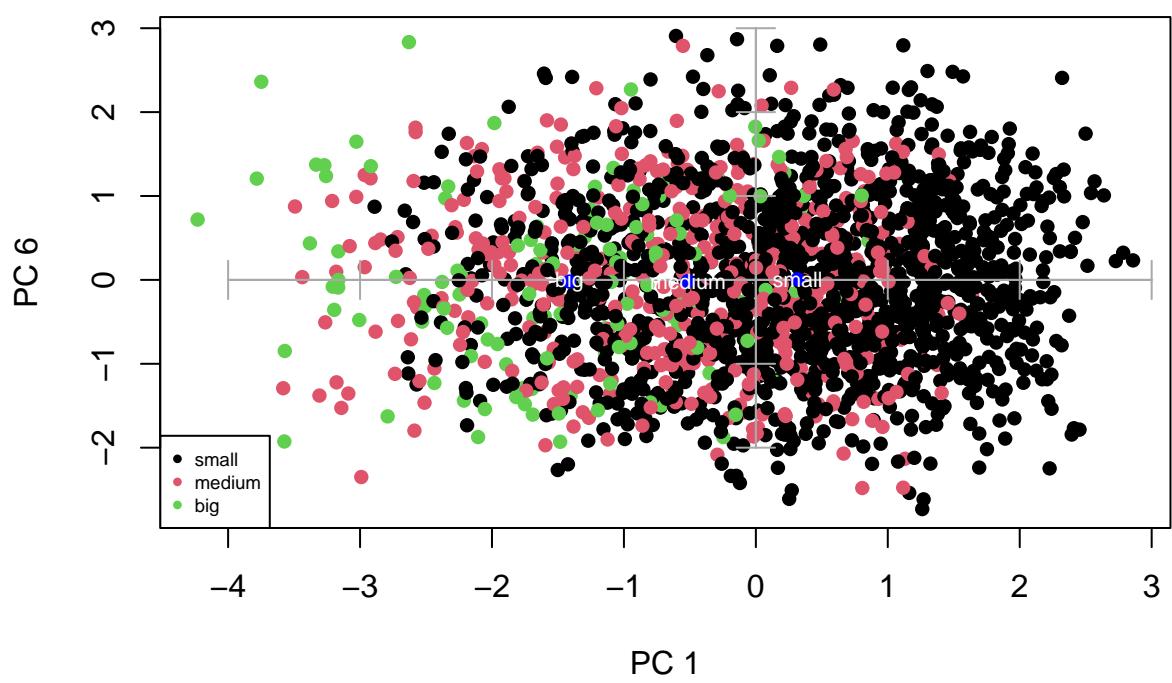
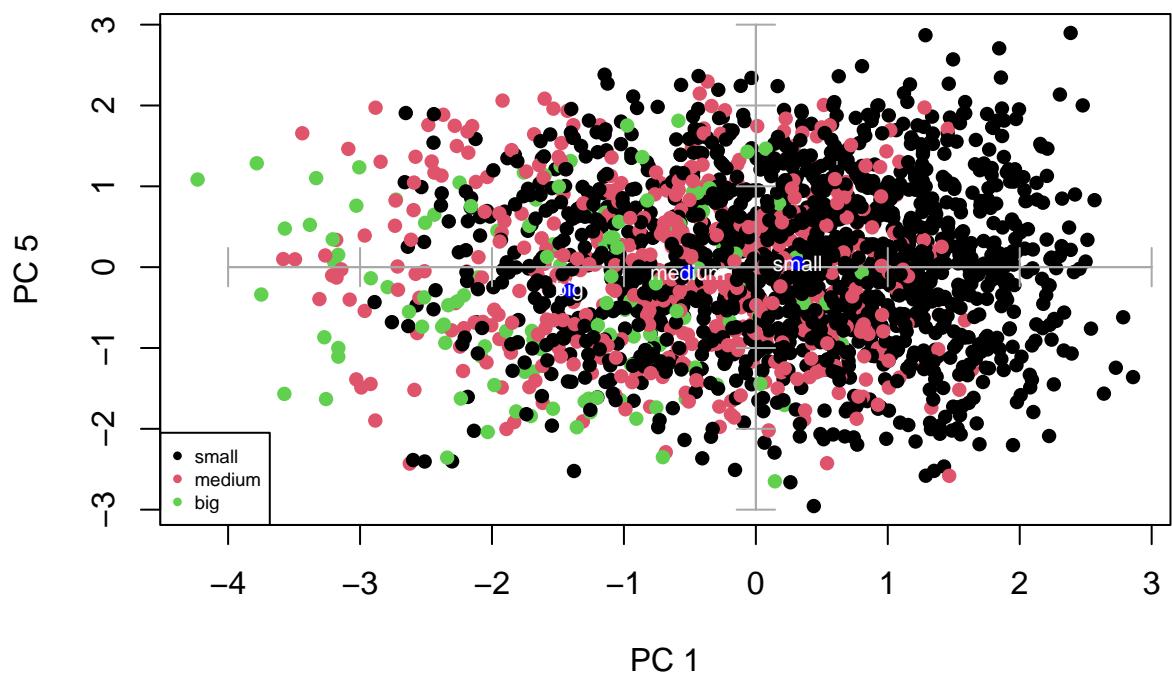
#select your qualitative variable
fdic1 = tapply(Psi[,i],varcat,mean)
fdic2 = tapply(Psi[,j],varcat,mean)
points(fdic1,fdic2,pch=16,col="blue")
text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
}

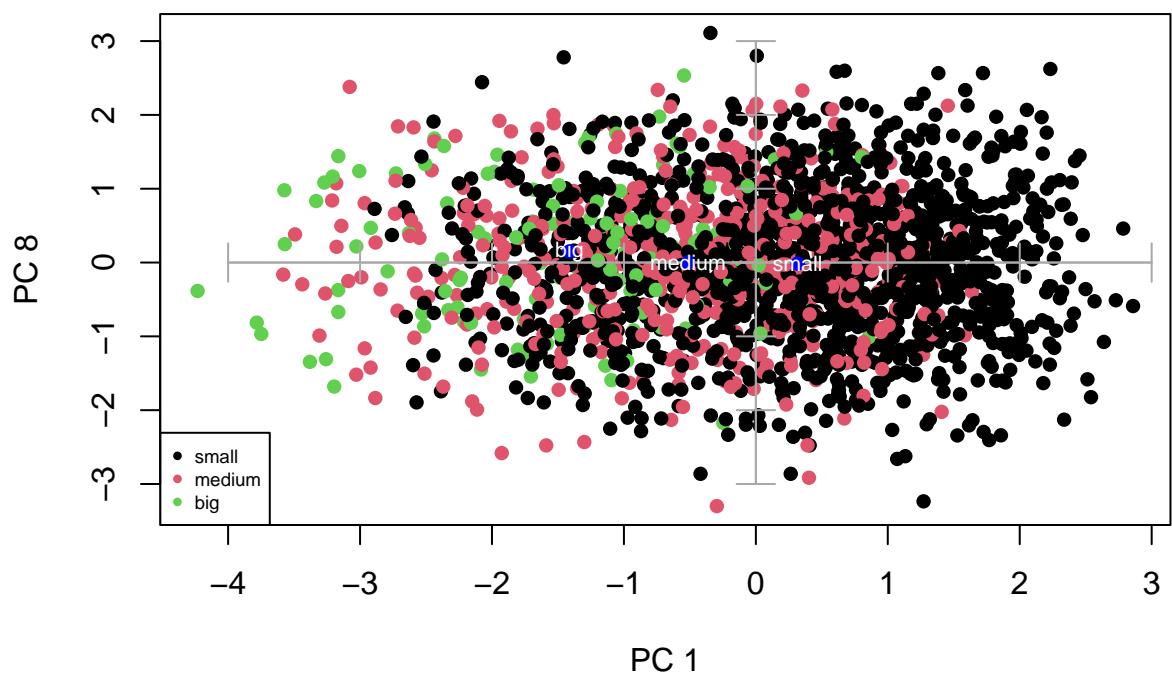
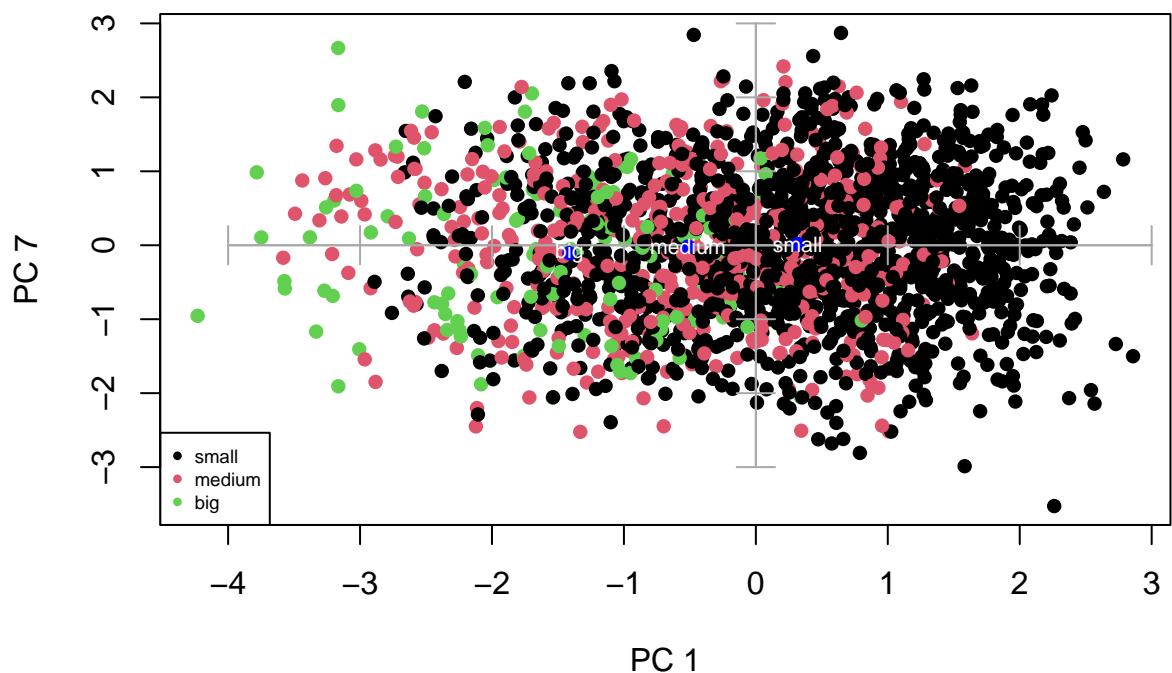
}

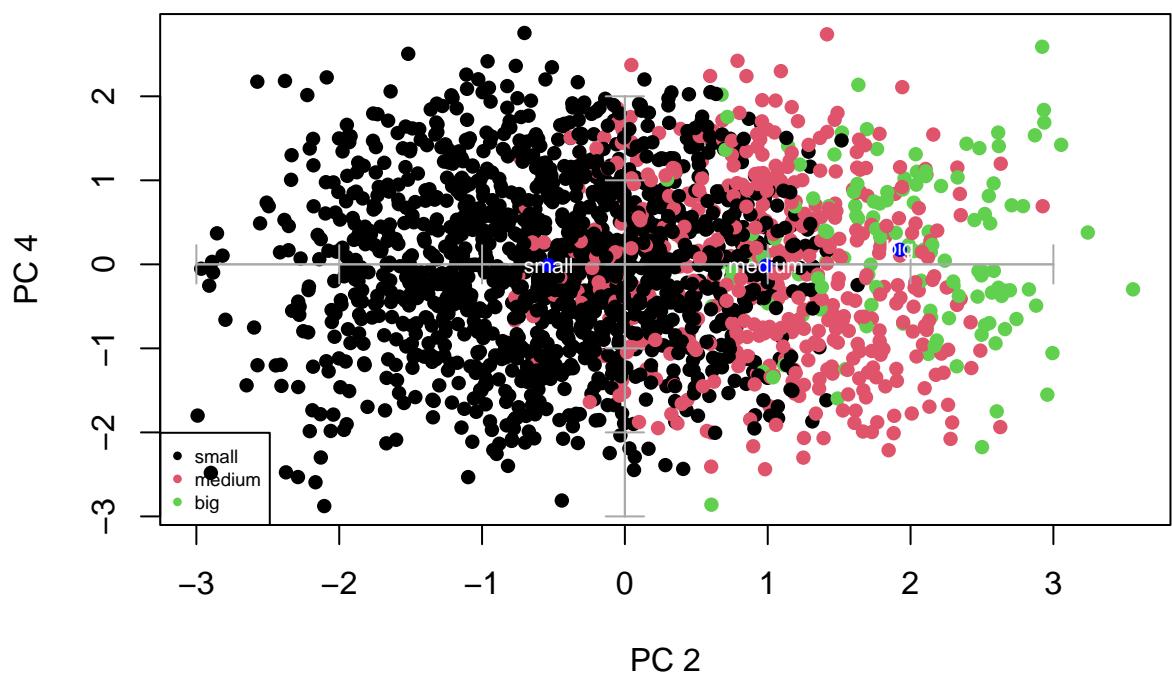
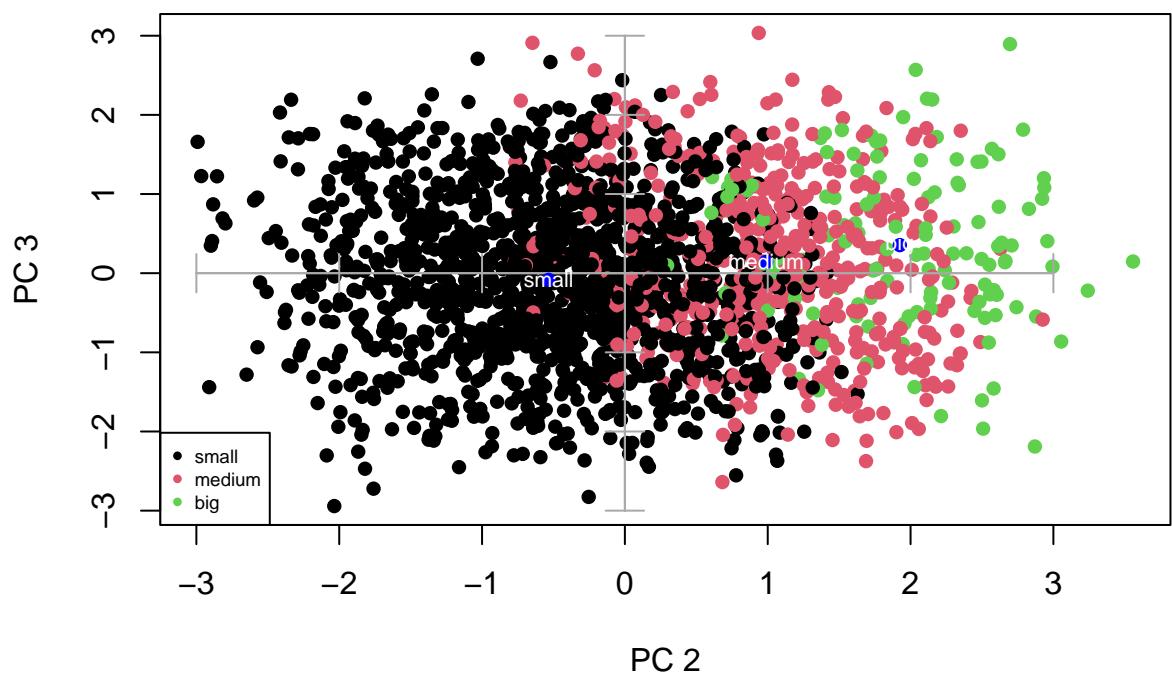
```

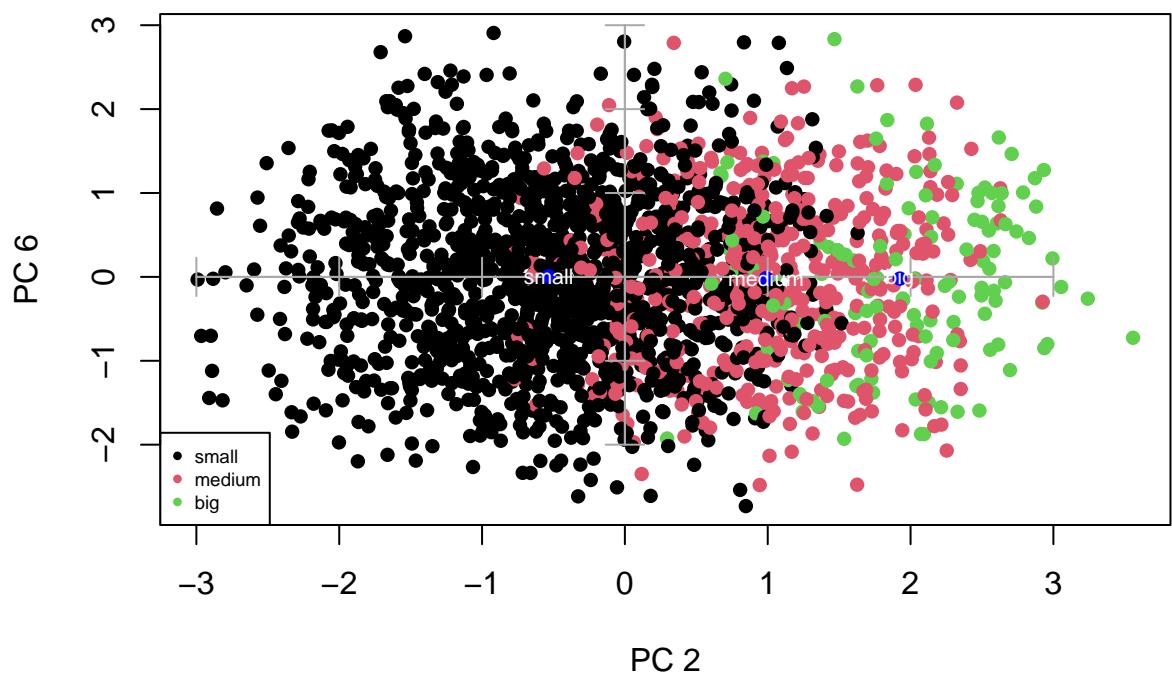
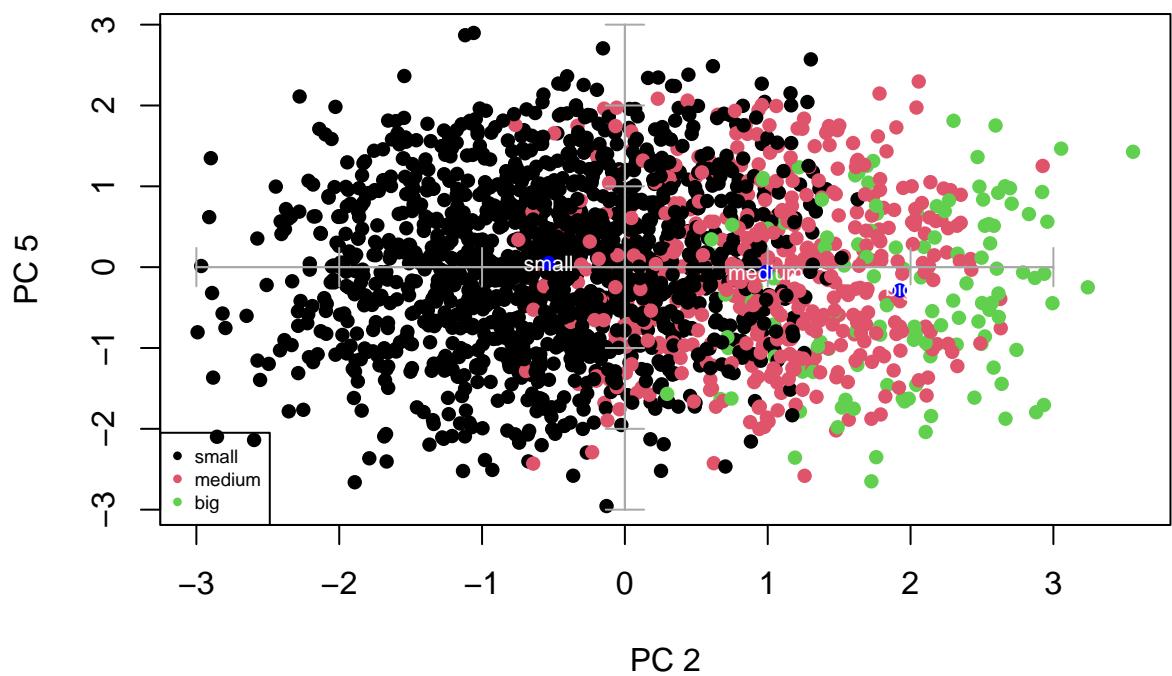


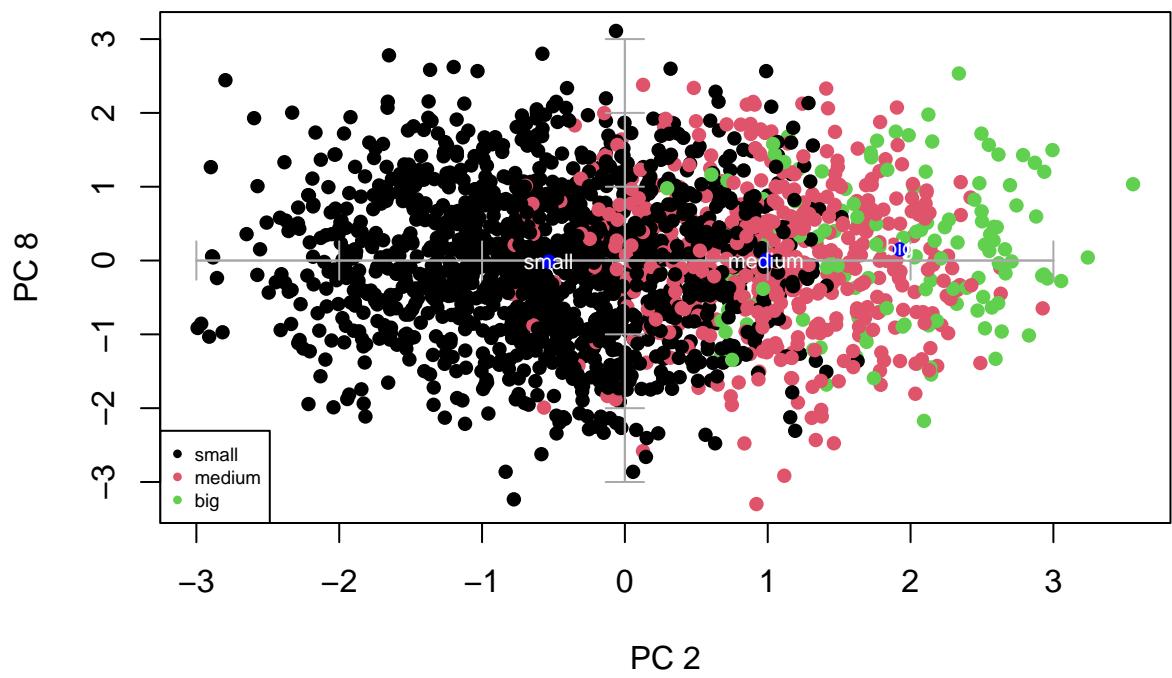
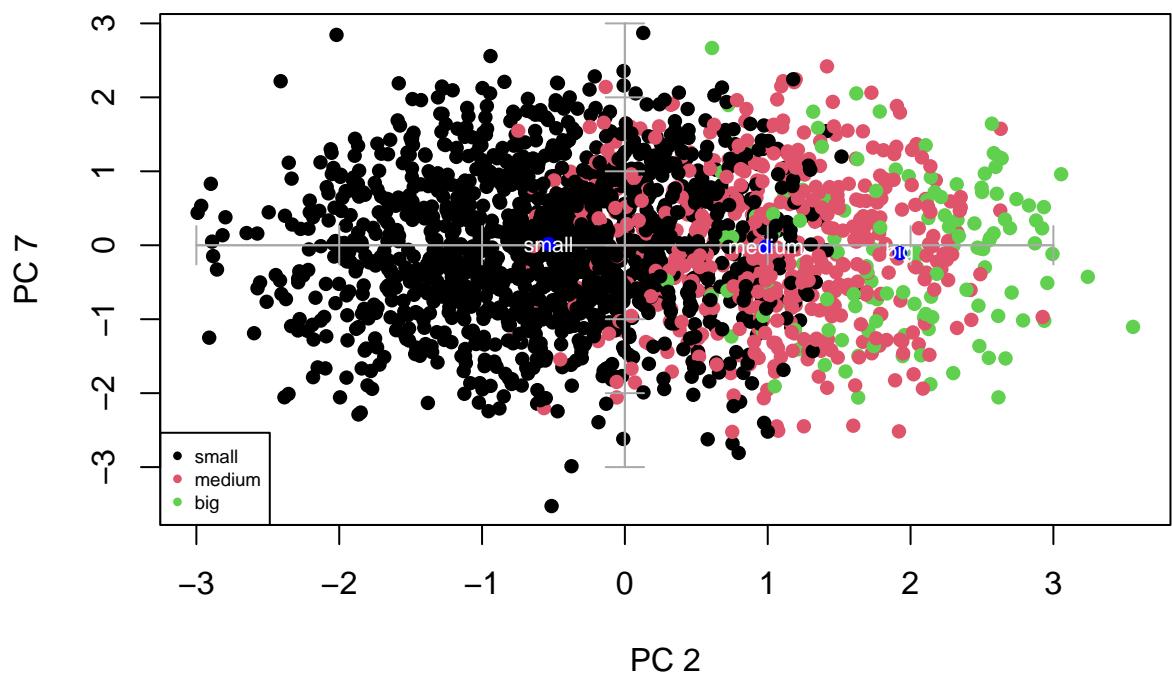


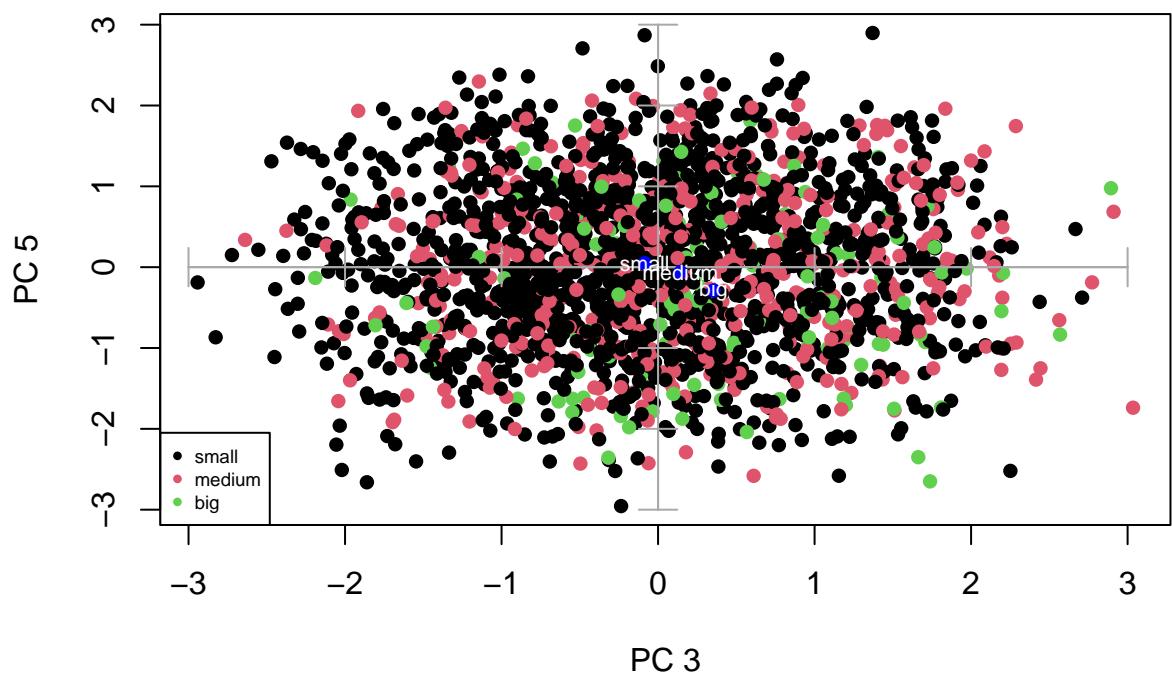
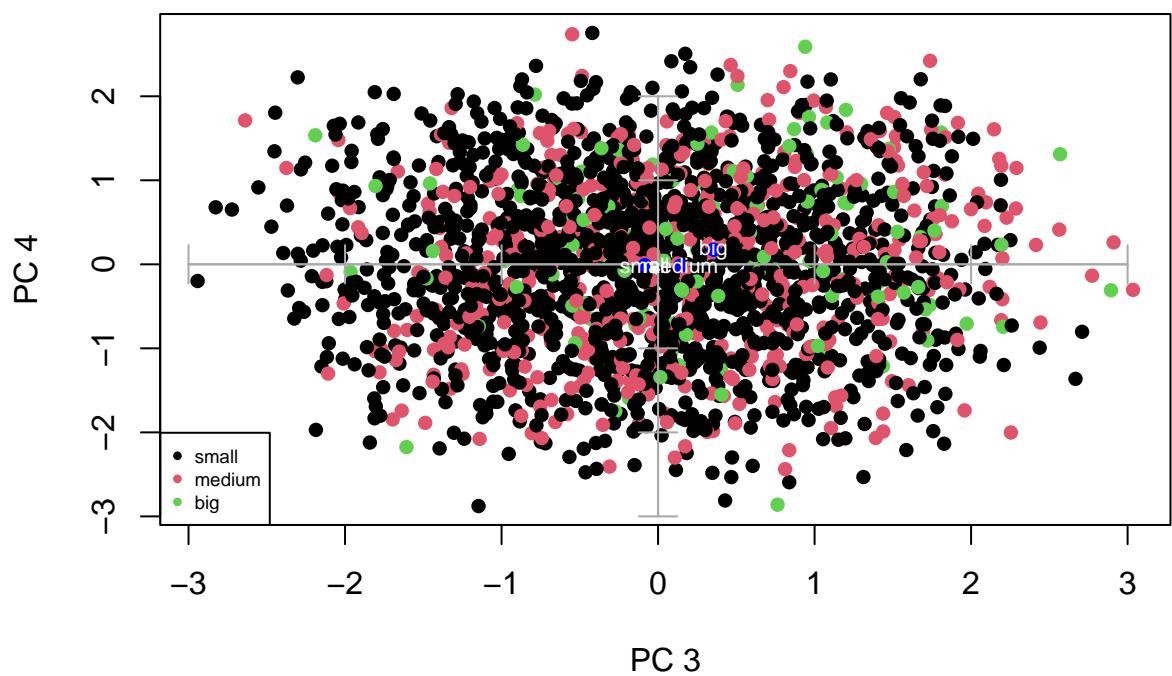


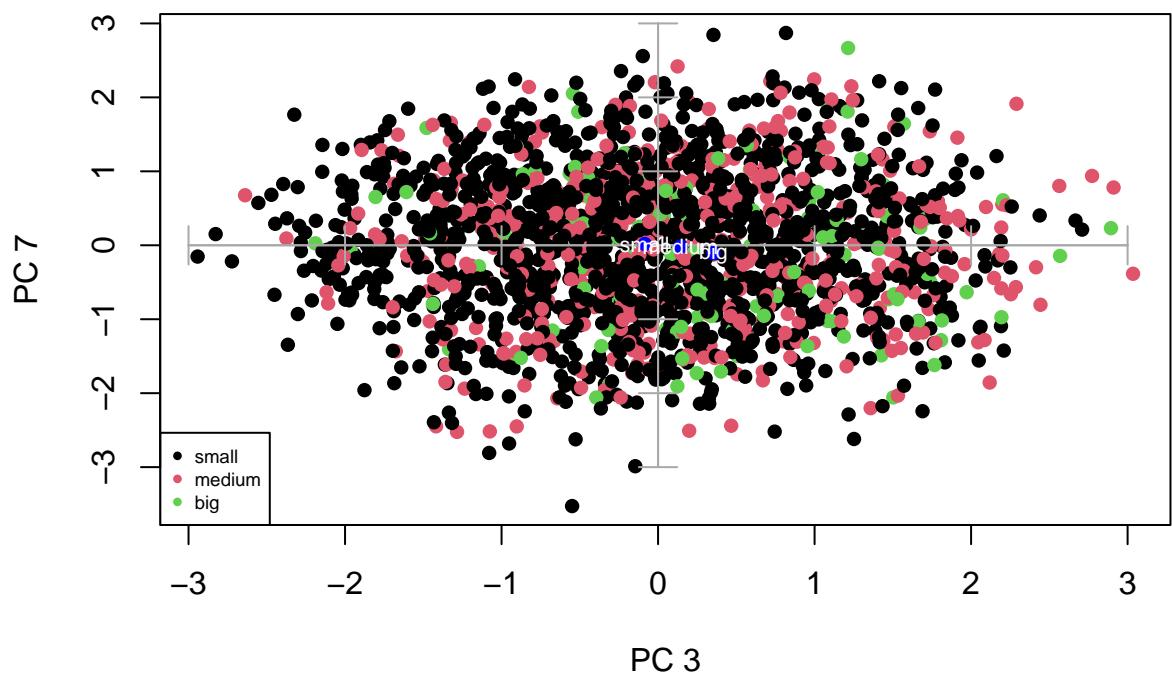
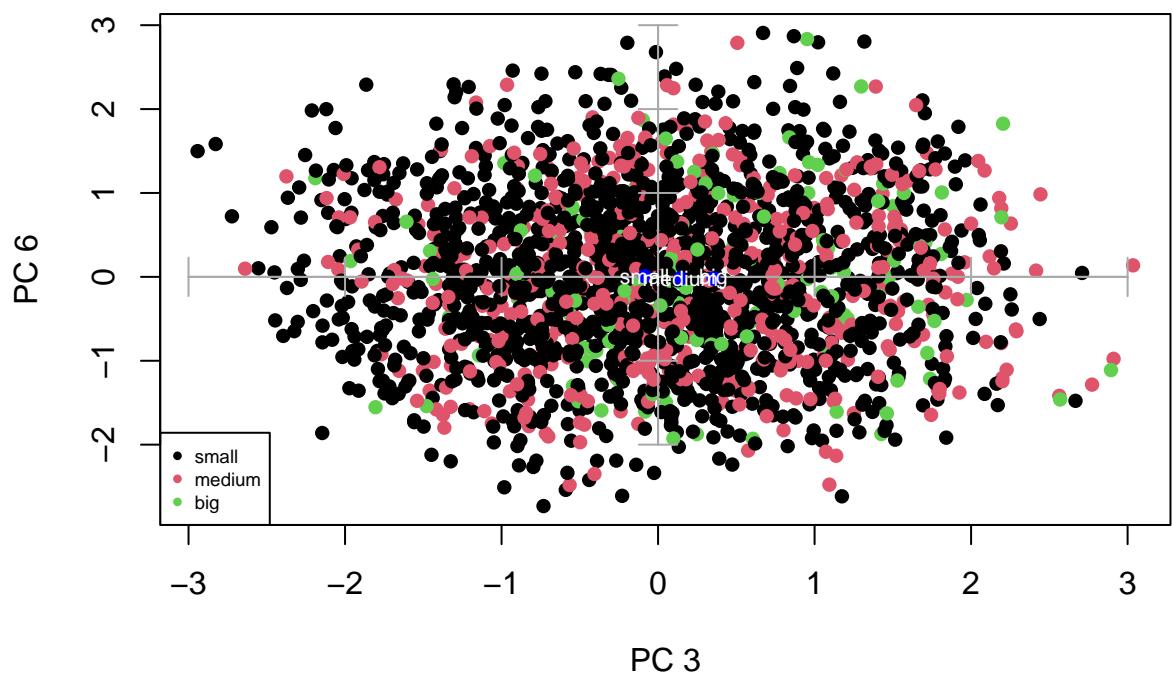


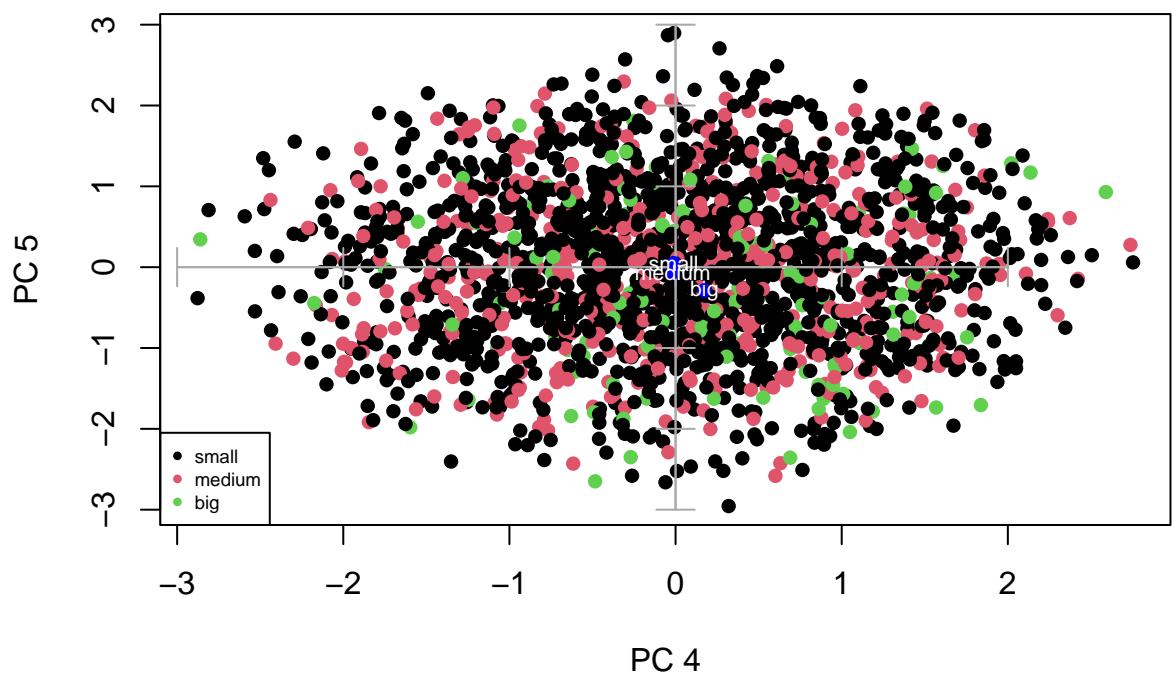
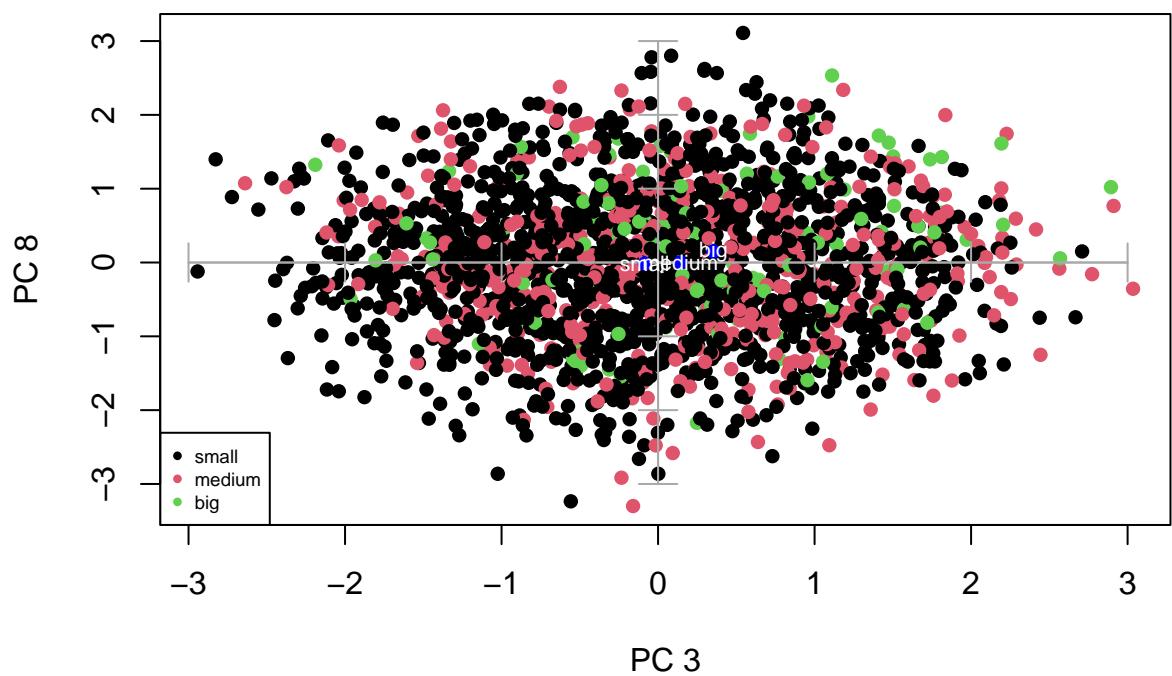


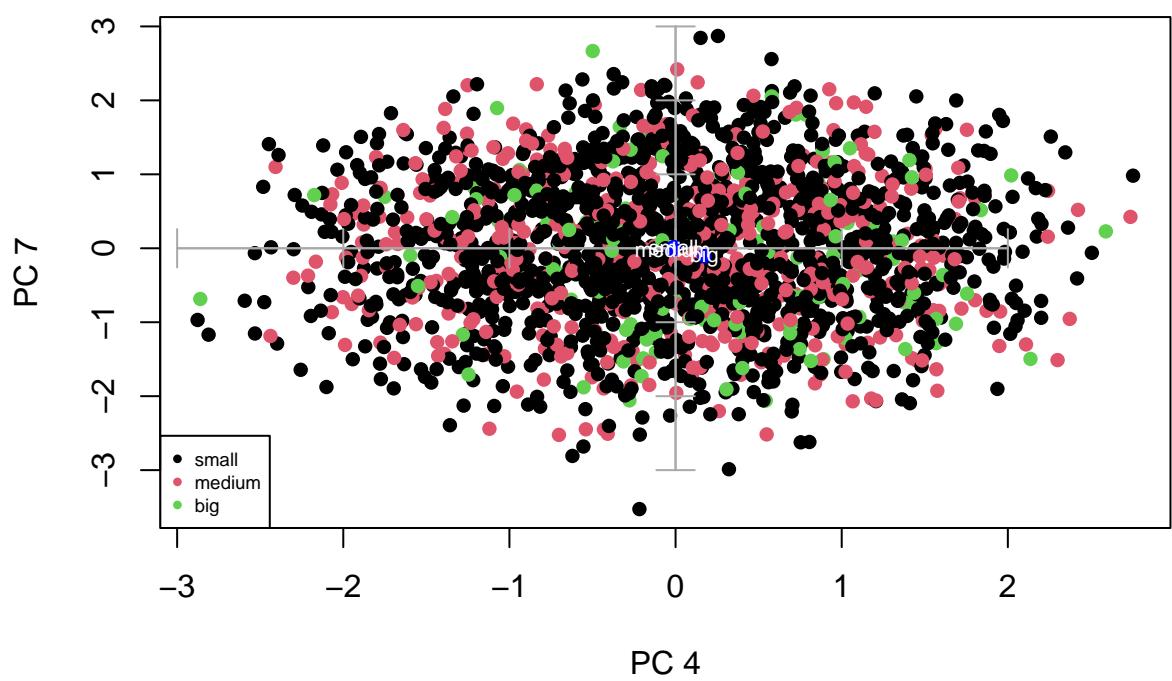
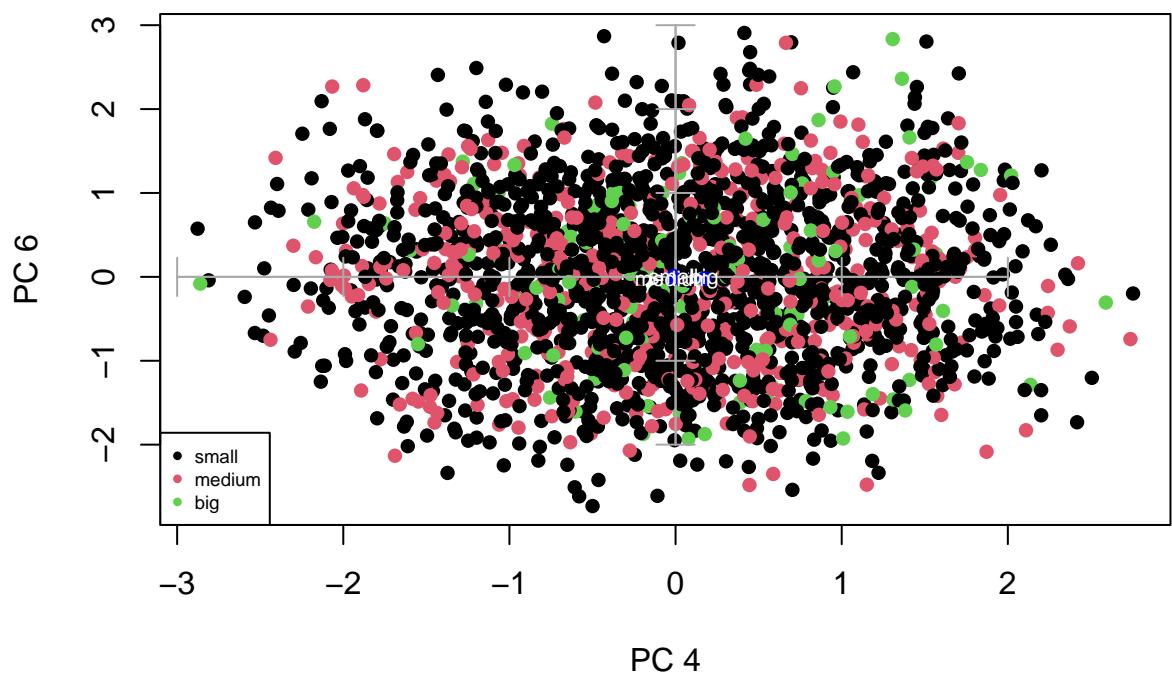


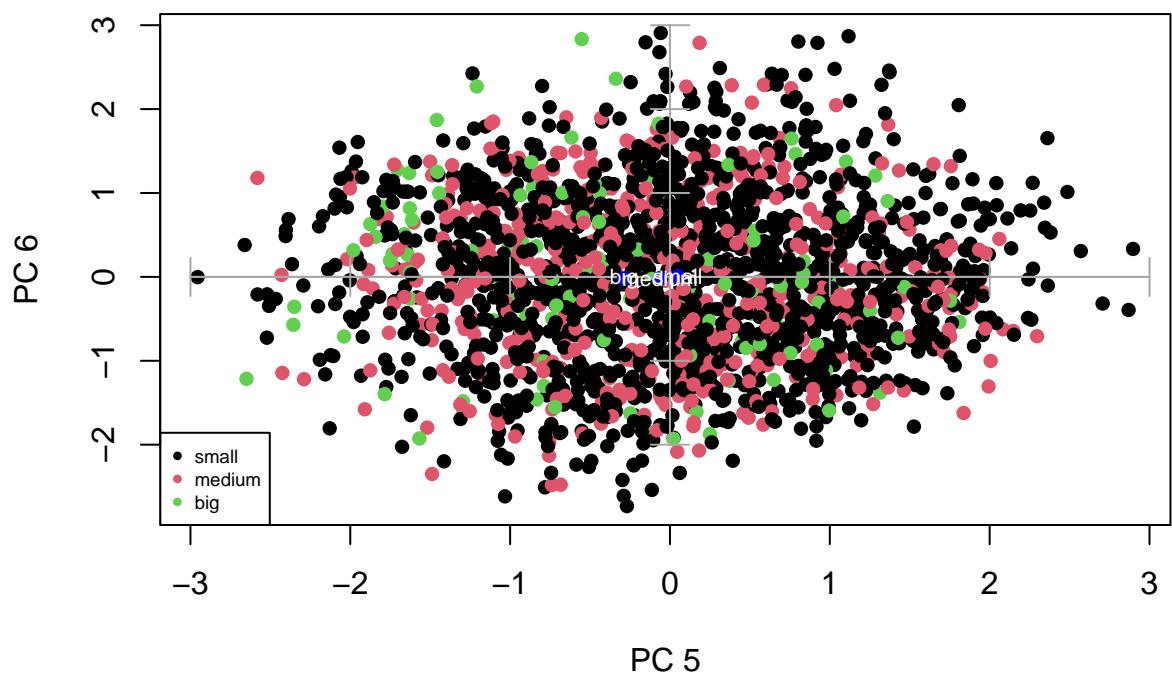
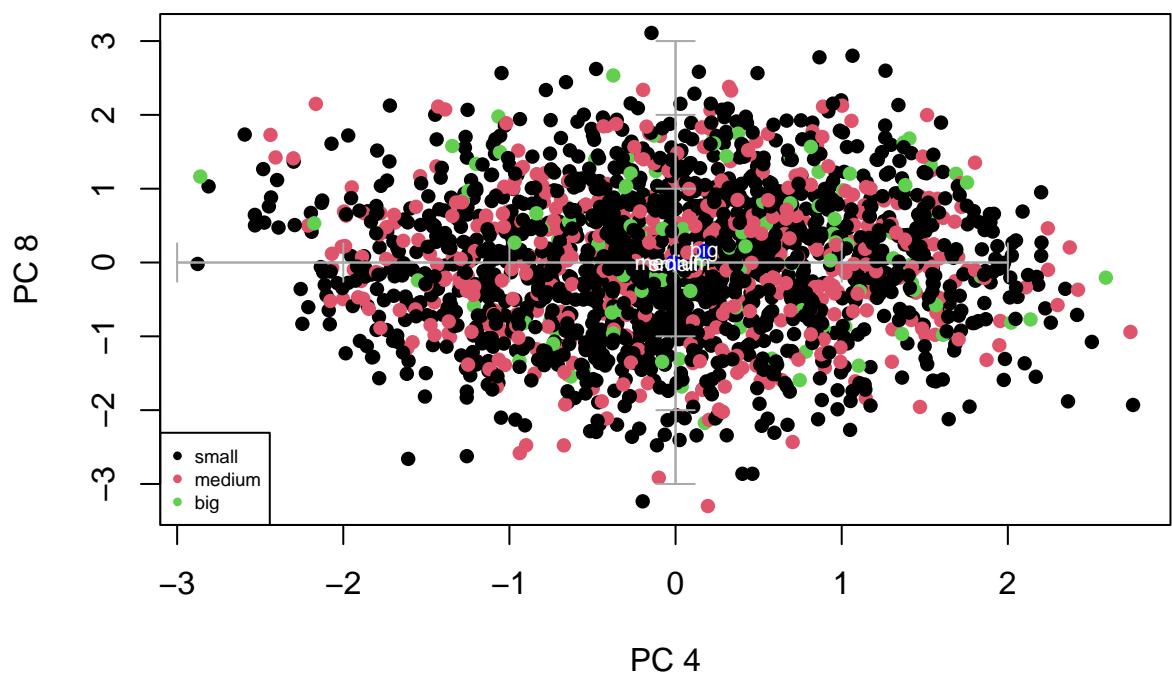


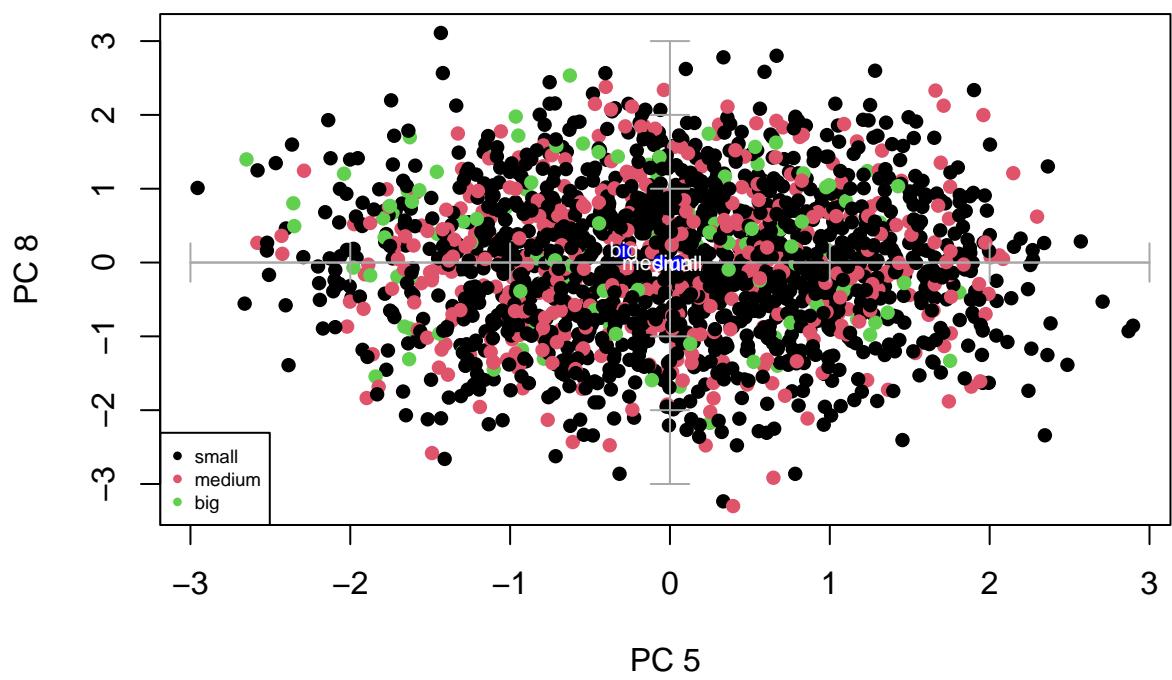
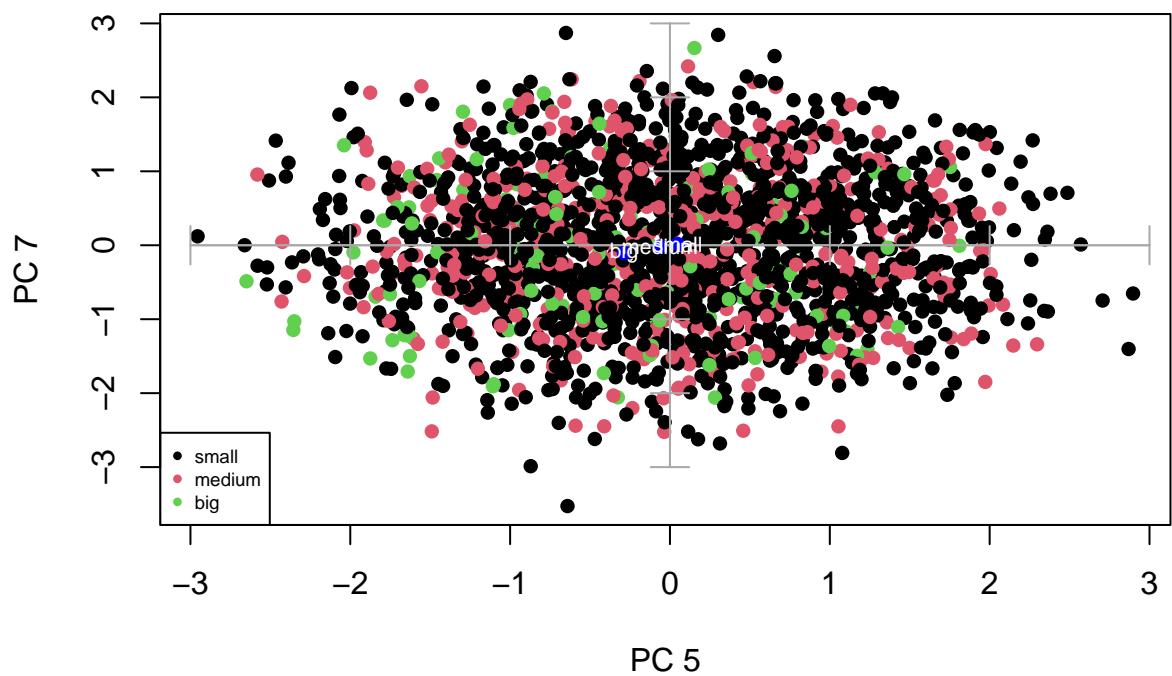


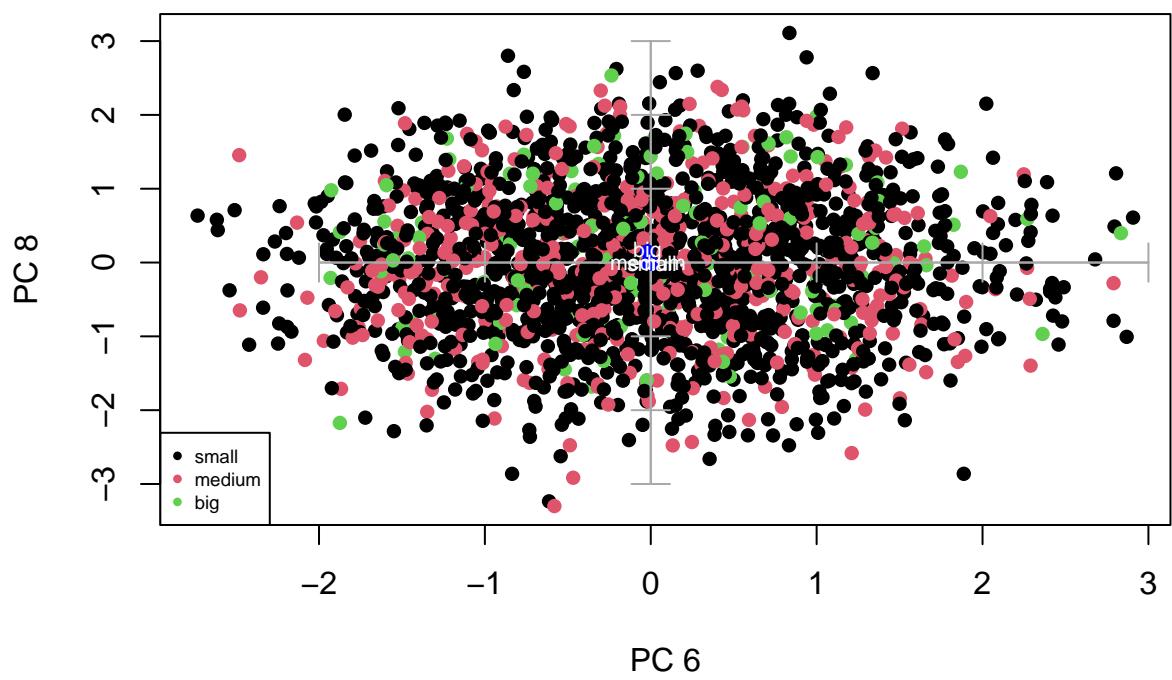
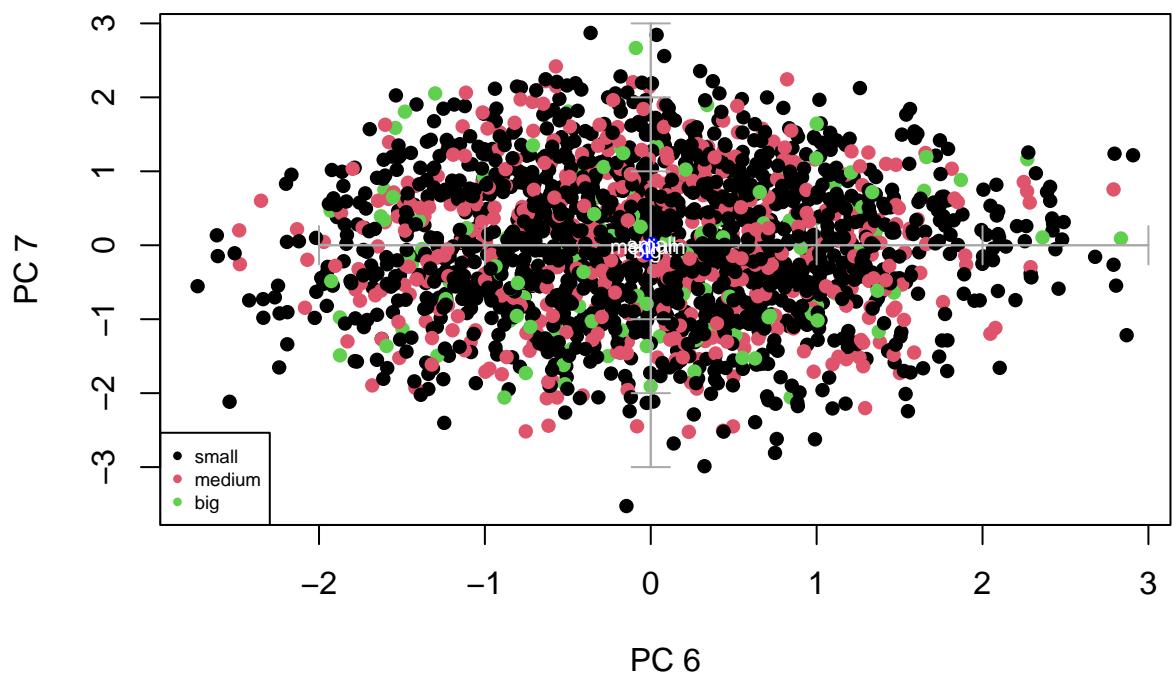


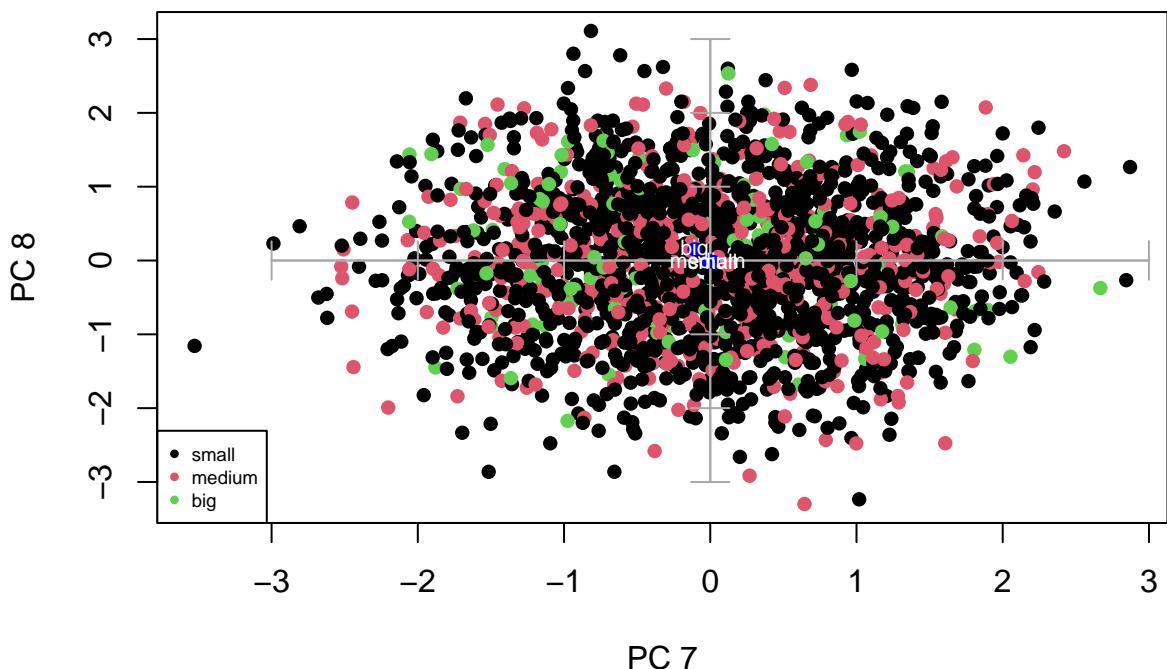










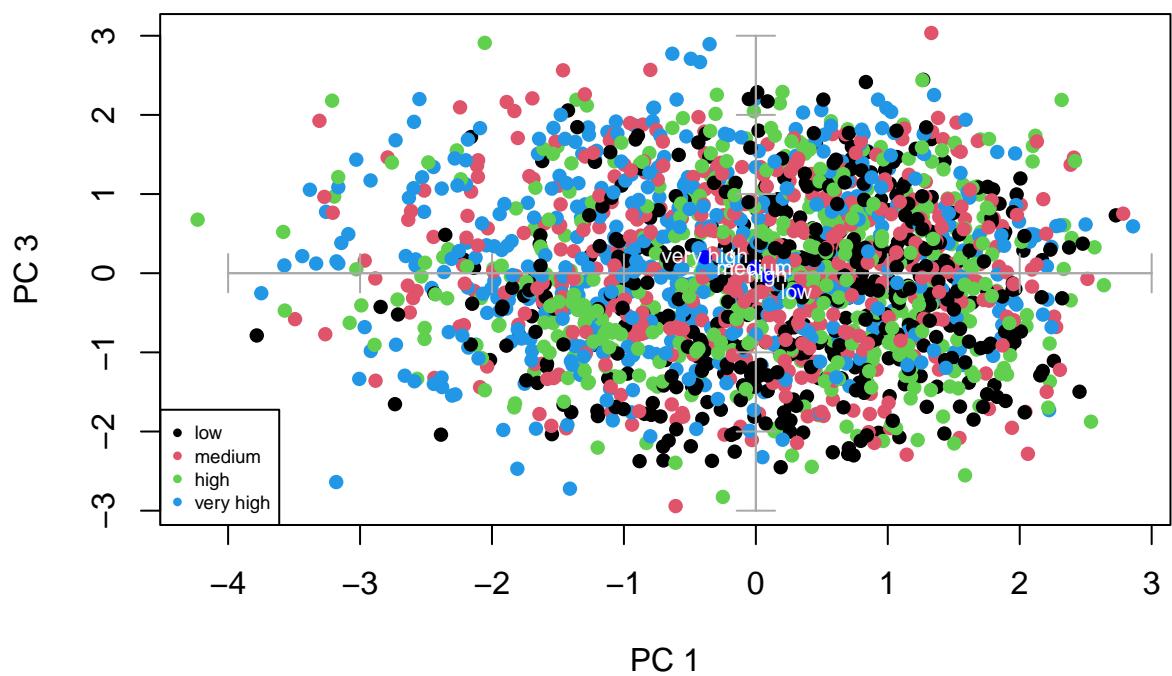
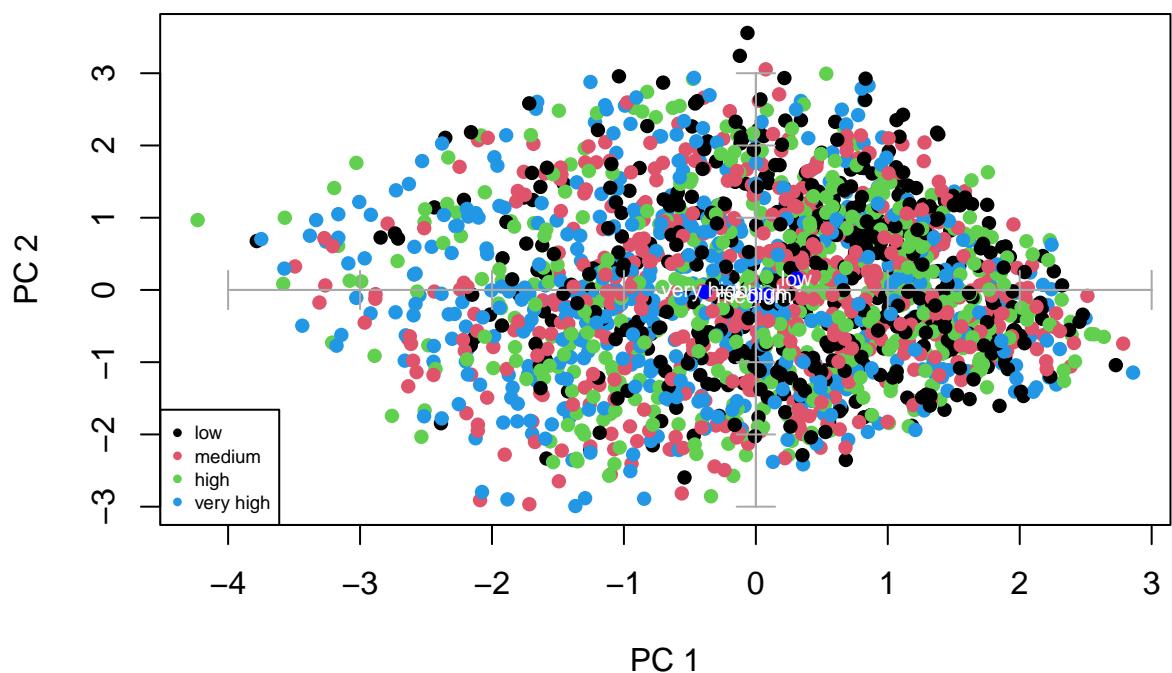


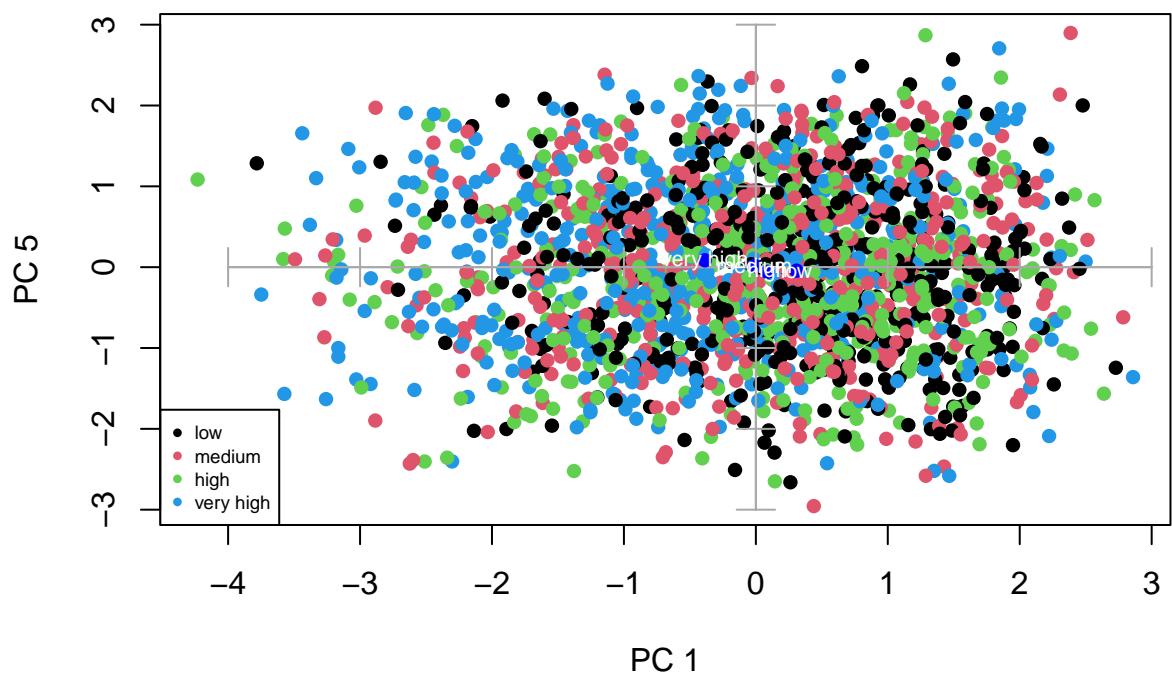
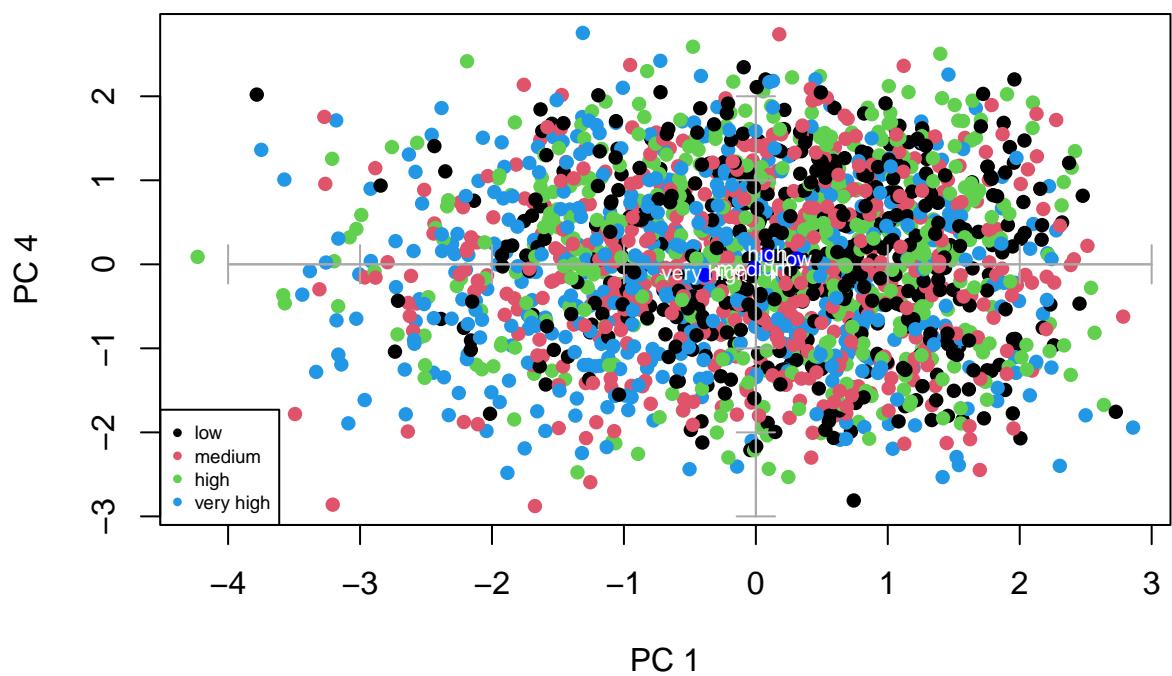
```

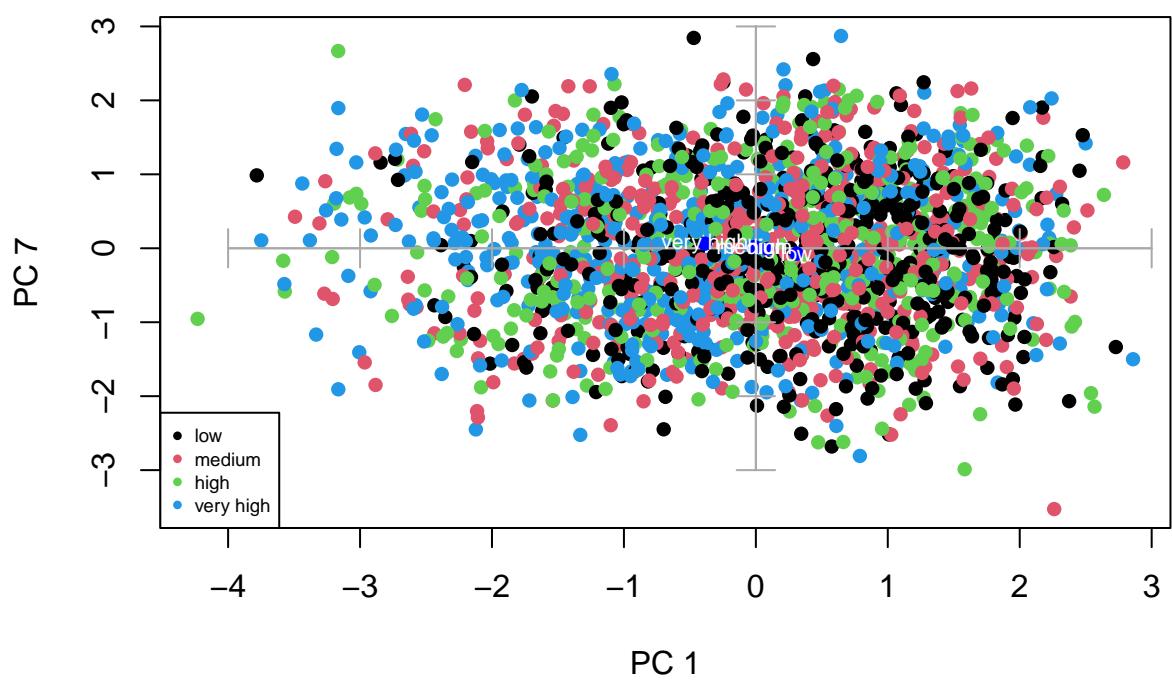
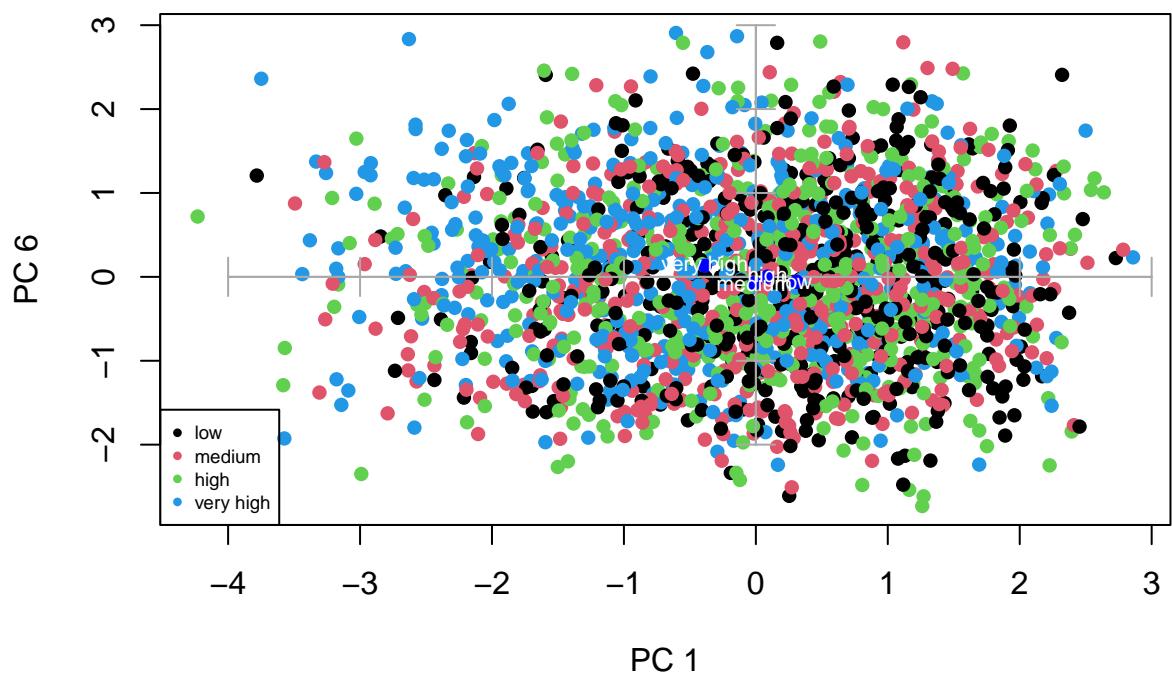
for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$price_range
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
    legend("bottomleft",levels(varcat),pch=16,col=c(1:4), cex=0.6)

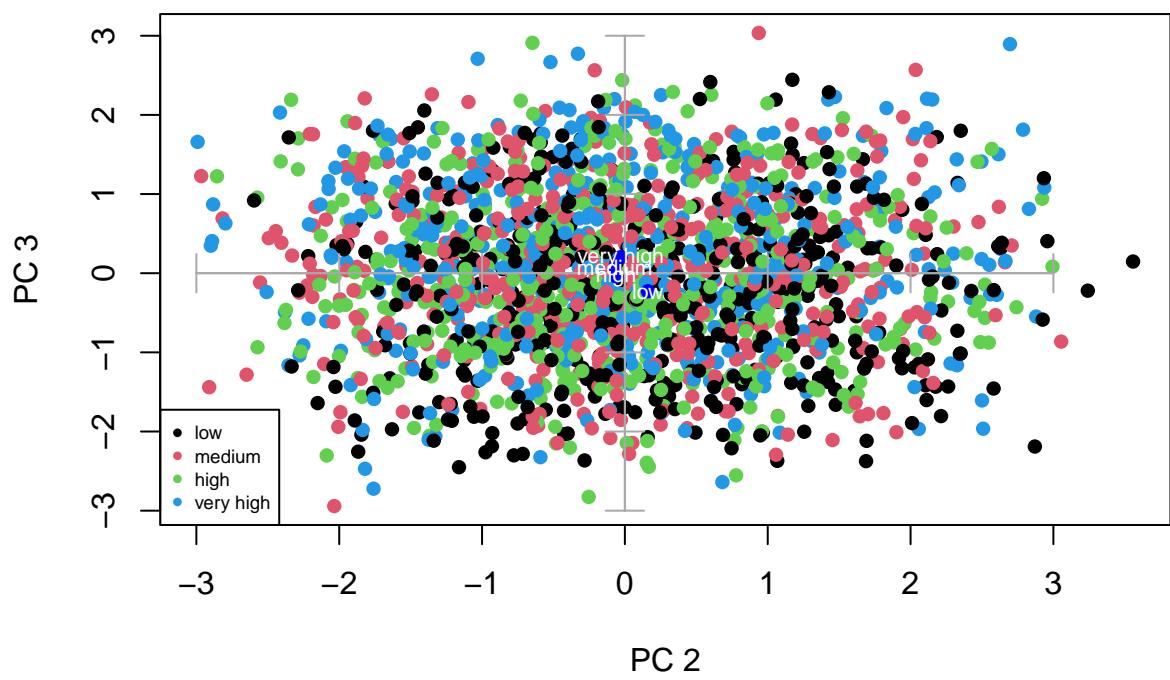
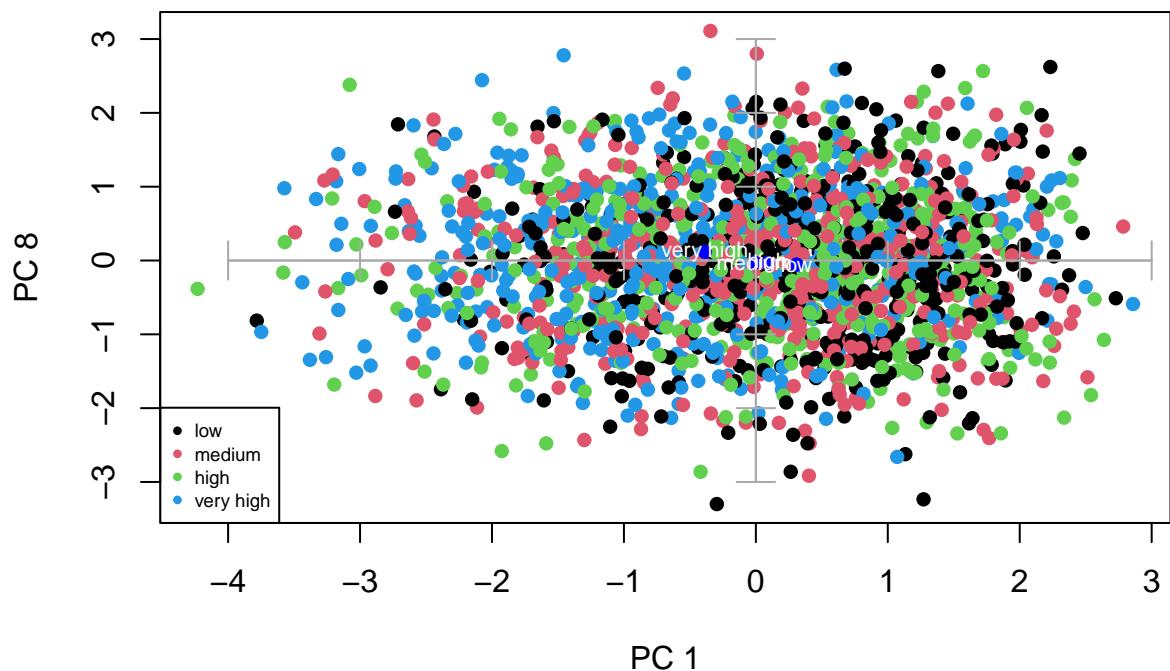
    #select your qualitative variable
    fdic1 = tapply(Psi[,i],varcat,mean)
    fdic2 = tapply(Psi[,j],varcat,mean)
    points(fdic1,fdic2,pch=16,col="blue")
    text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
  }
}

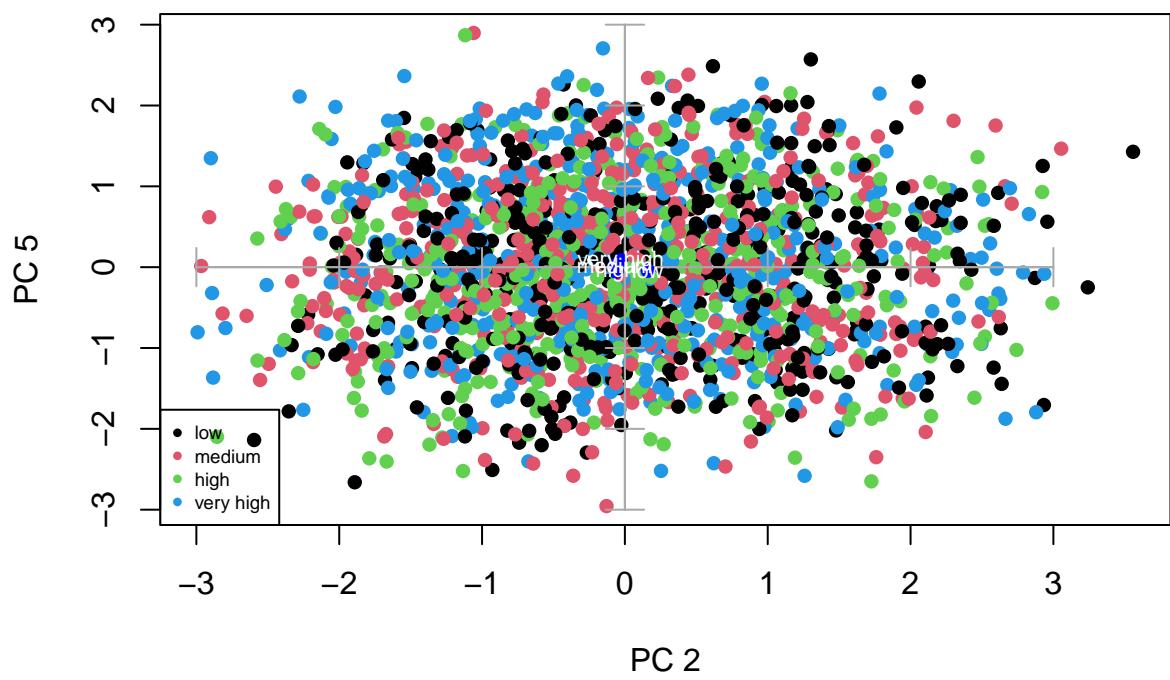
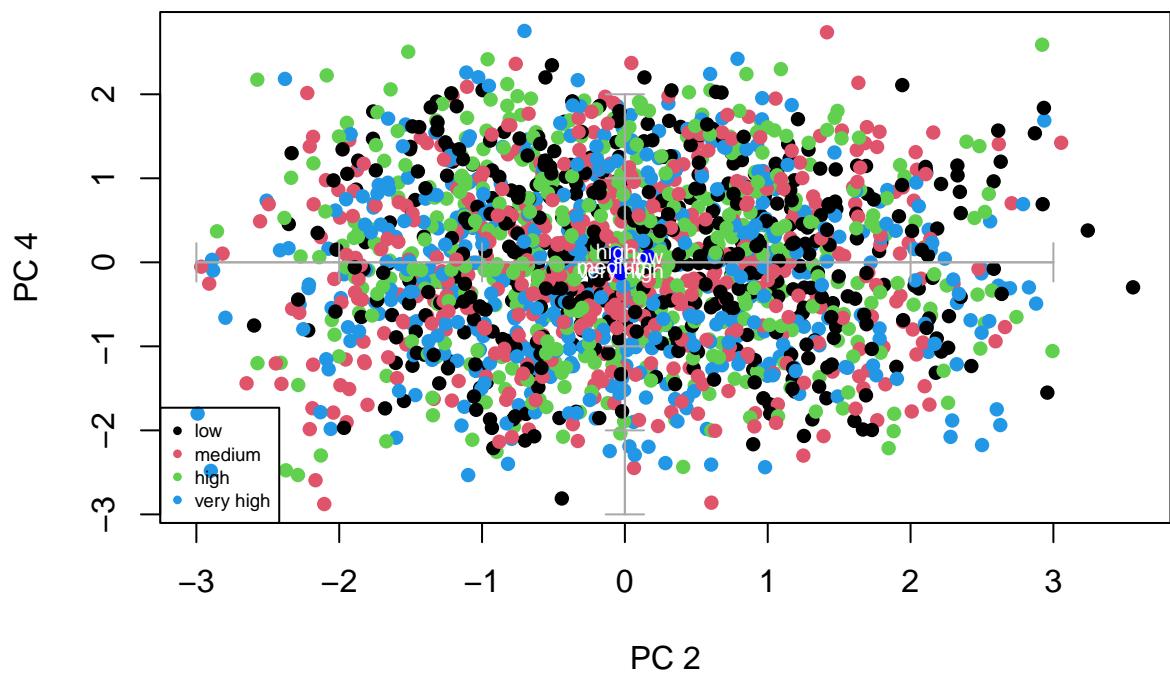
```

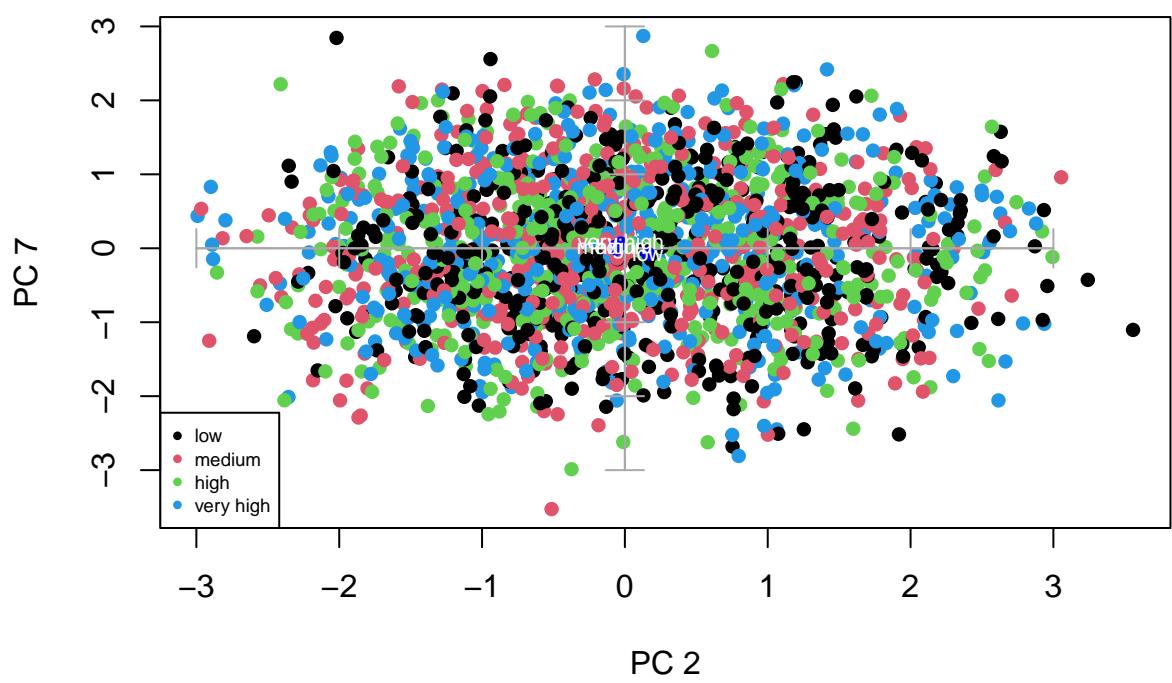
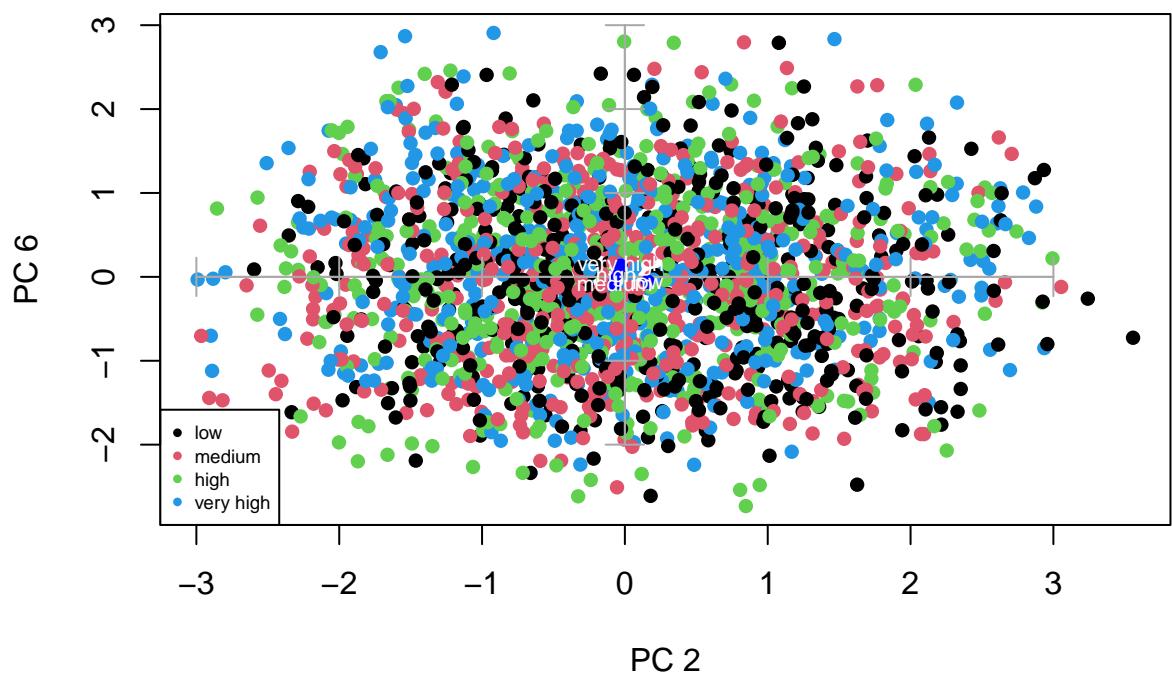


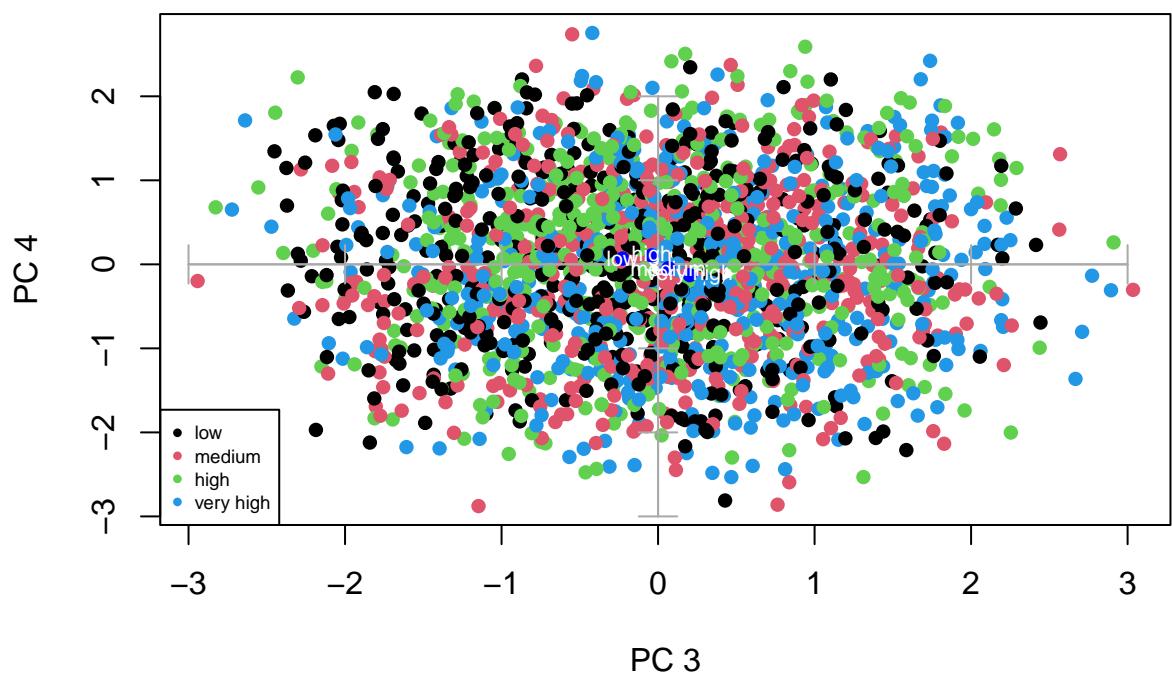
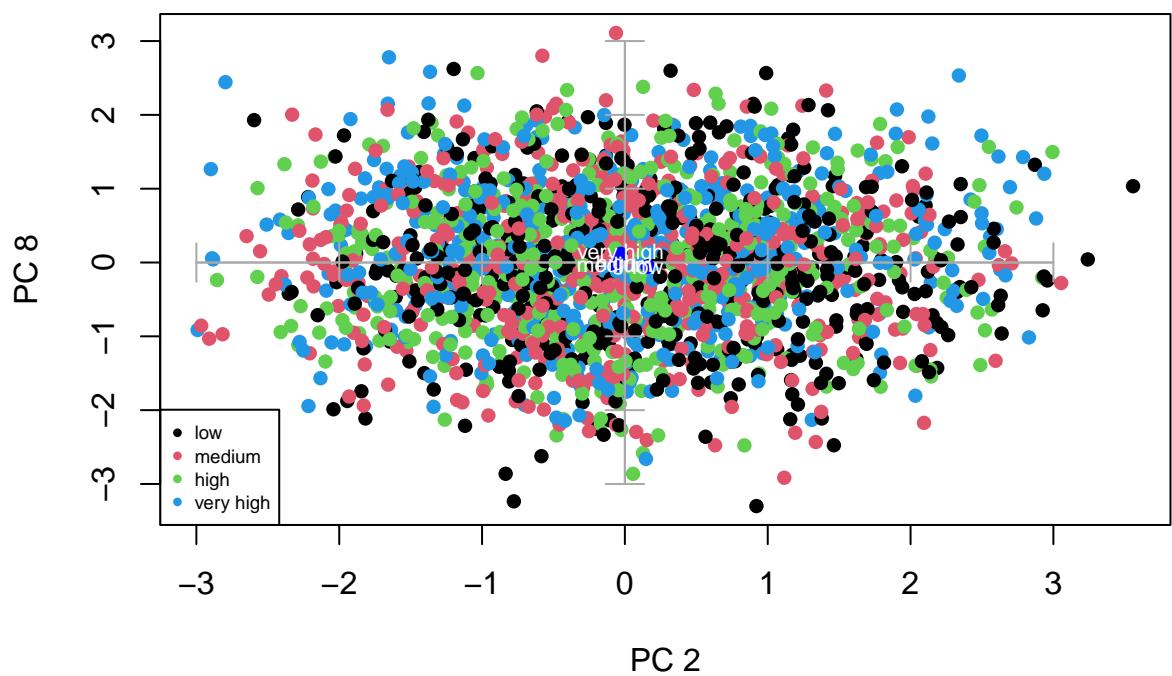


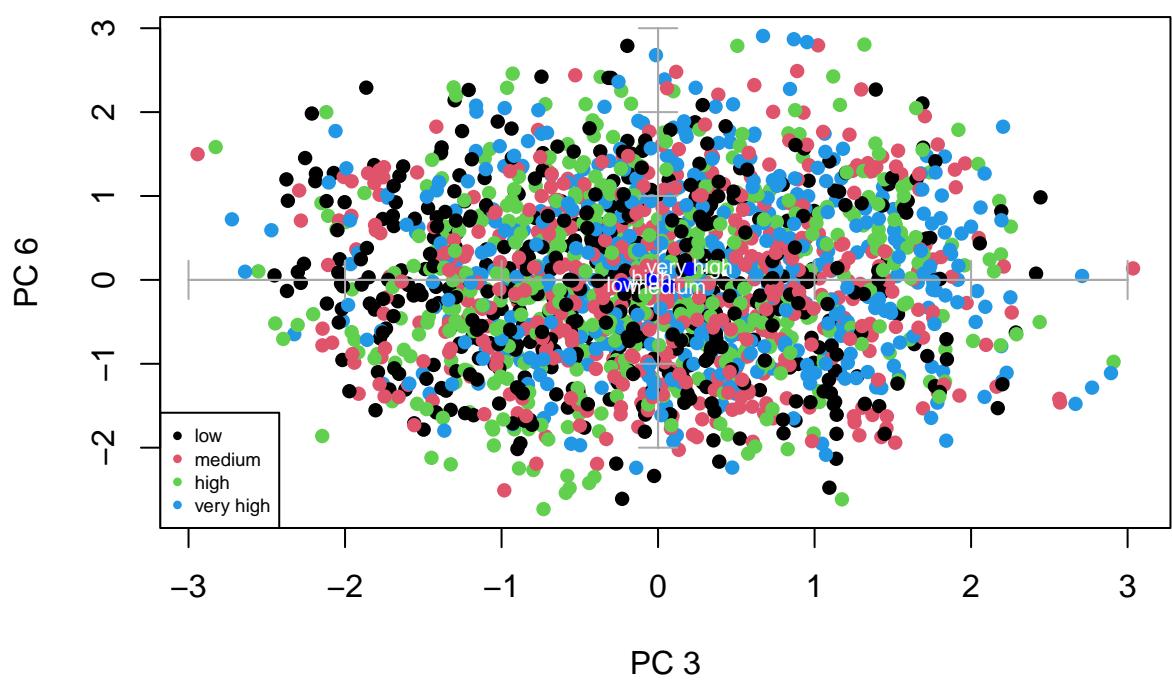
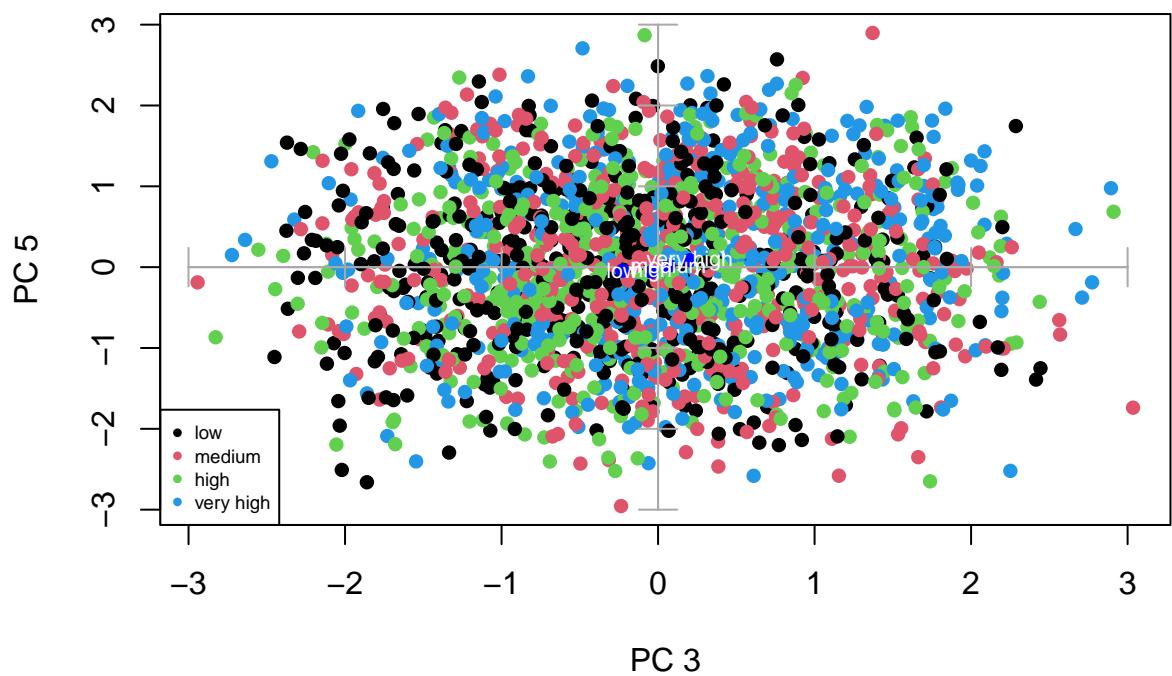


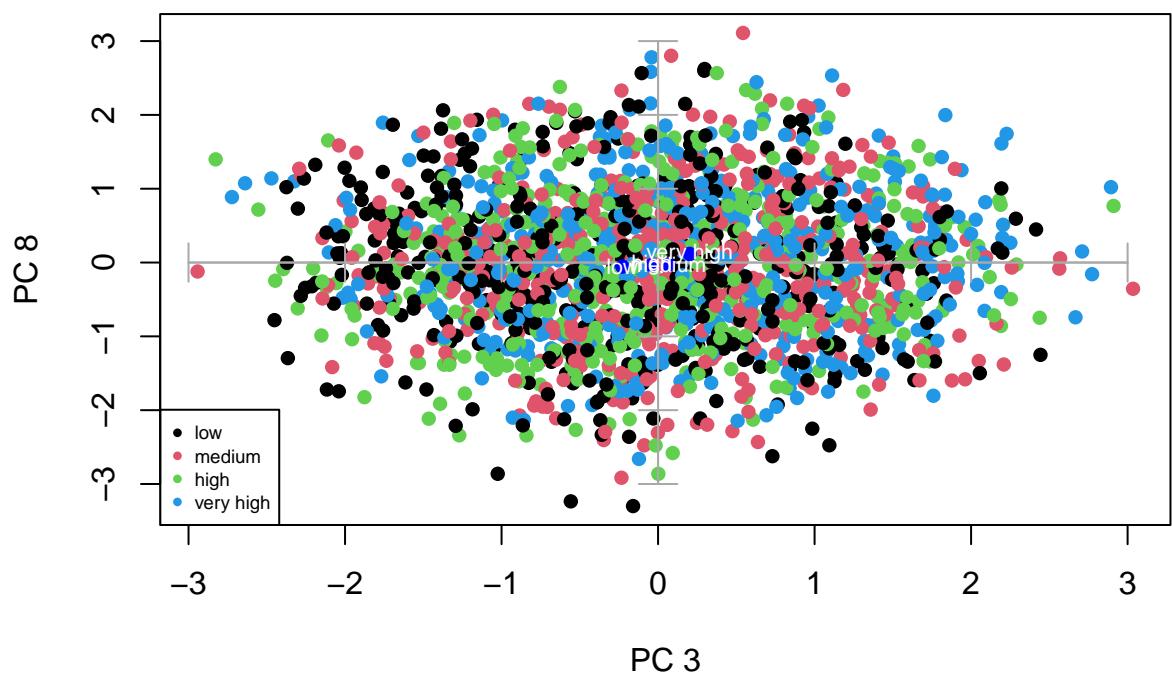
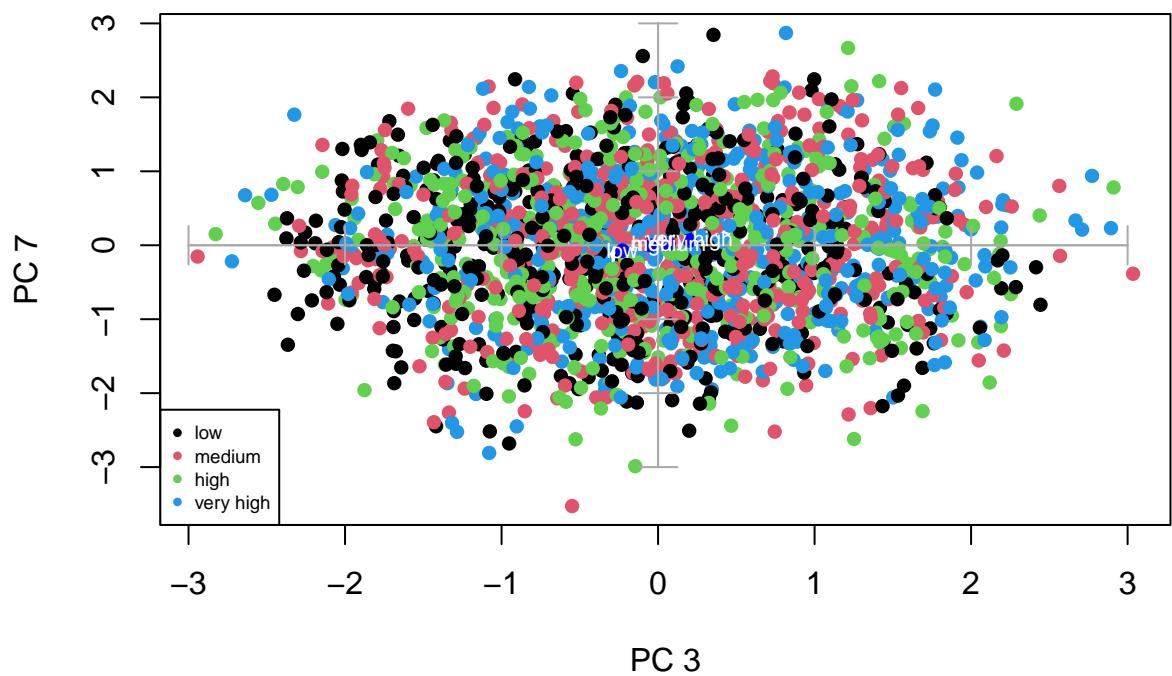


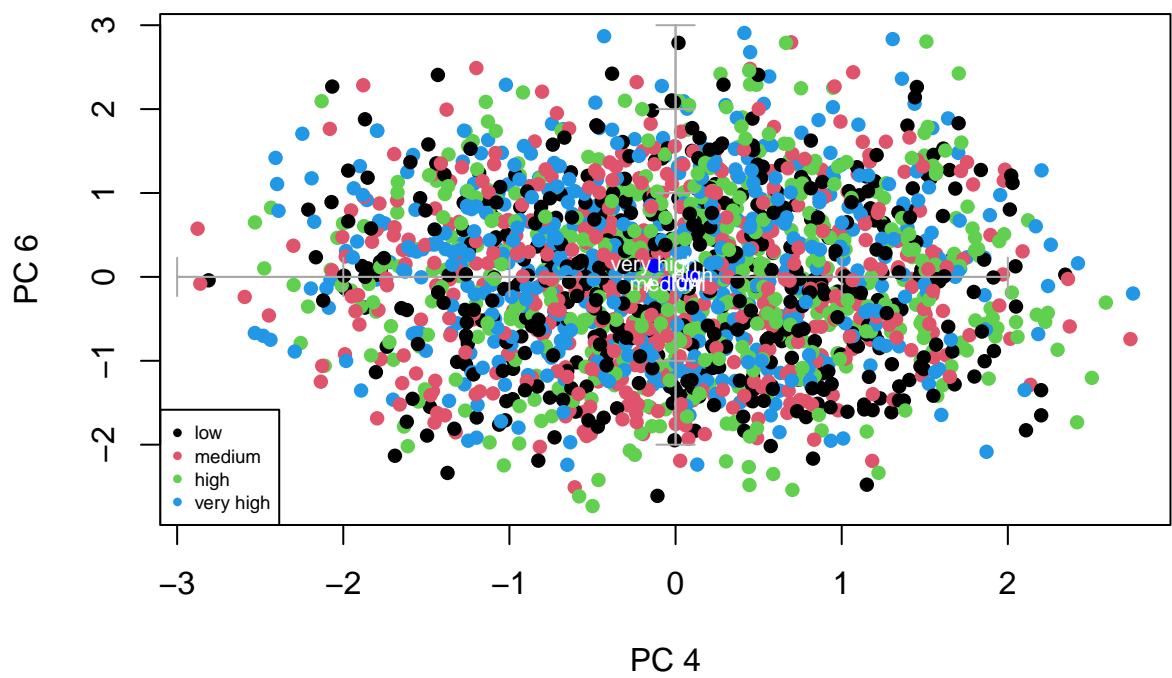
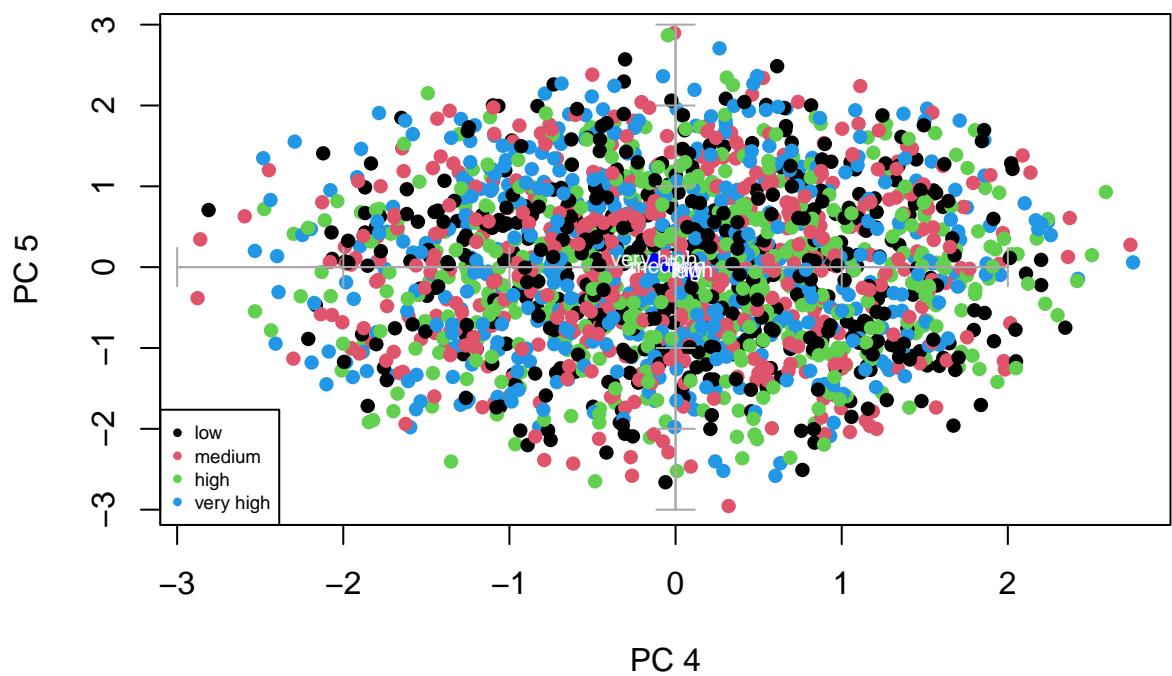


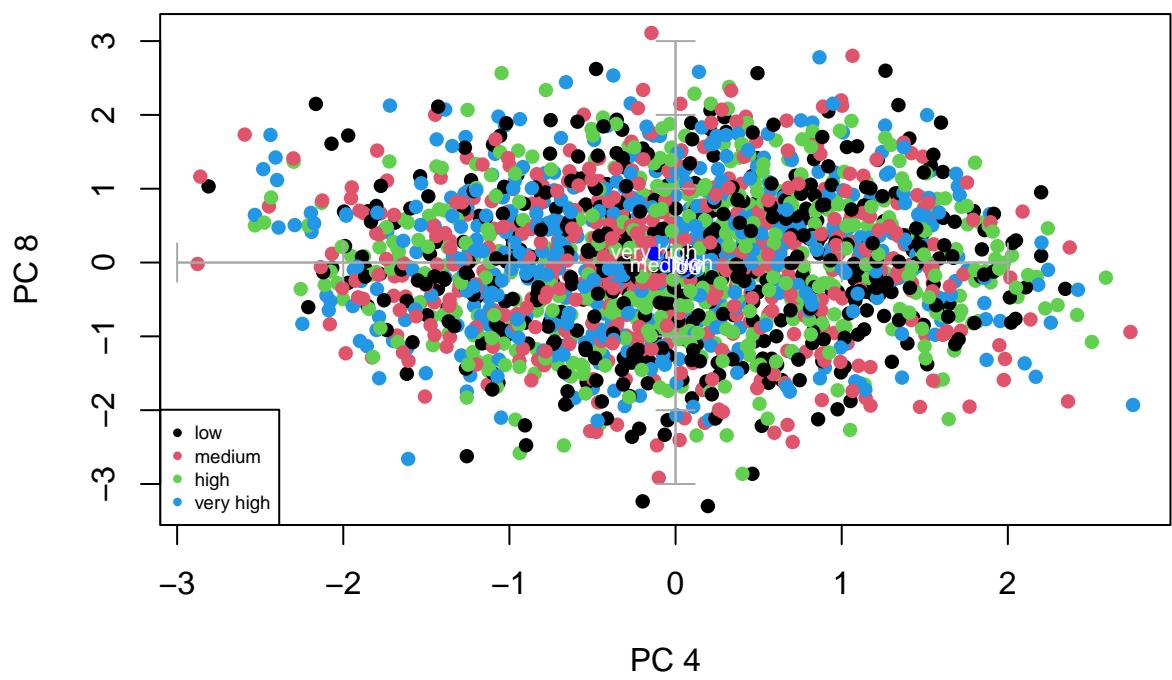
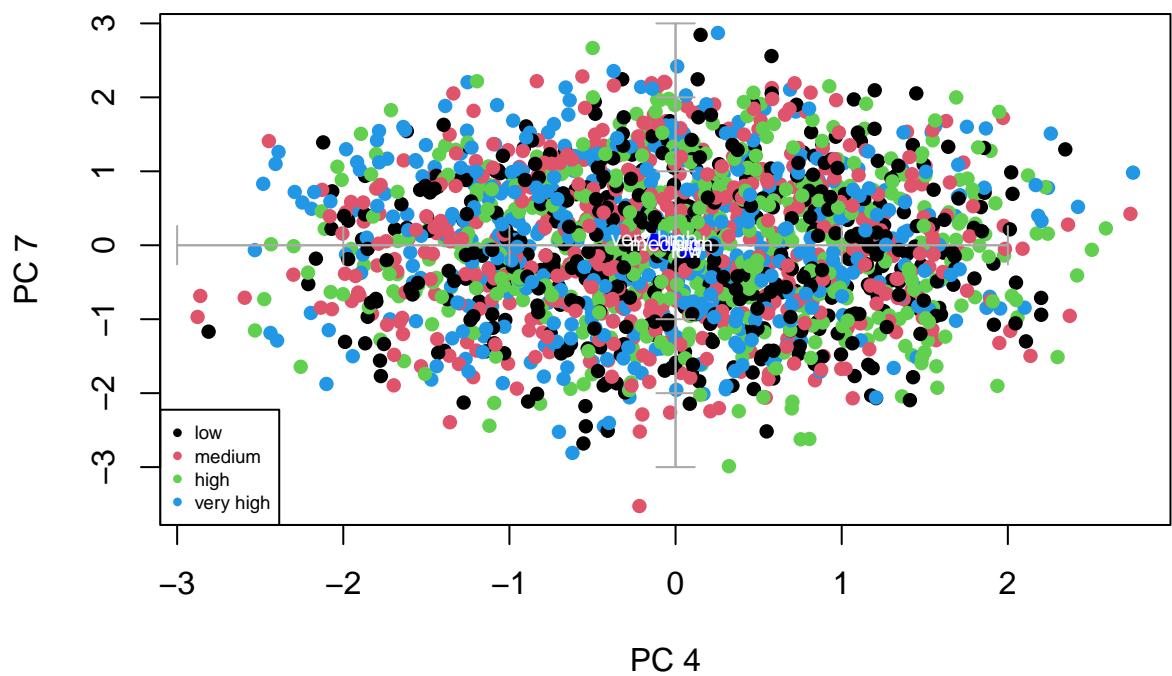


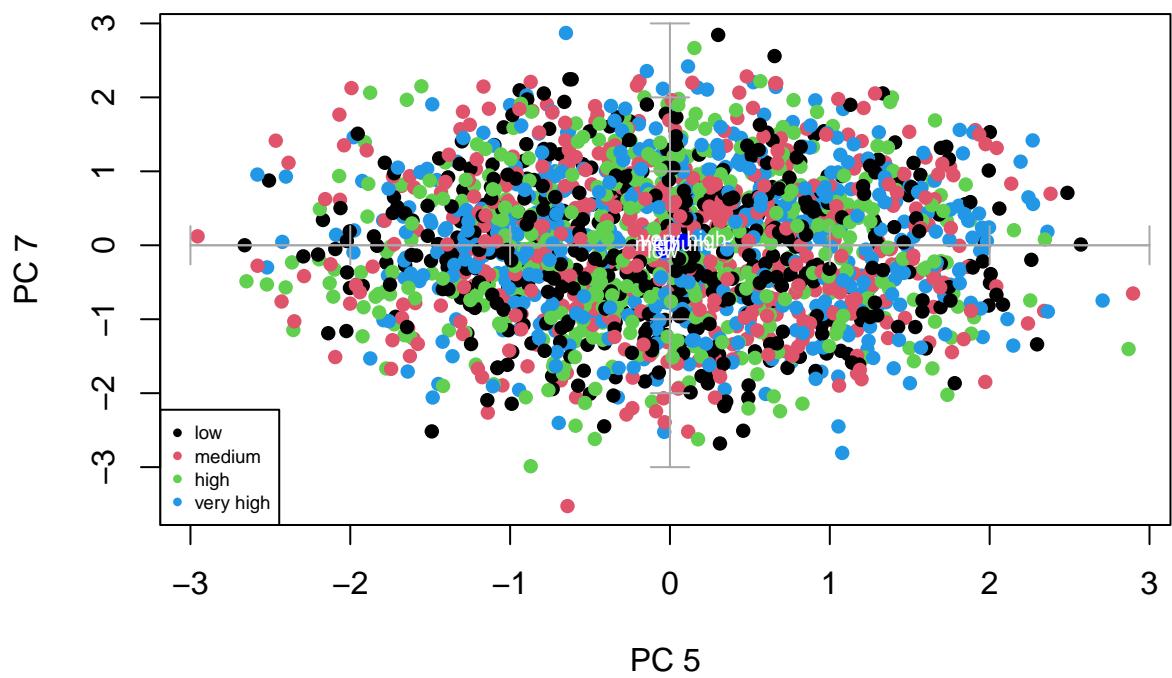
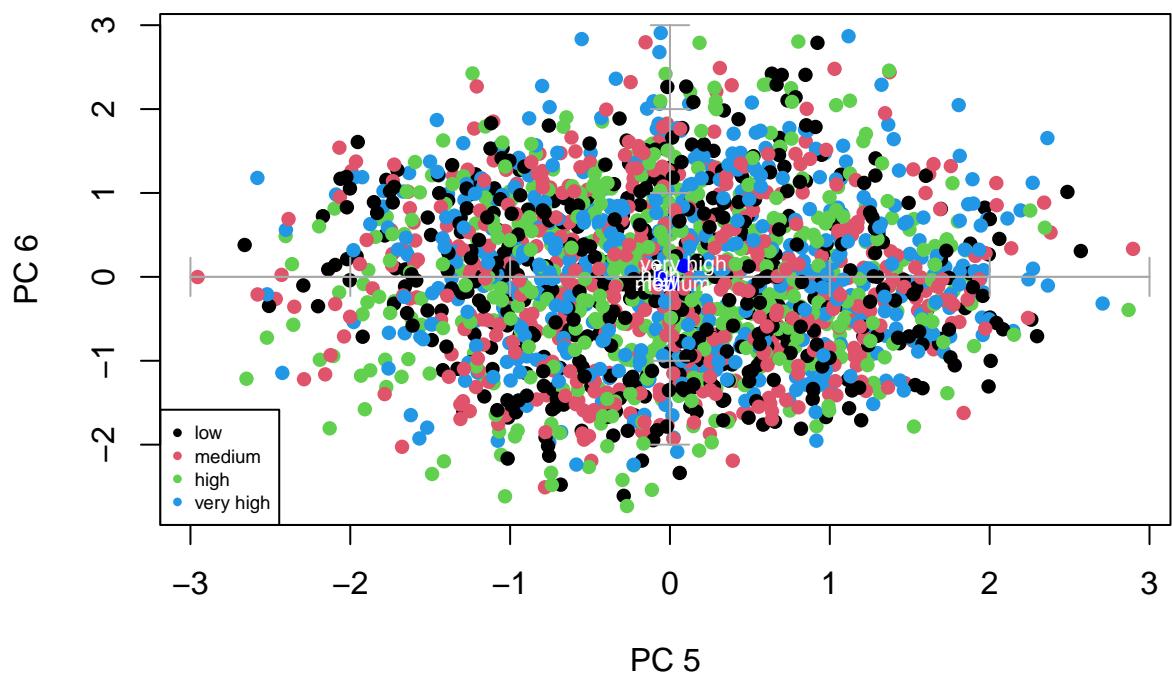


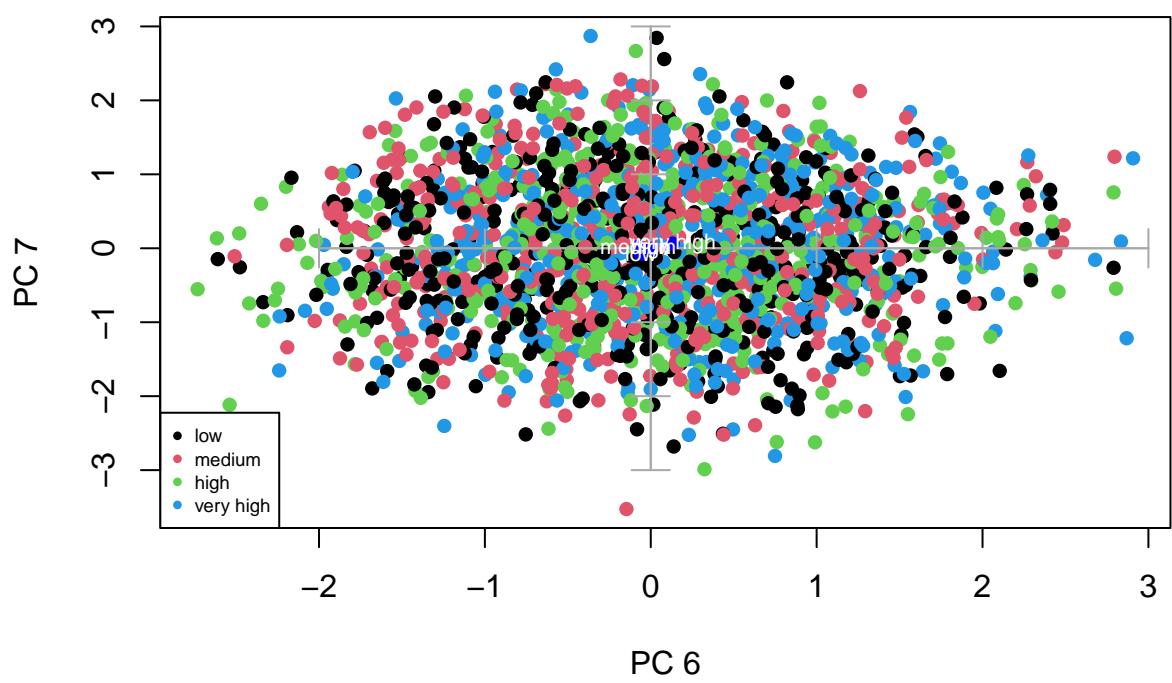
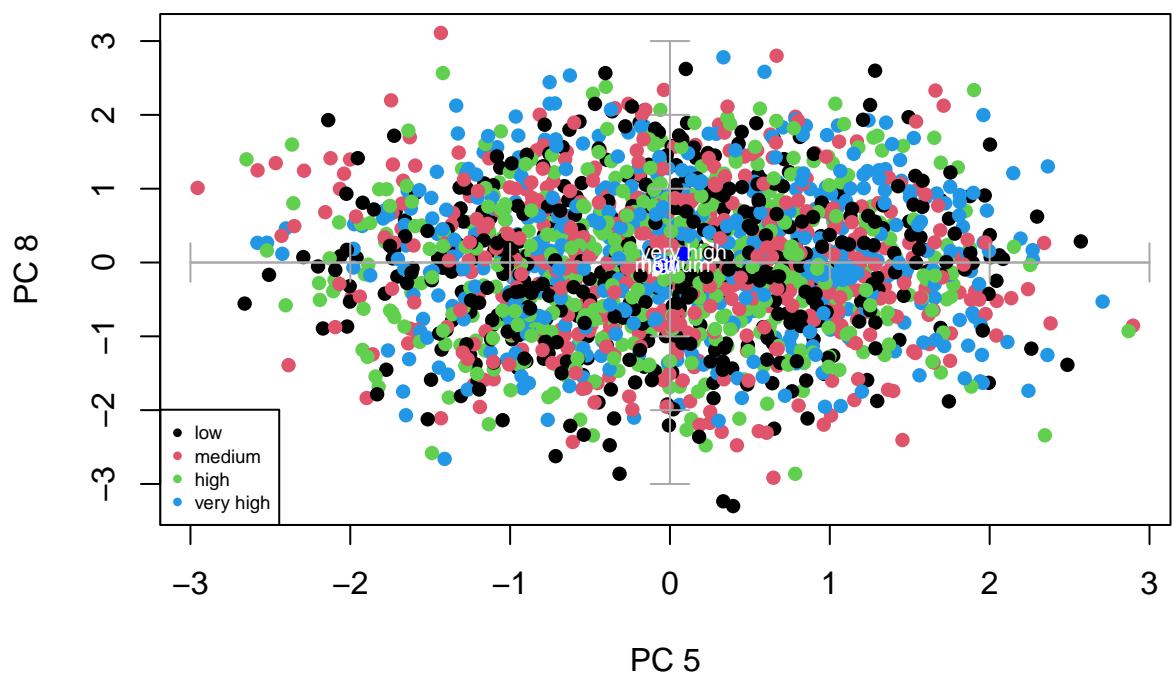


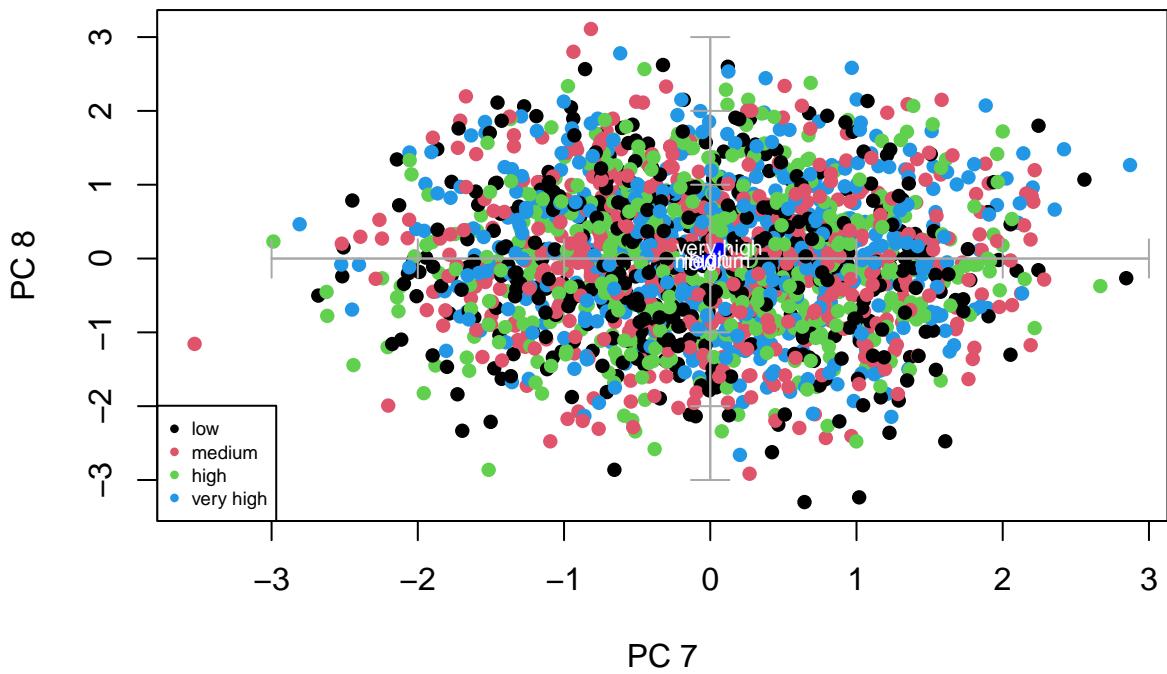
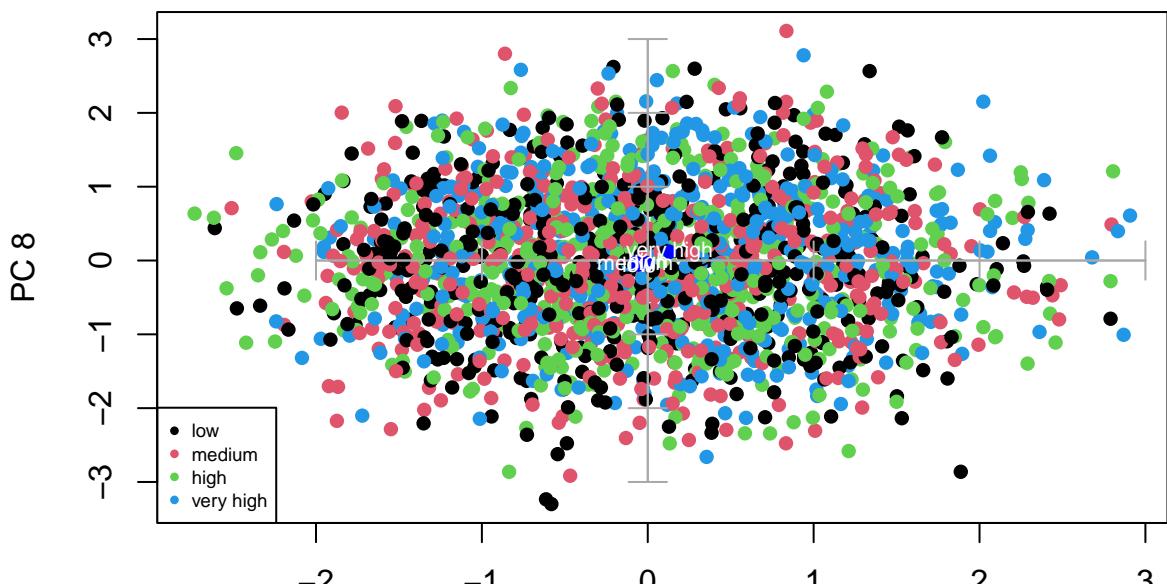












```

for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$wifi
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
  }
}
  
```

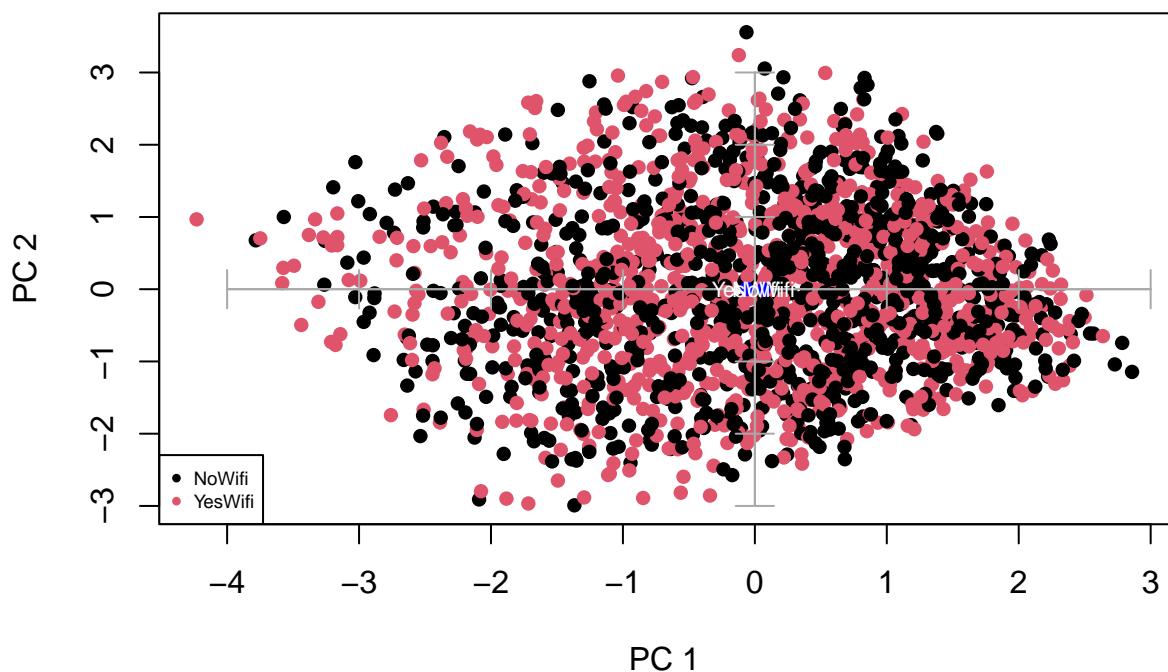
```

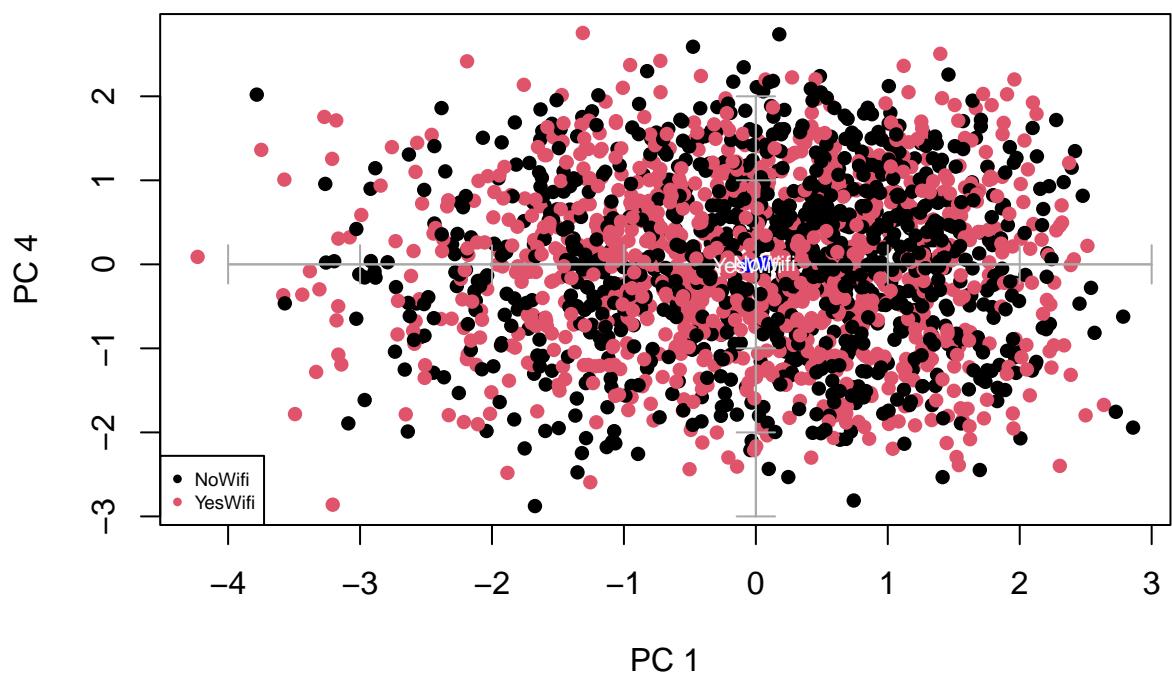
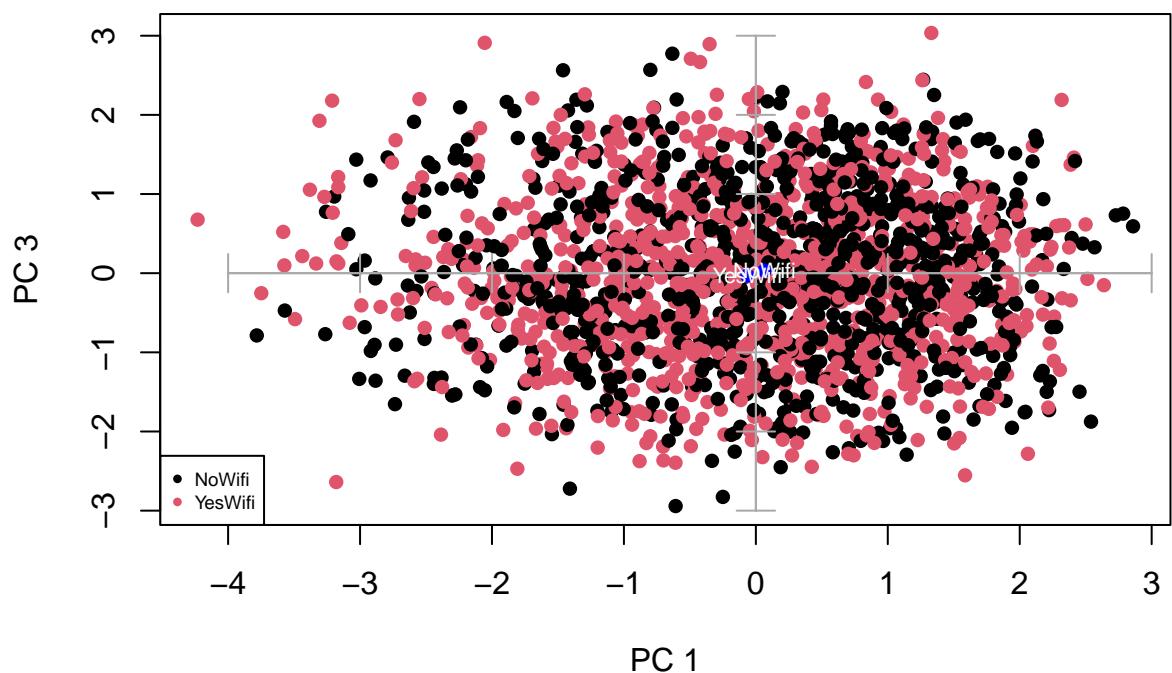
legend("bottomleft",levels(varcat),pch=16,col=c(1:2), cex=0.6)

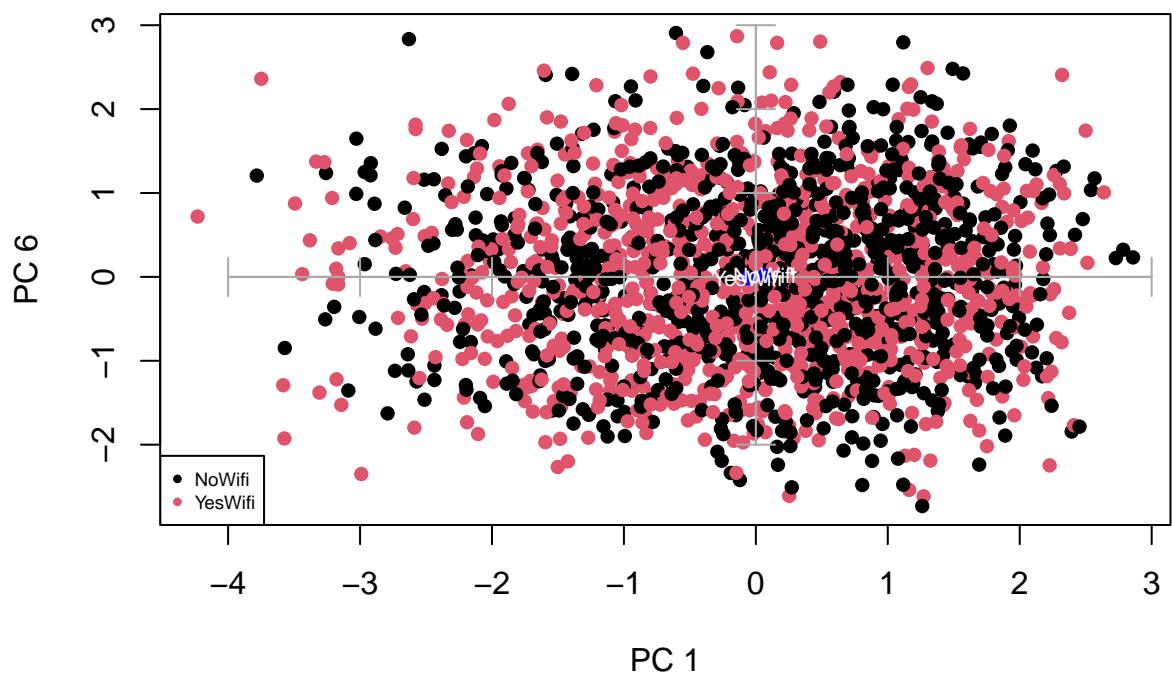
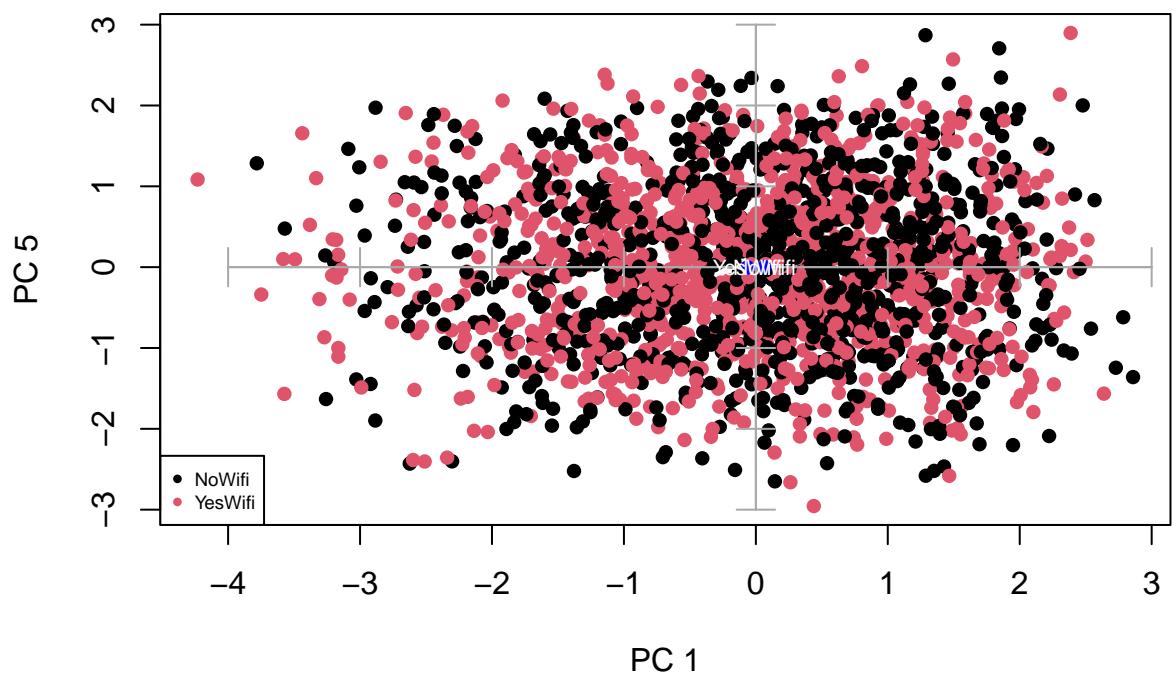
#select your qualitative variable
fdic1 = tapply(Psi[,i],varcat,mean)
fdic2 = tapply(Psi[,j],varcat,mean)
points(fdic1,fdic2,pch=16,col="blue")
text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
}

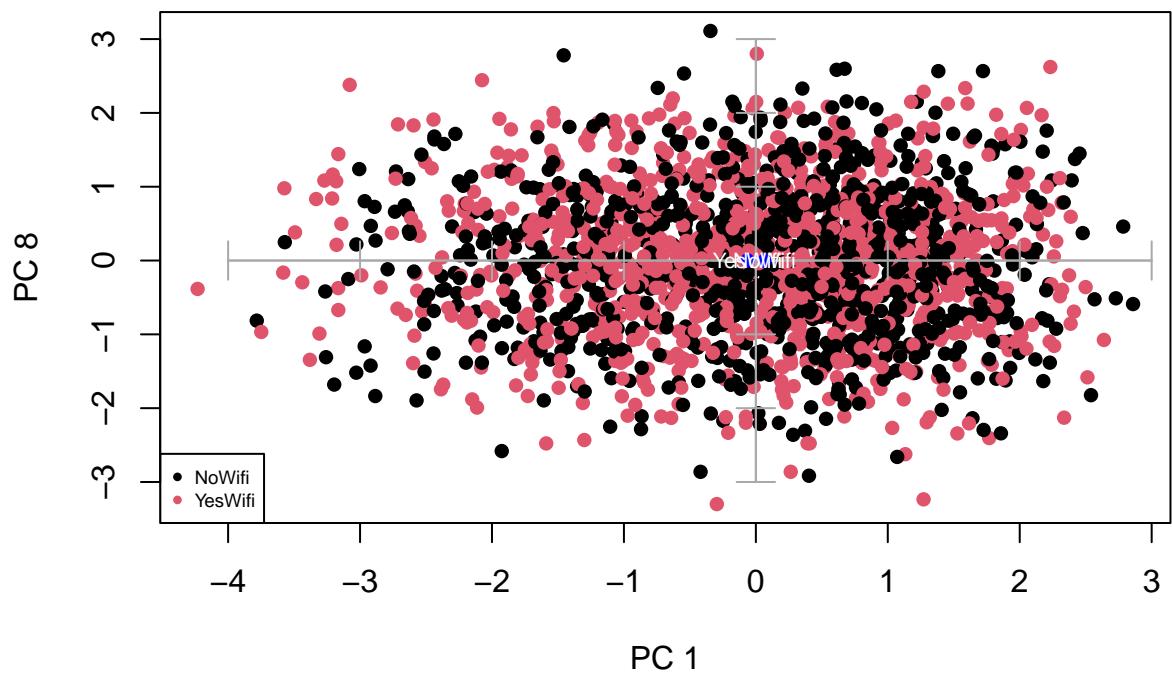
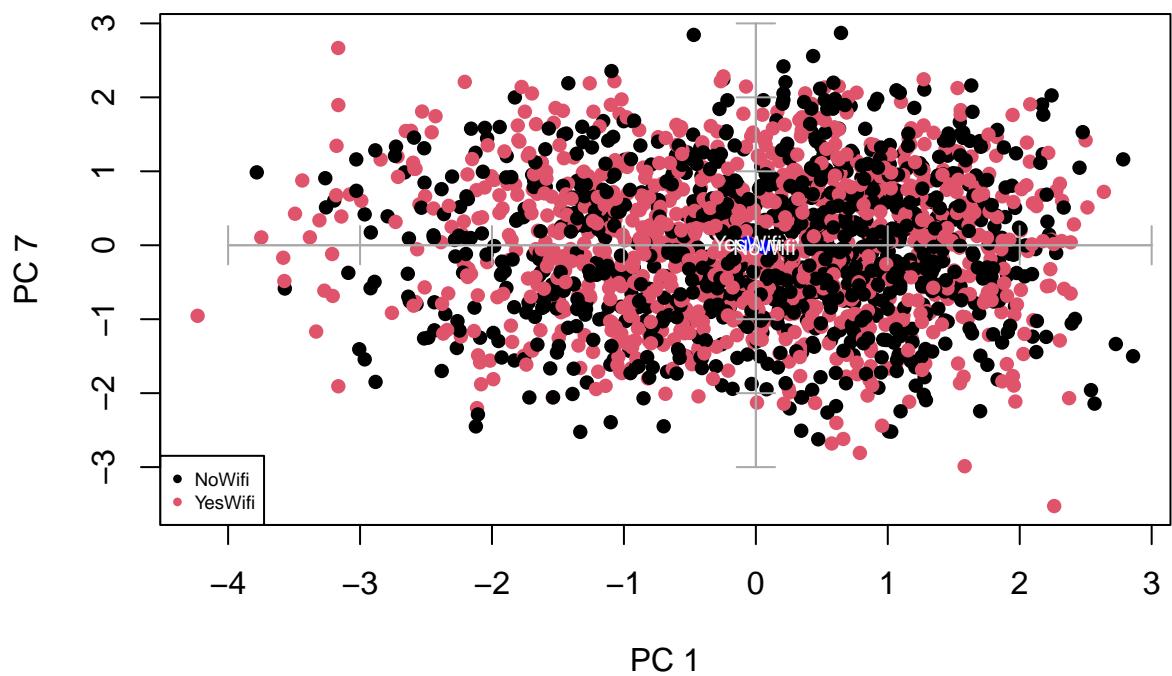
}

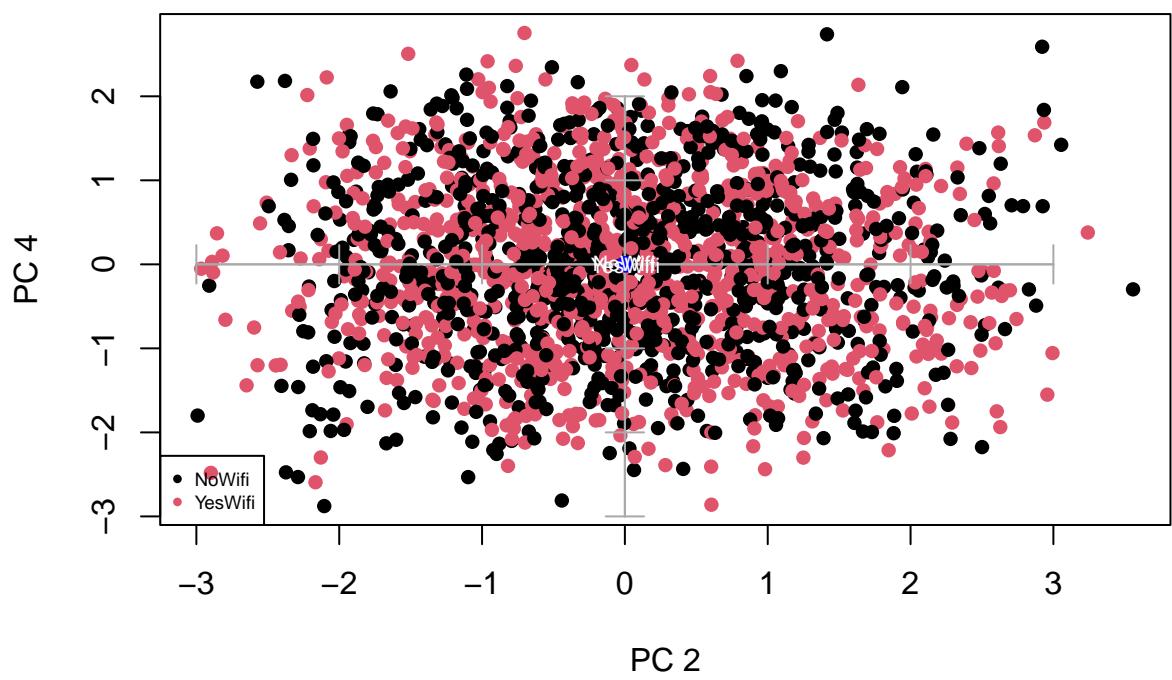
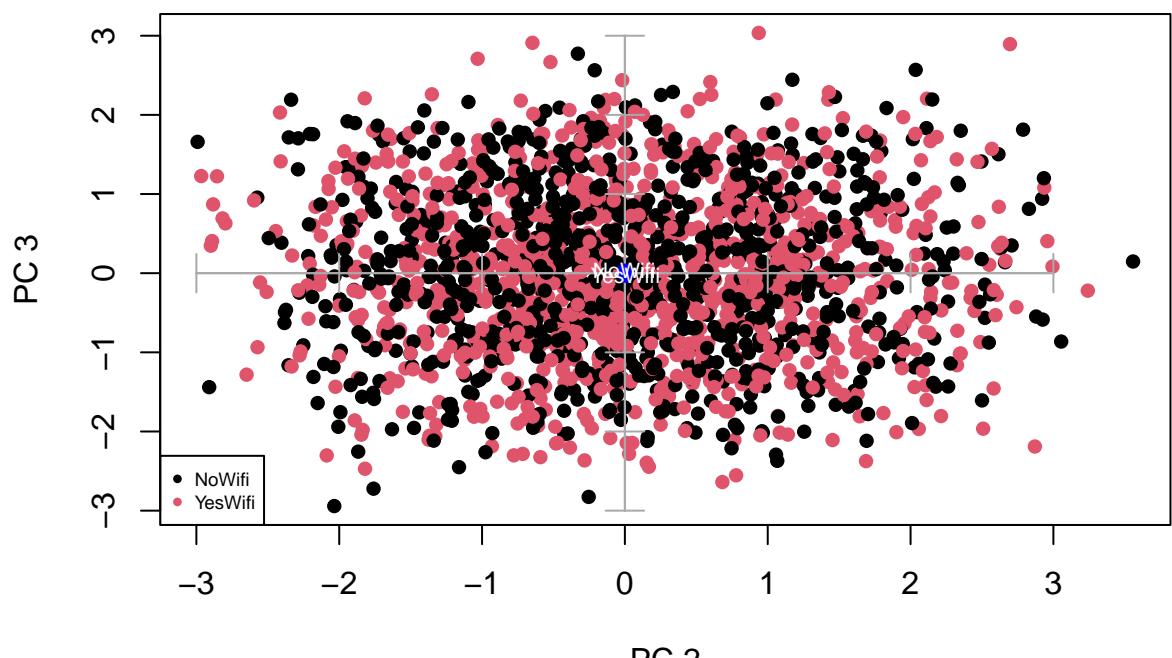
```

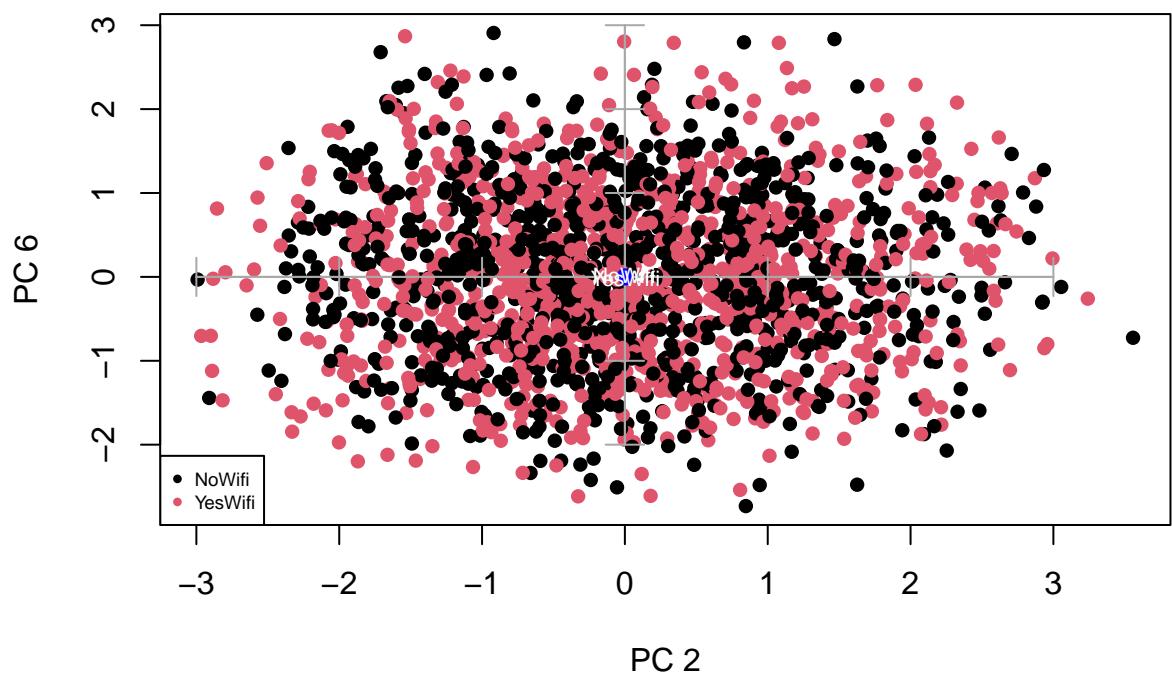
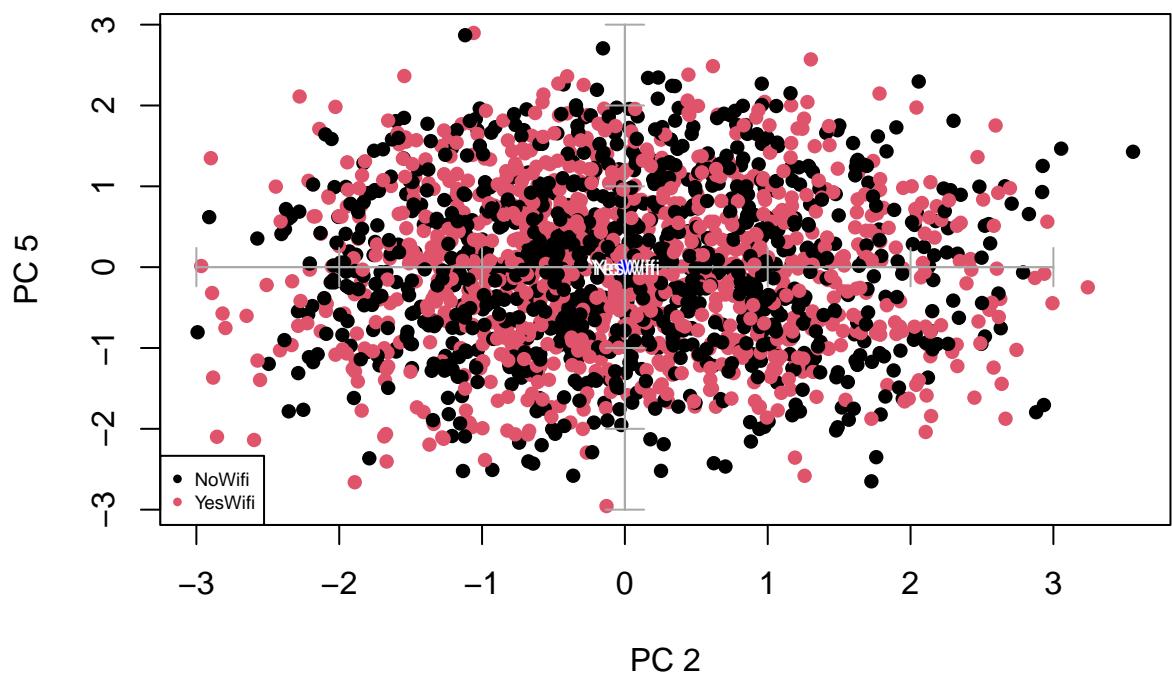


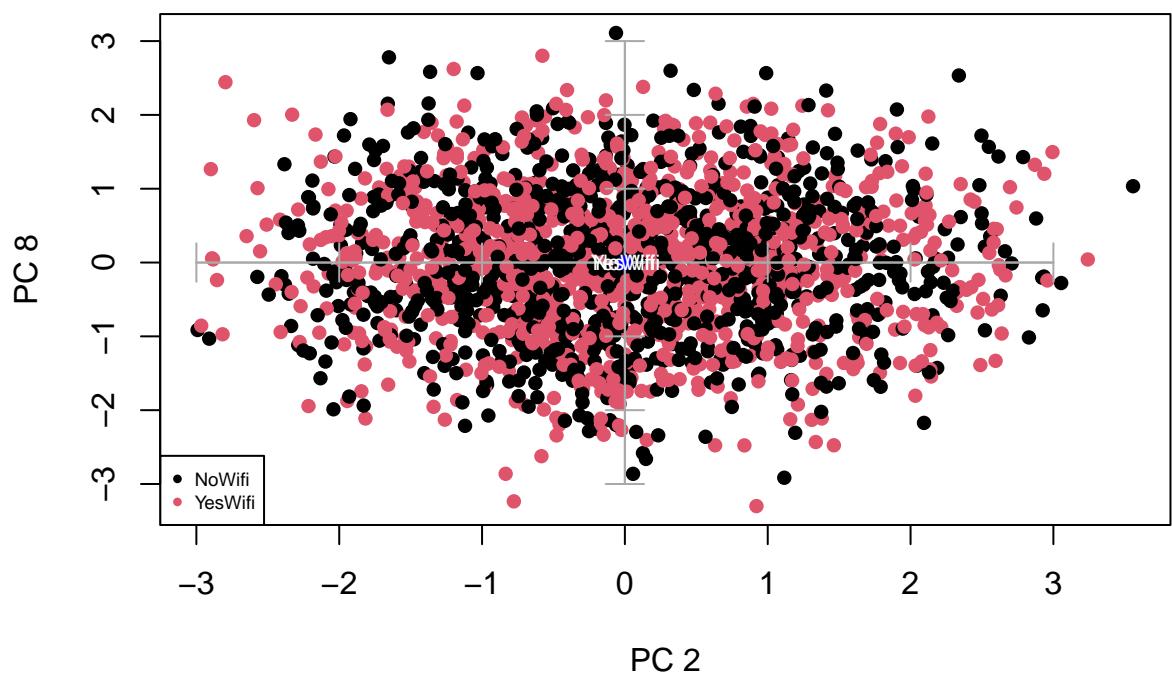
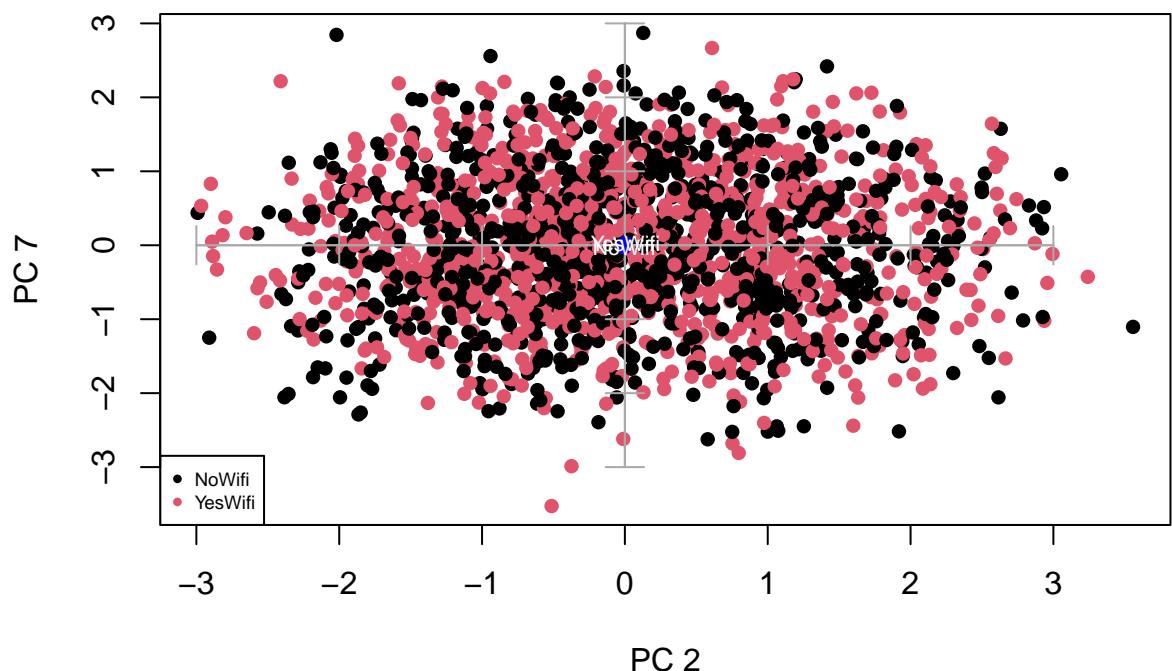


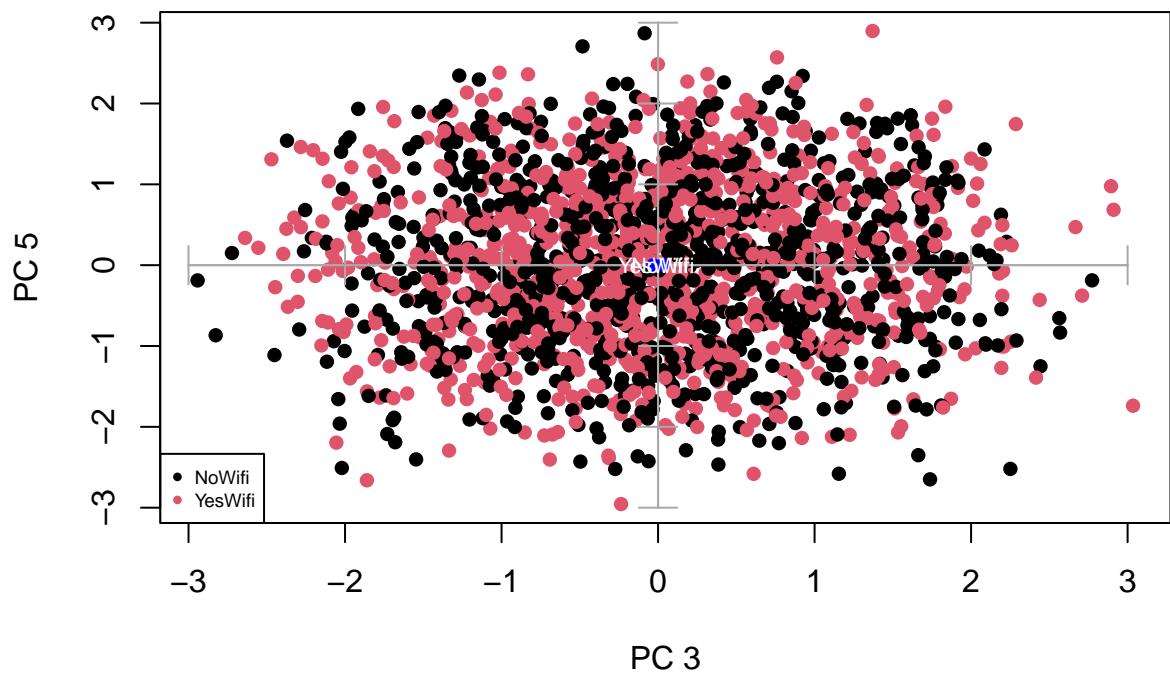
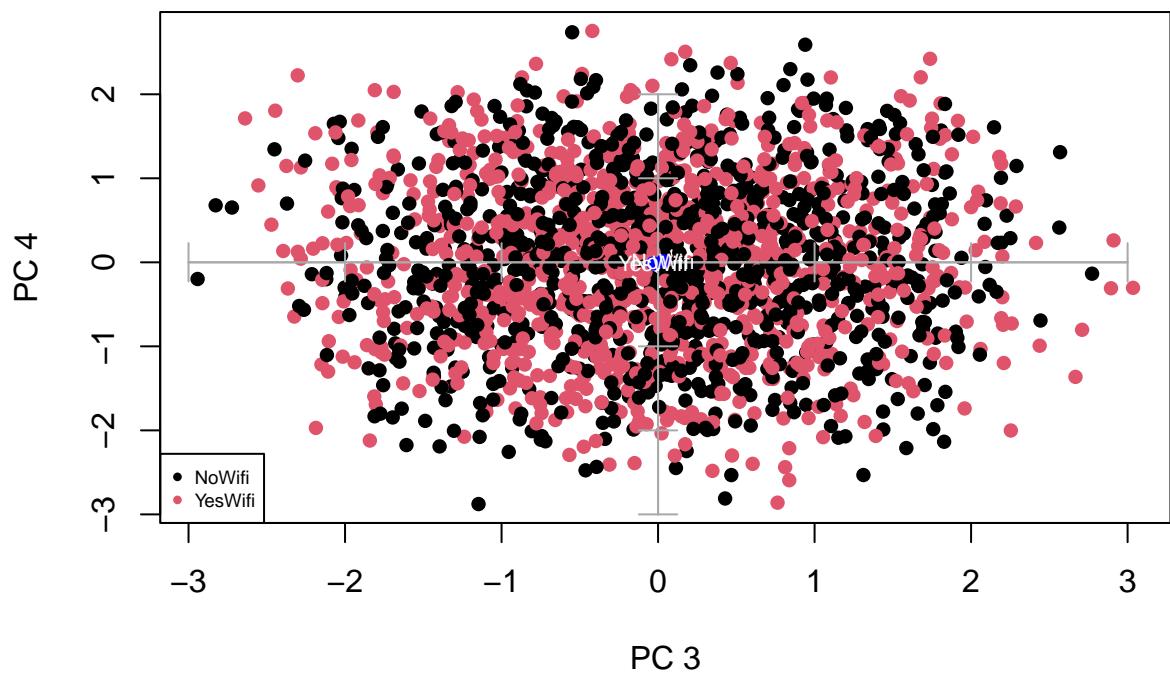


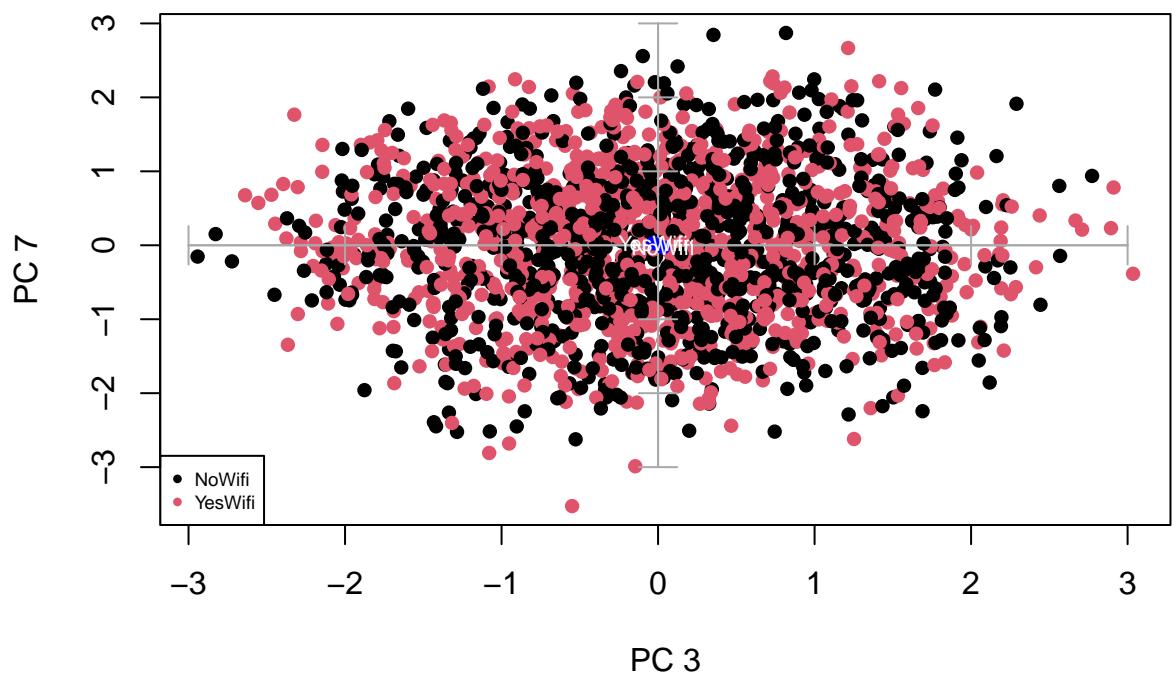
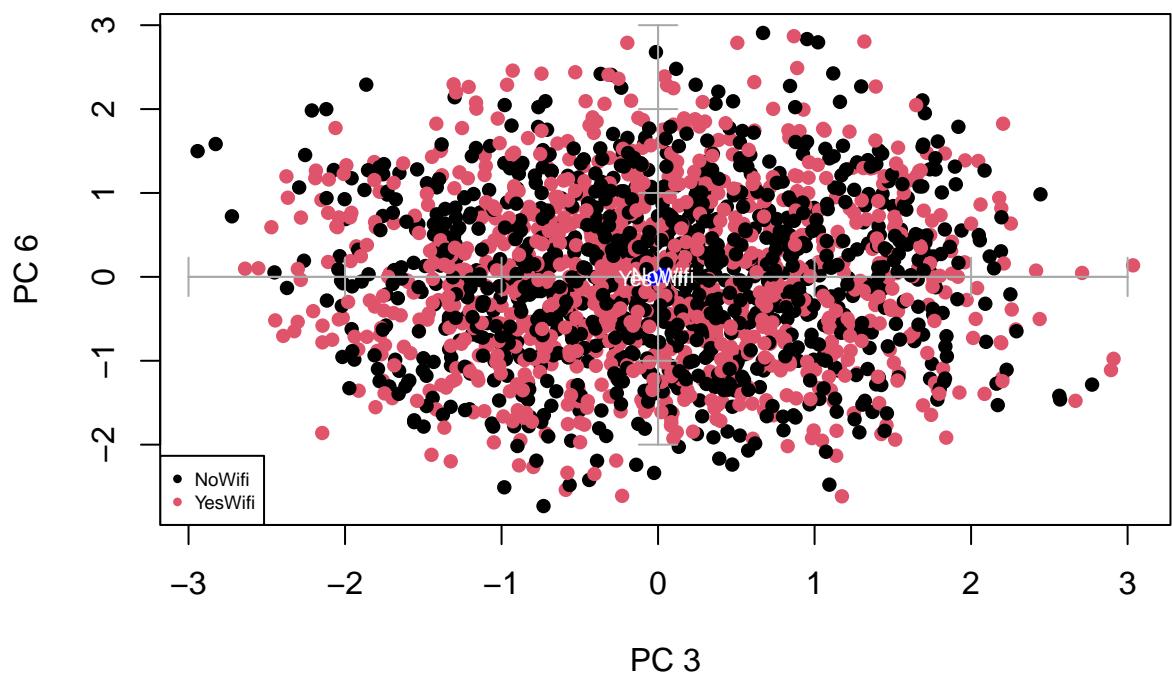


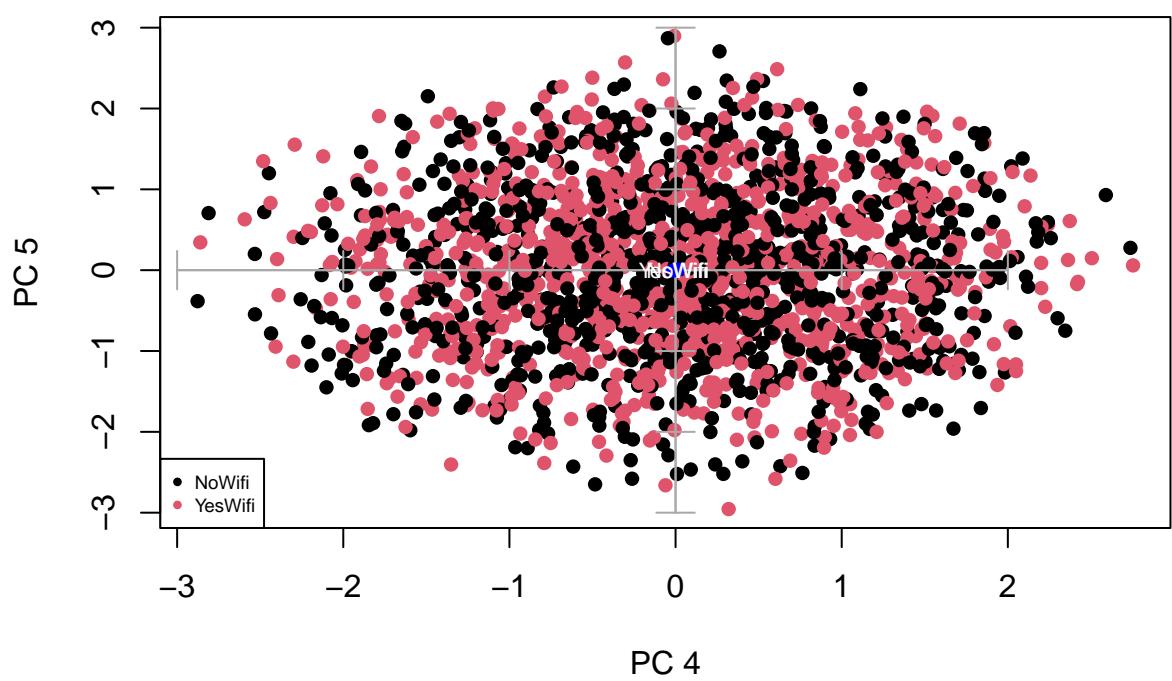
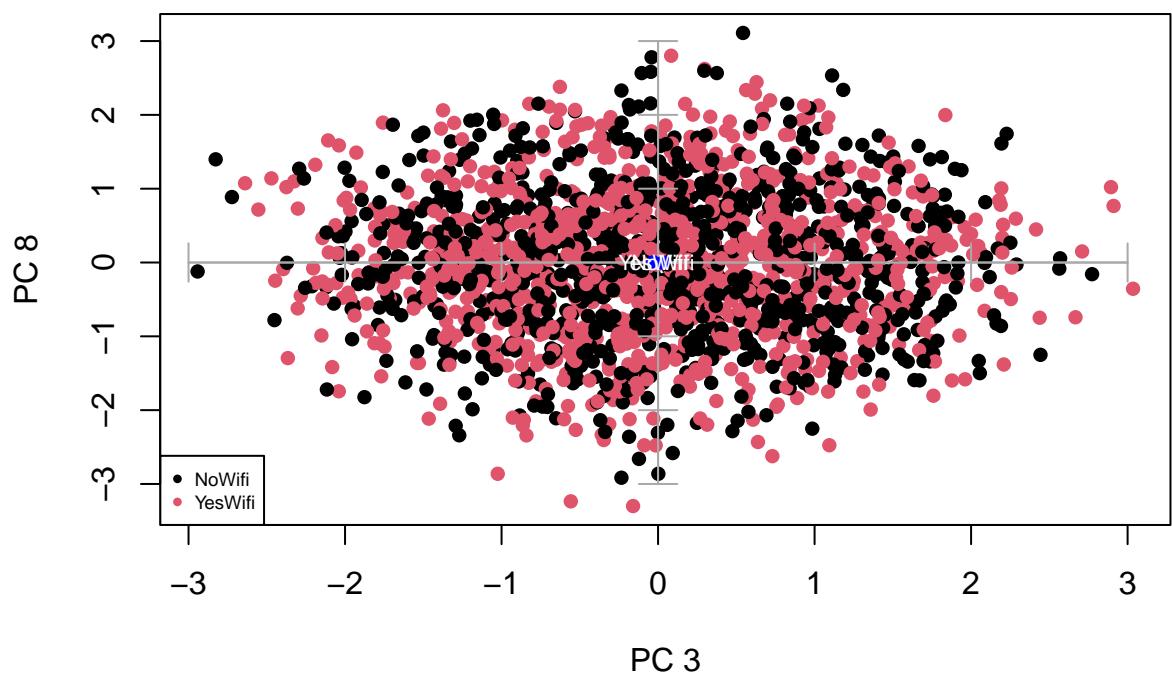


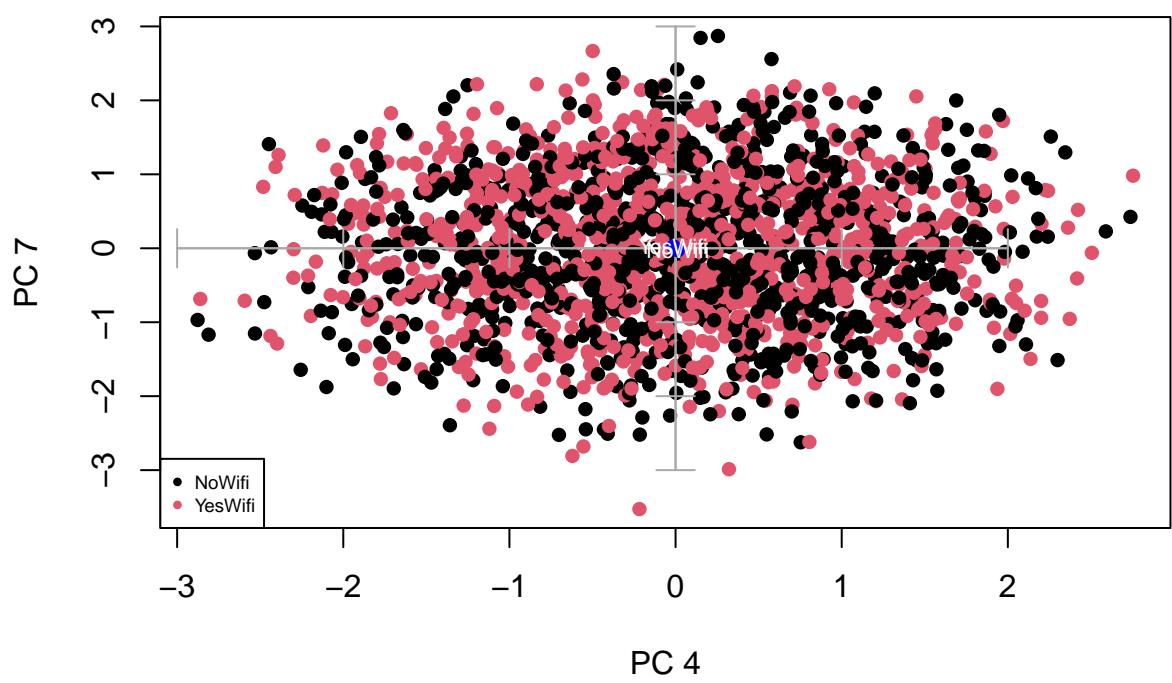
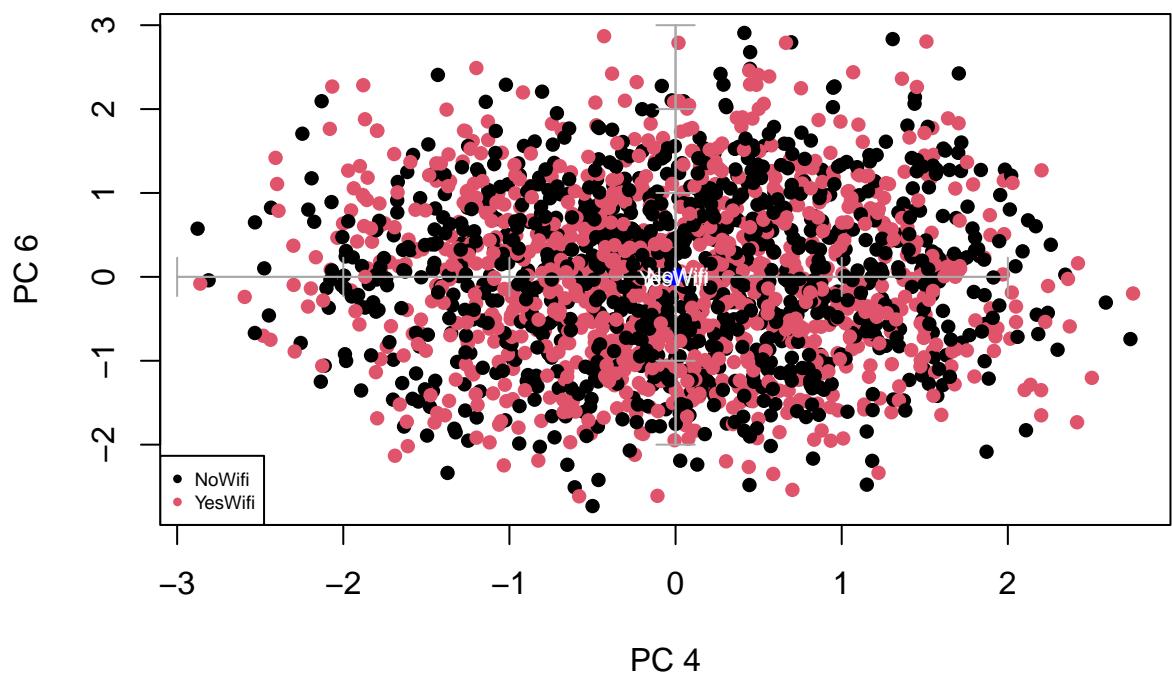


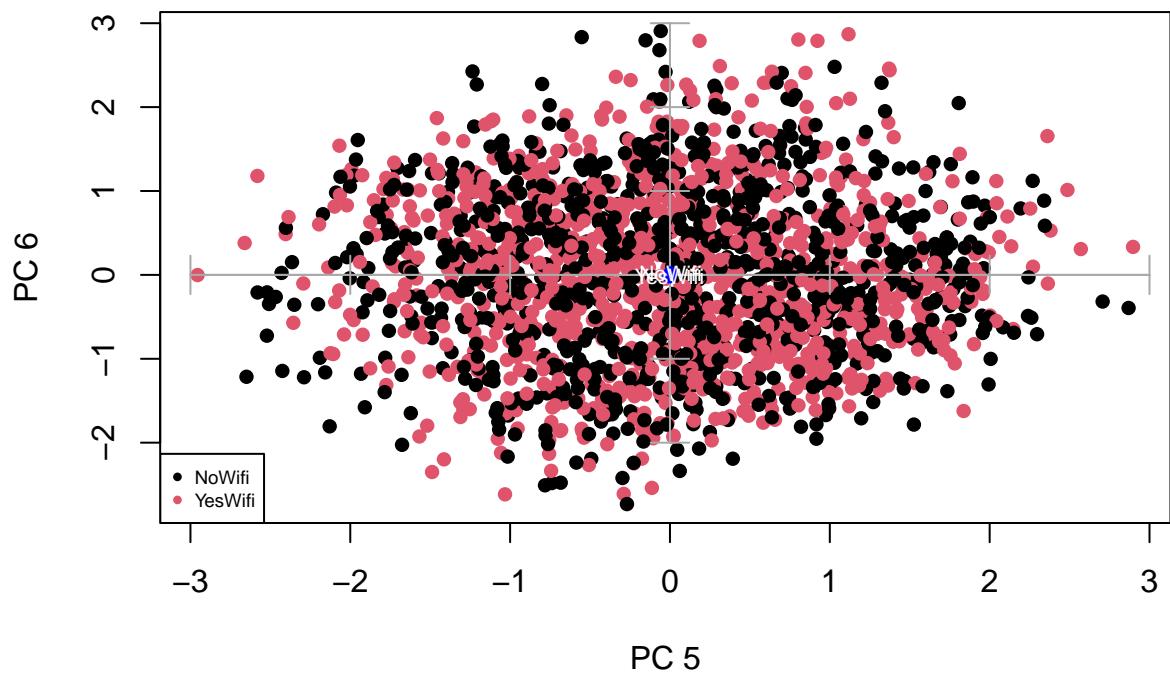
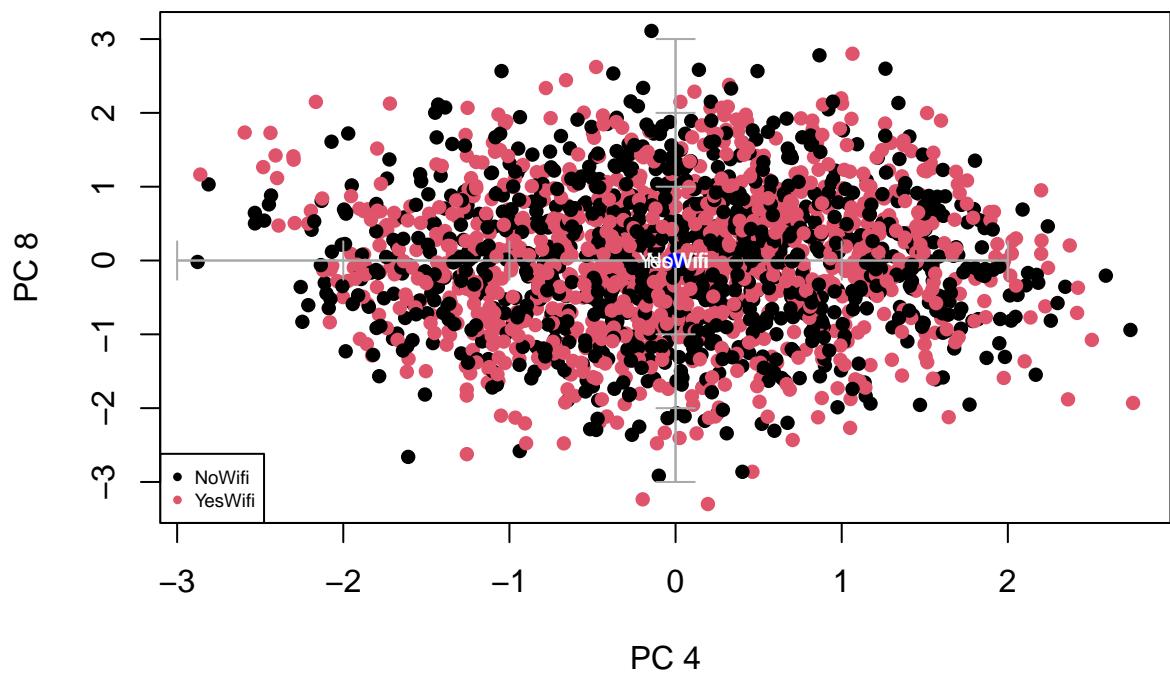


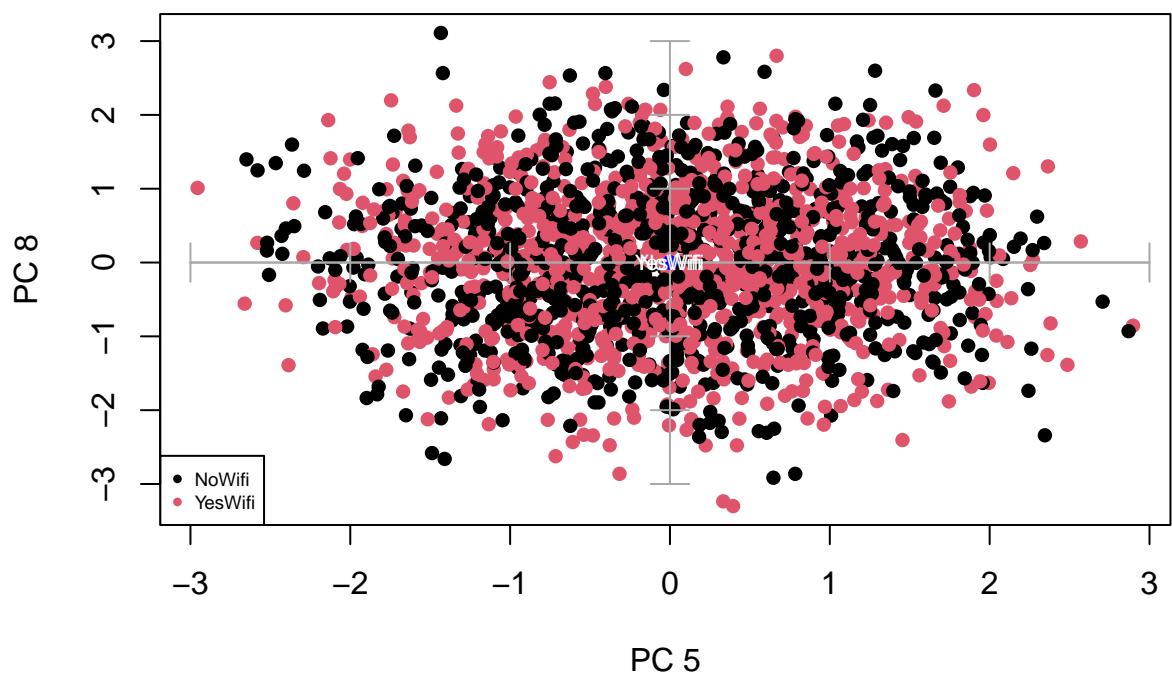
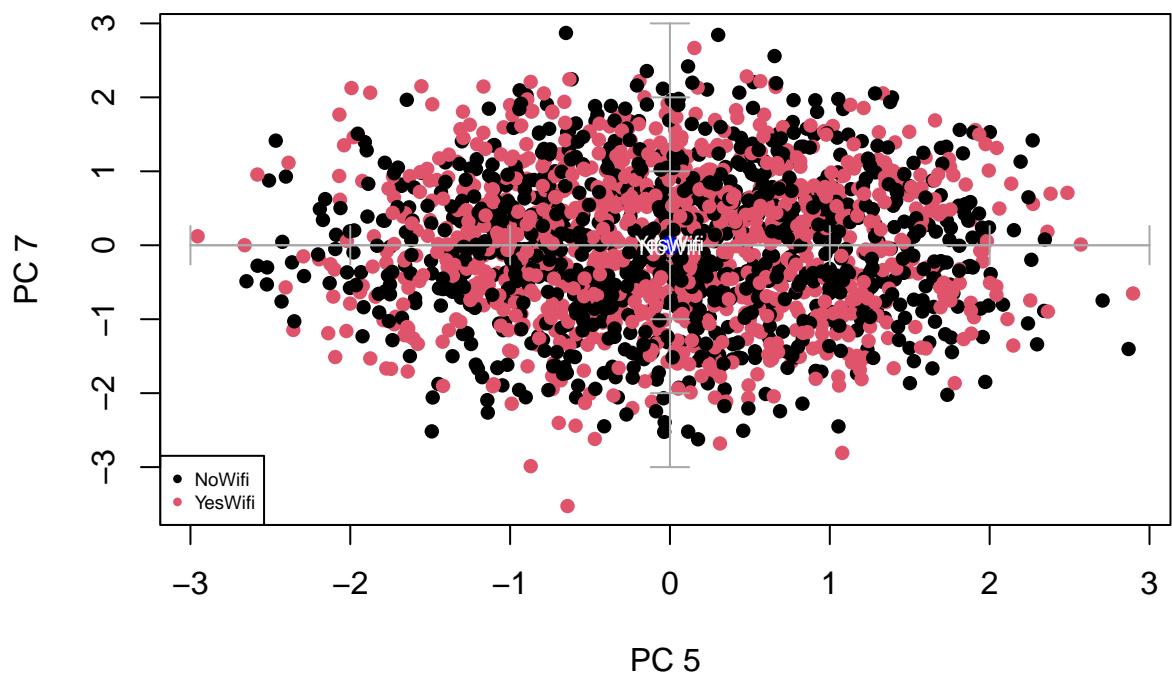


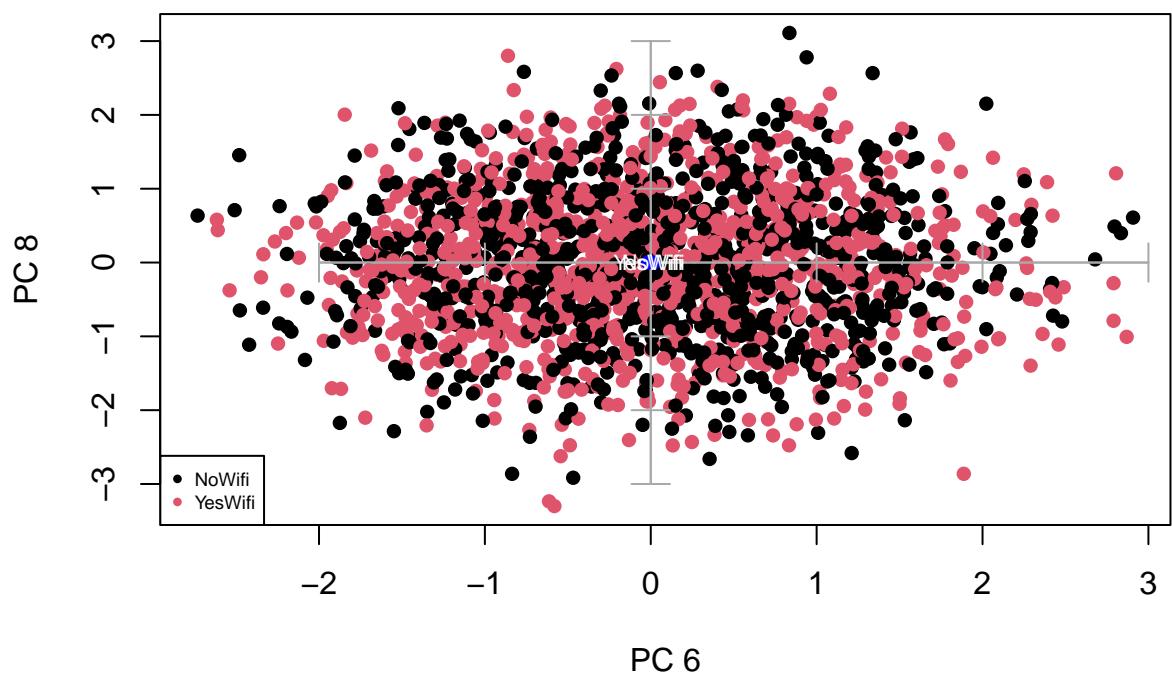
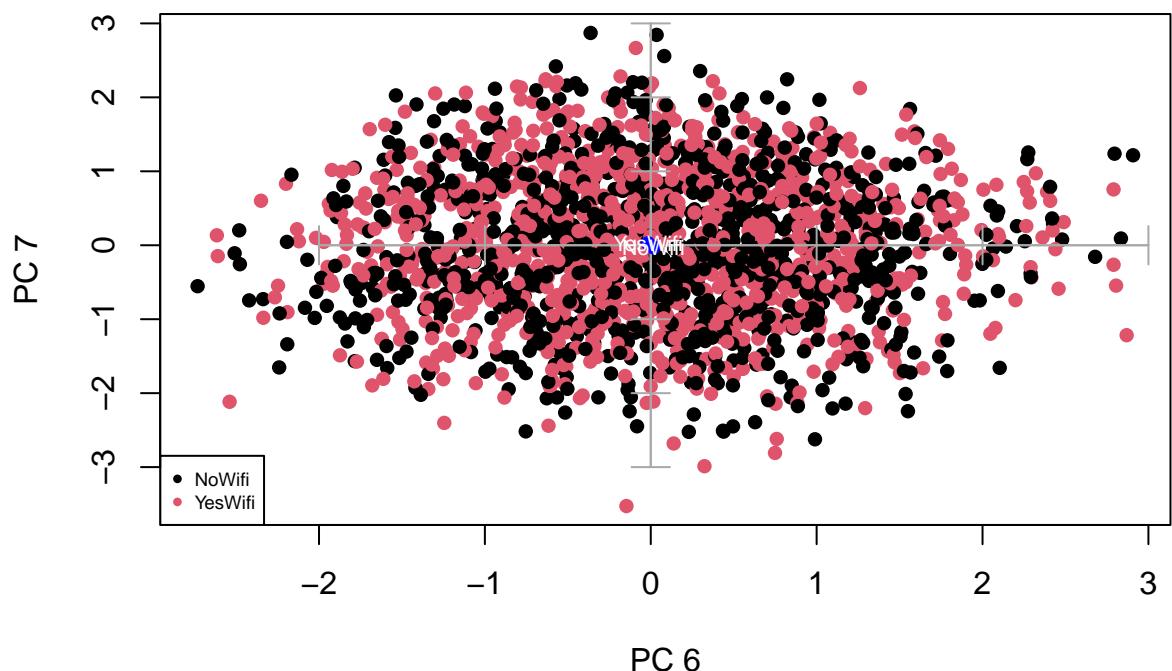


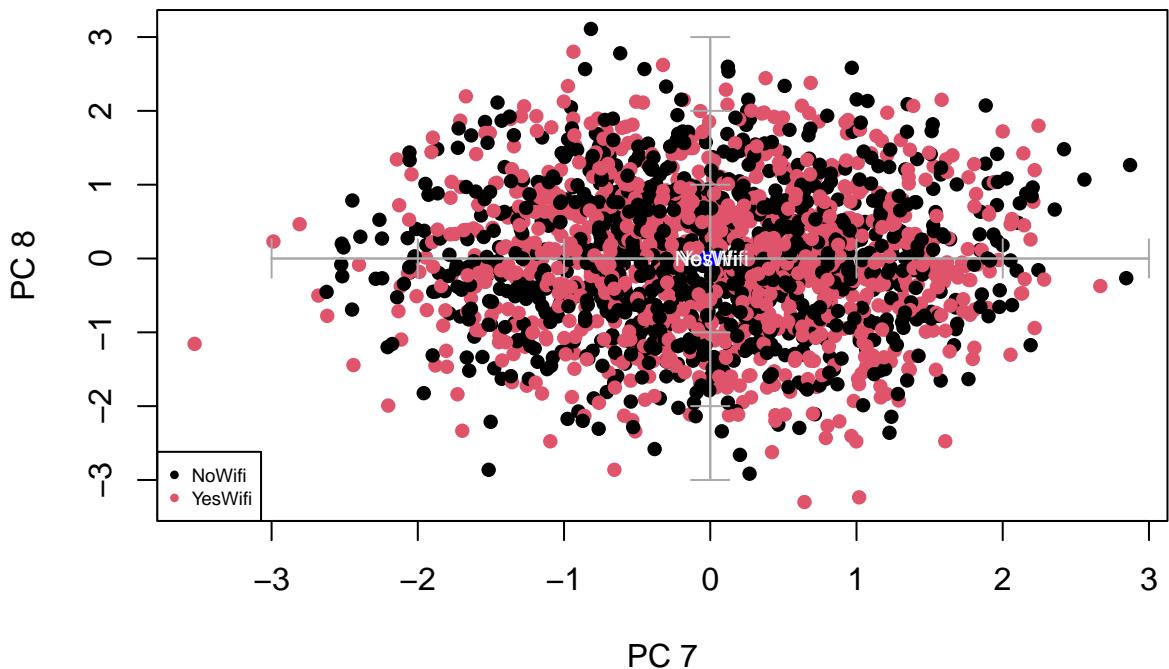








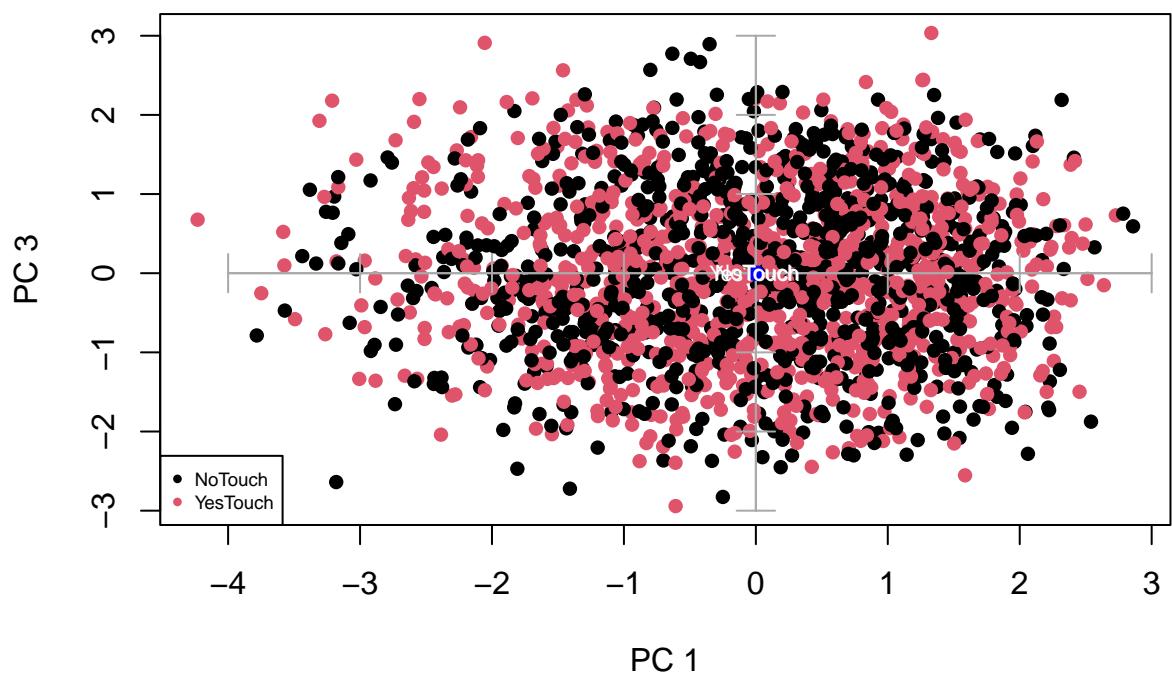
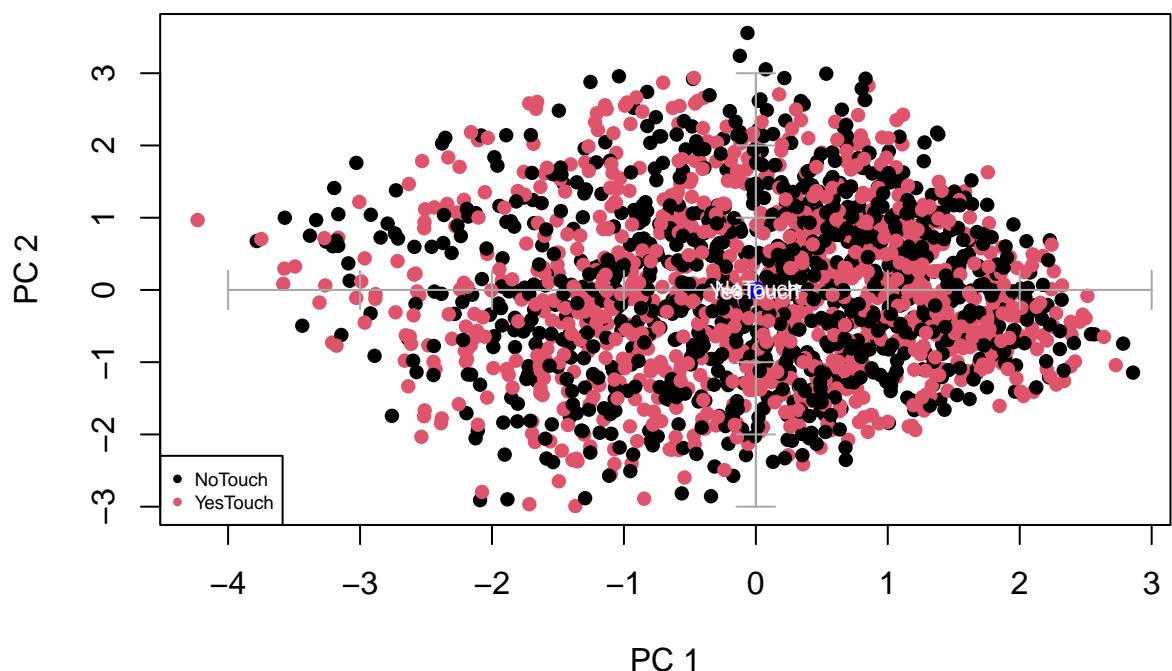


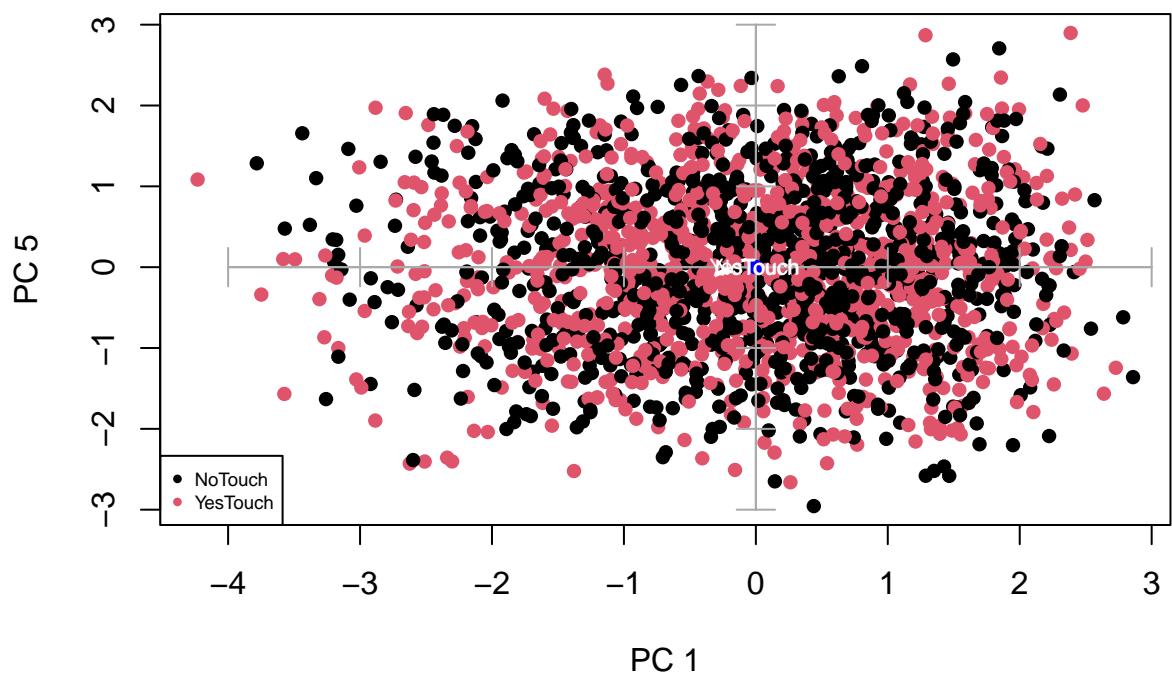
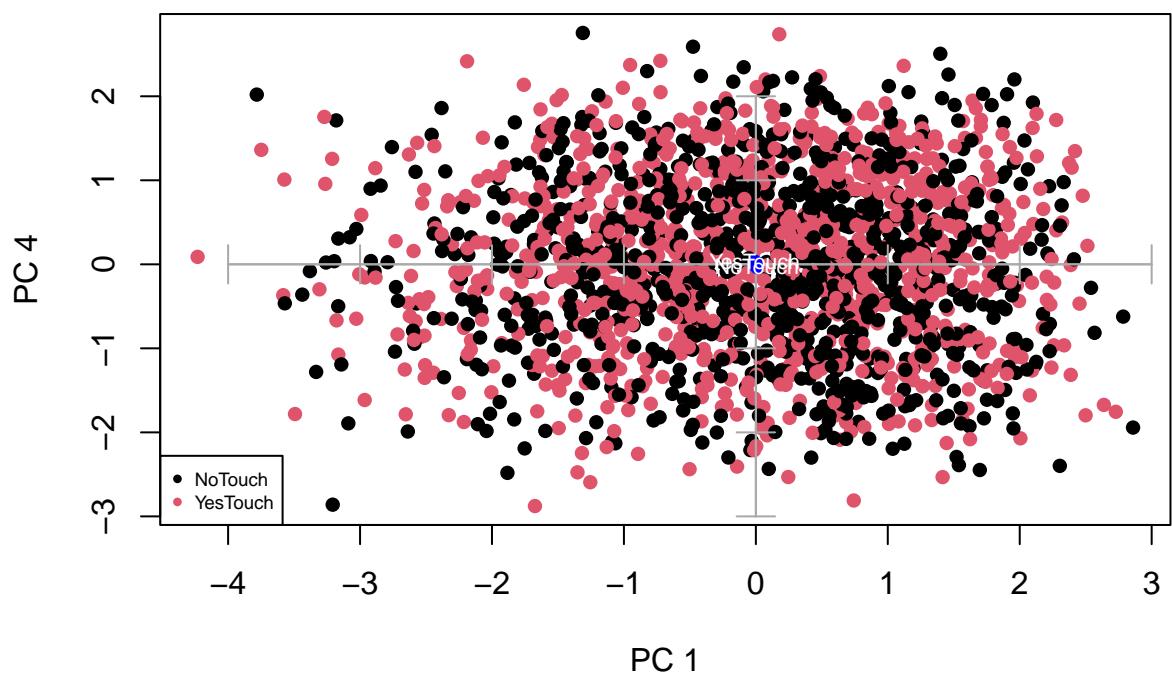


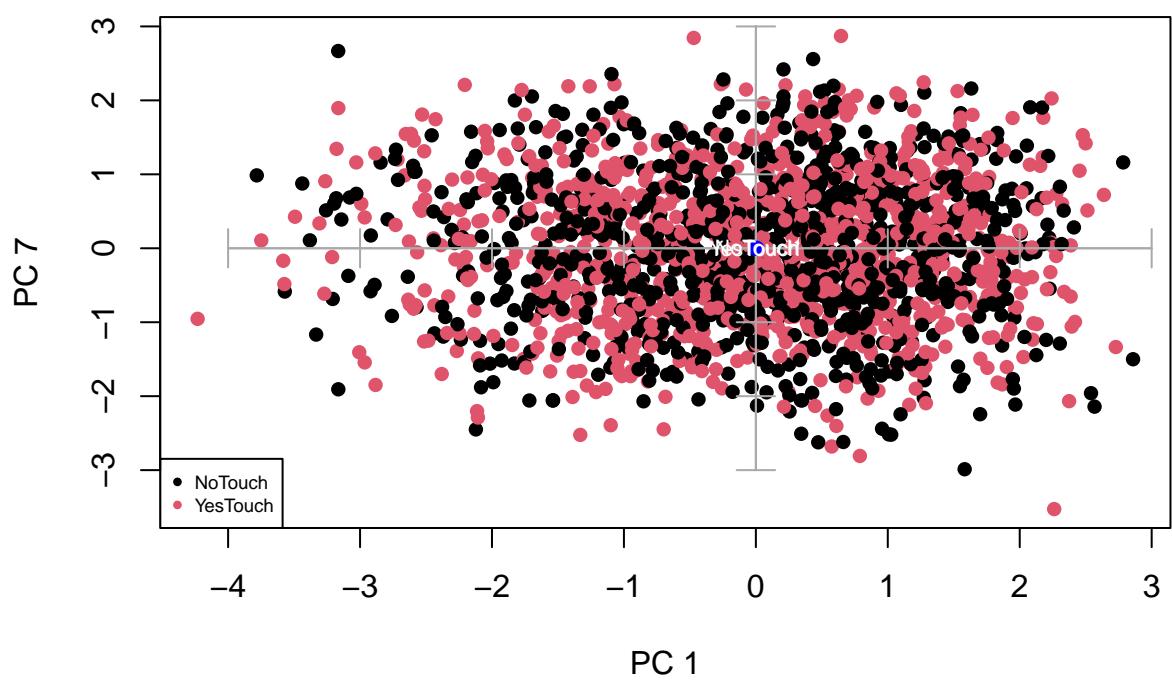
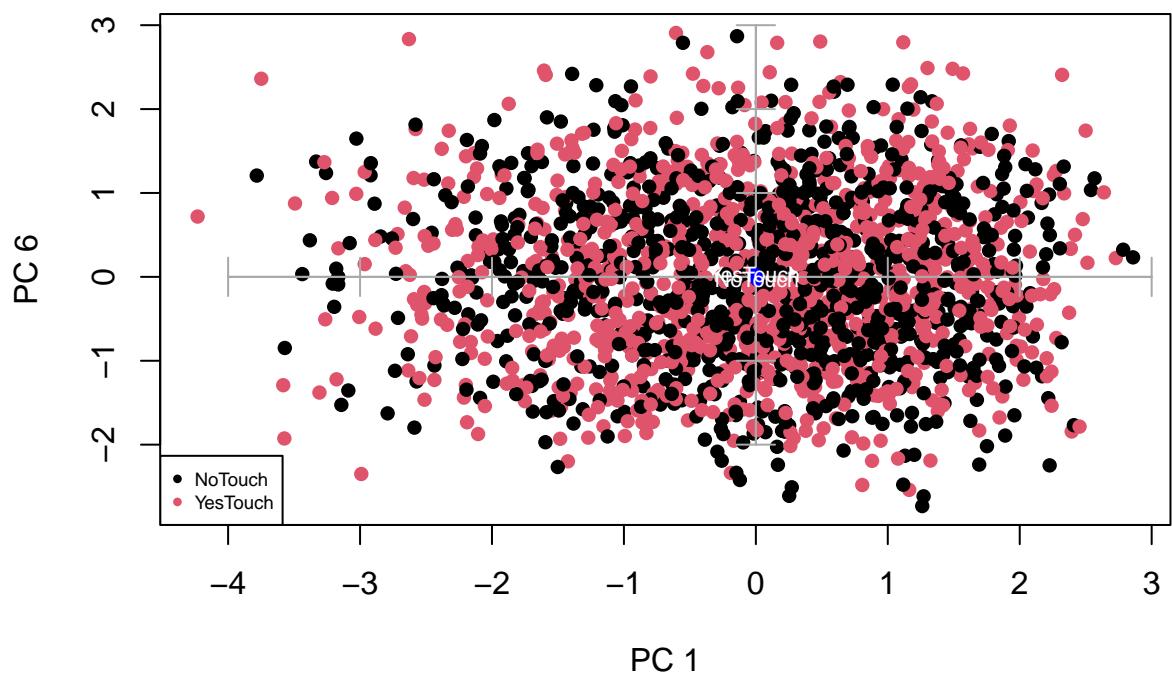
```

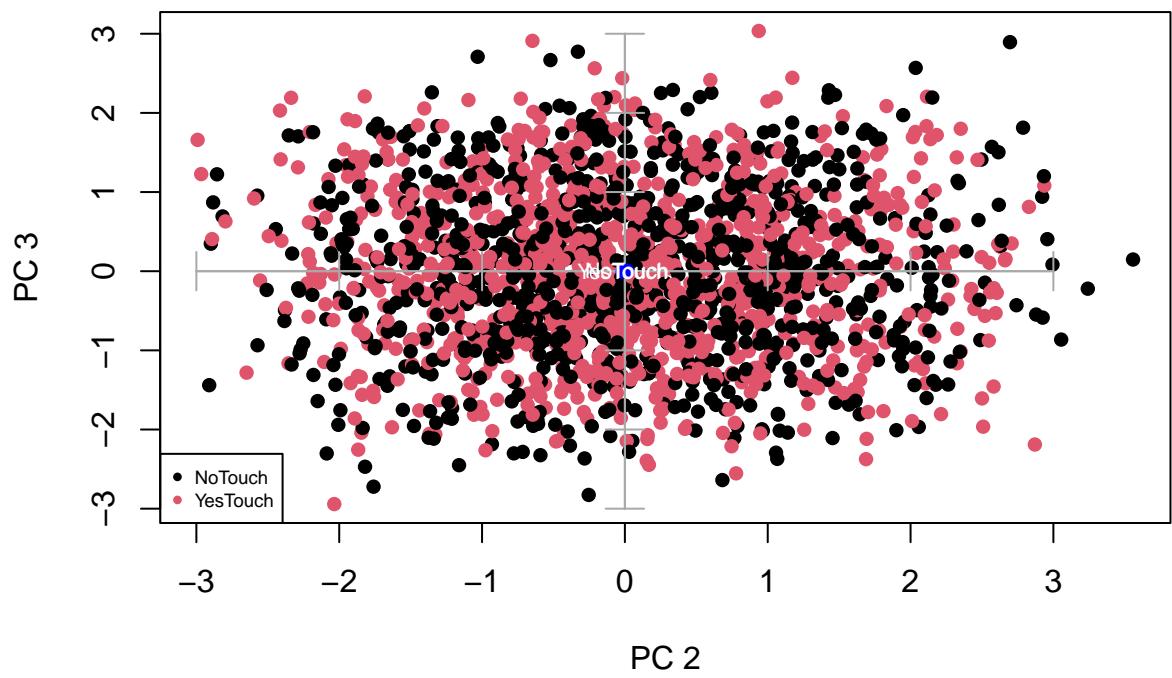
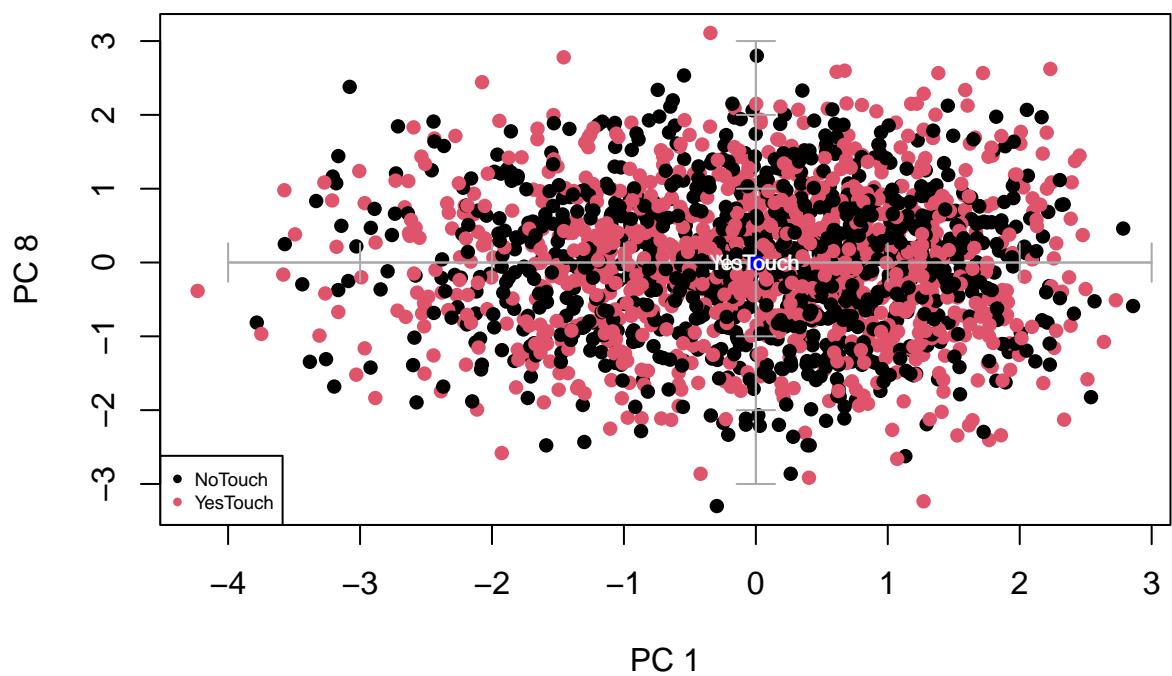
for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$touch_screen
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab= paste("PC",toString(i)) , ylab= paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
    legend("bottomleft",levels(varcat),pch=16,col=c(1:2), cex=0.6)

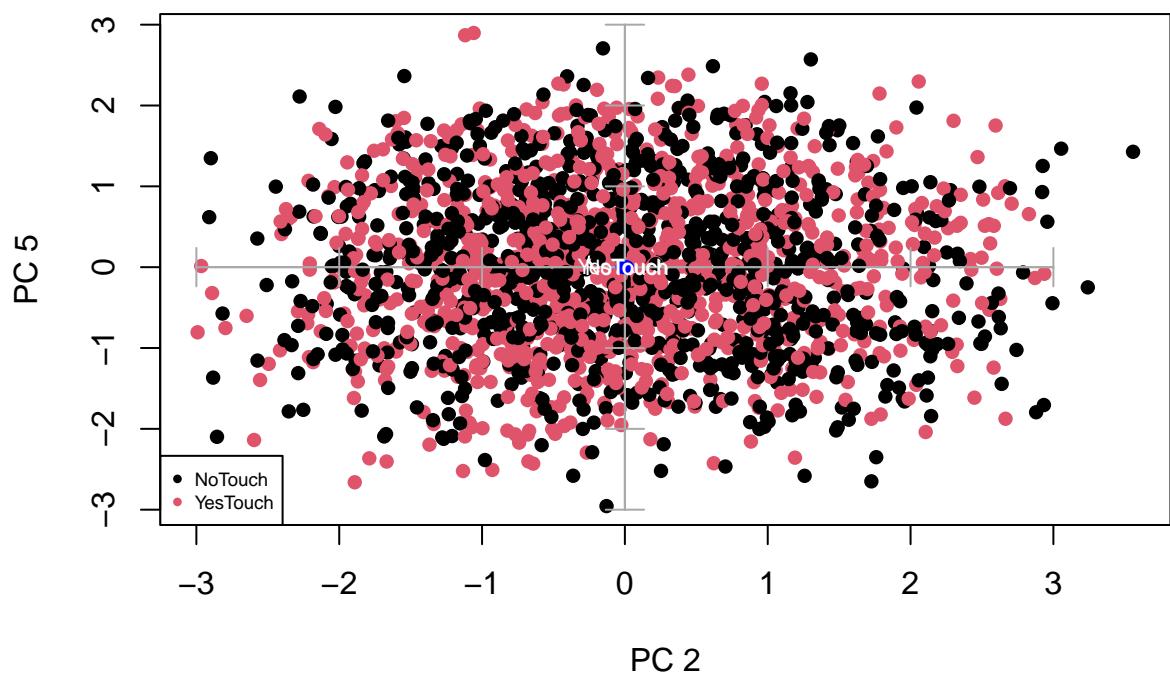
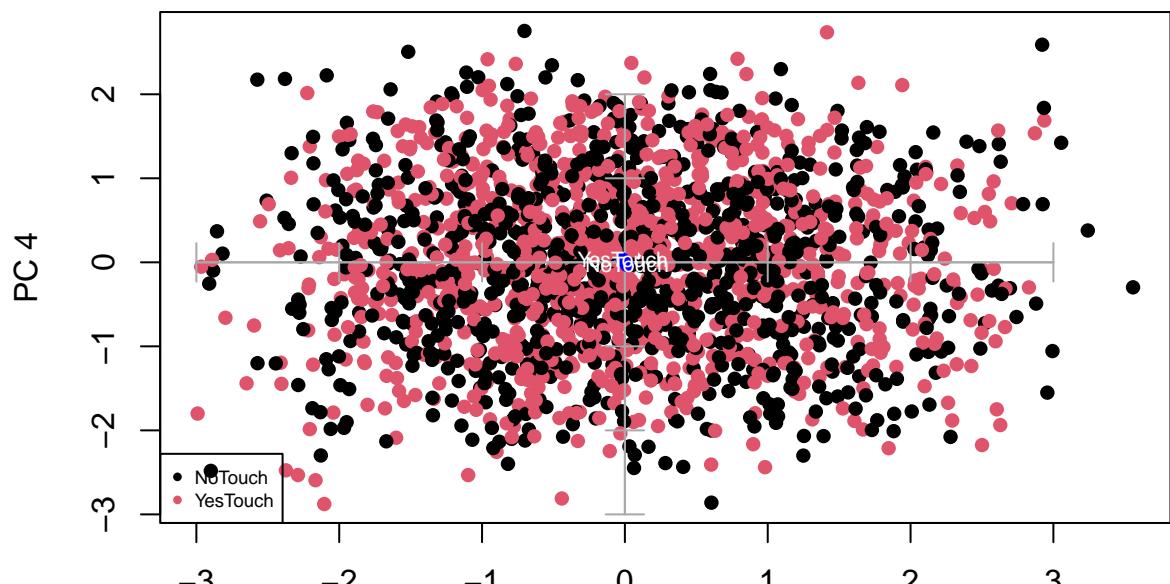
    #select your qualitative variable
    fdic1 = tapply(Psi[,i],varcat,mean)
    fdic2 = tapply(Psi[,j],varcat,mean)
    points(fdic1,fdic2,pch=16,col="blue")
    text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
  }
}
  
```

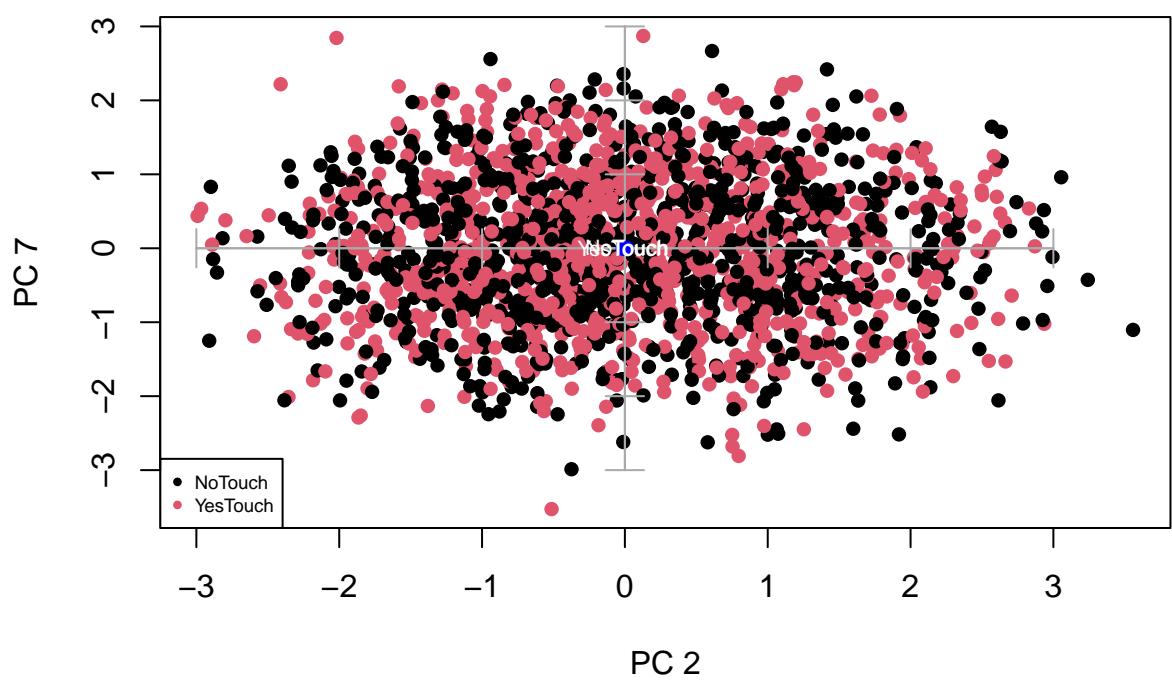
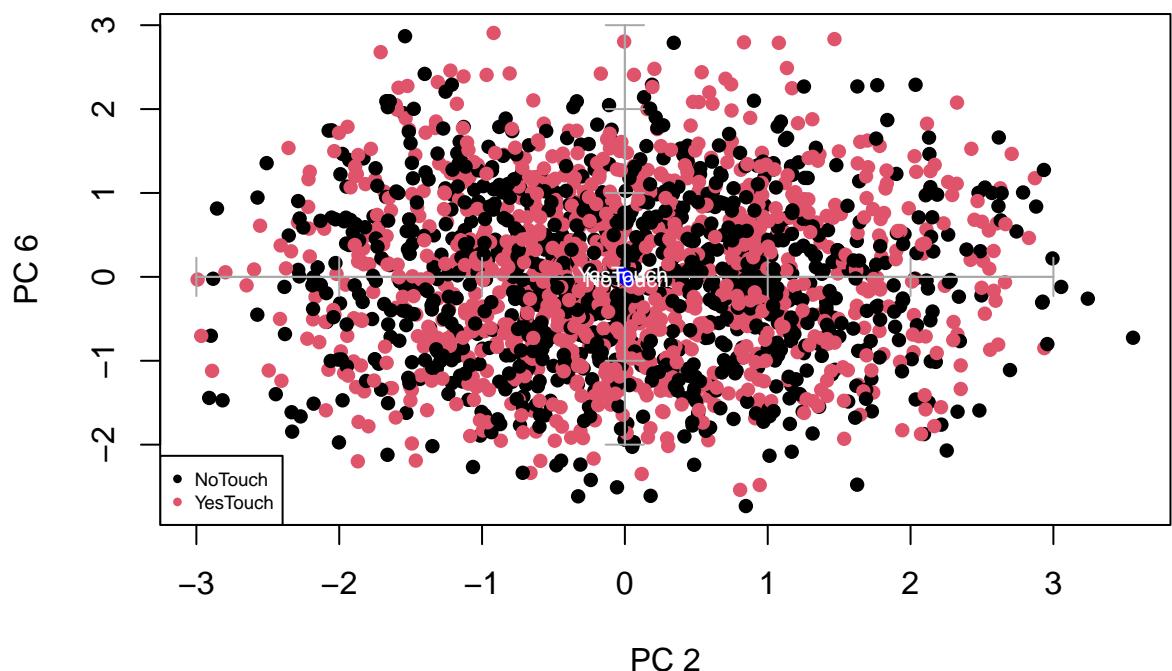


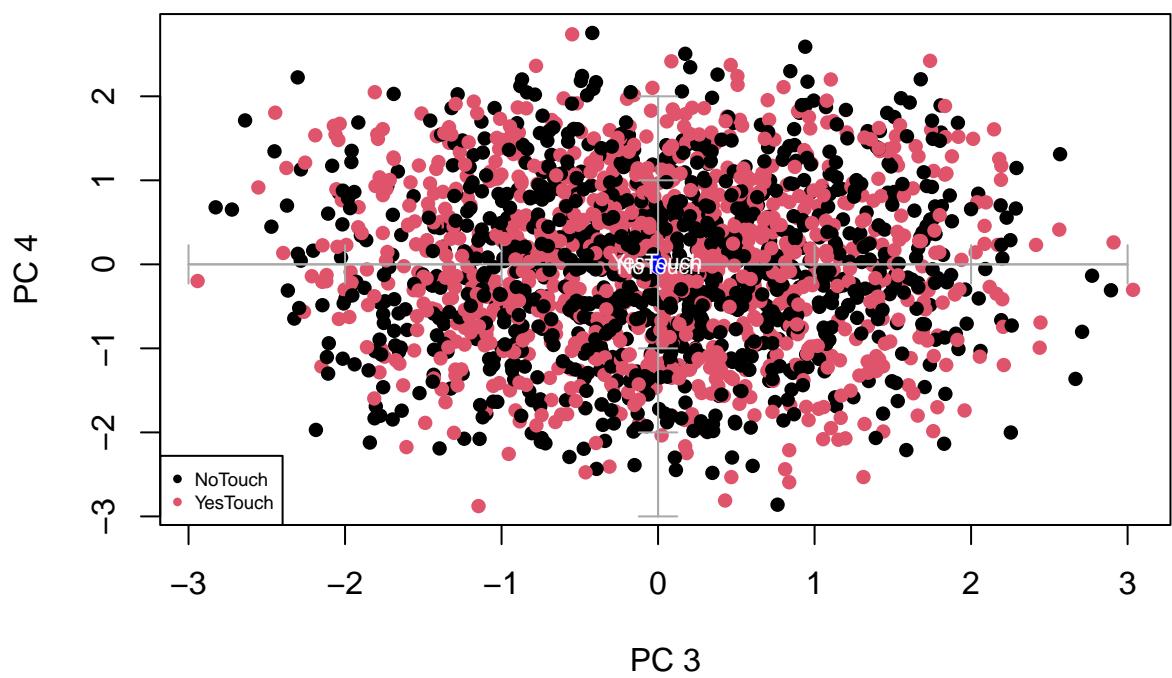
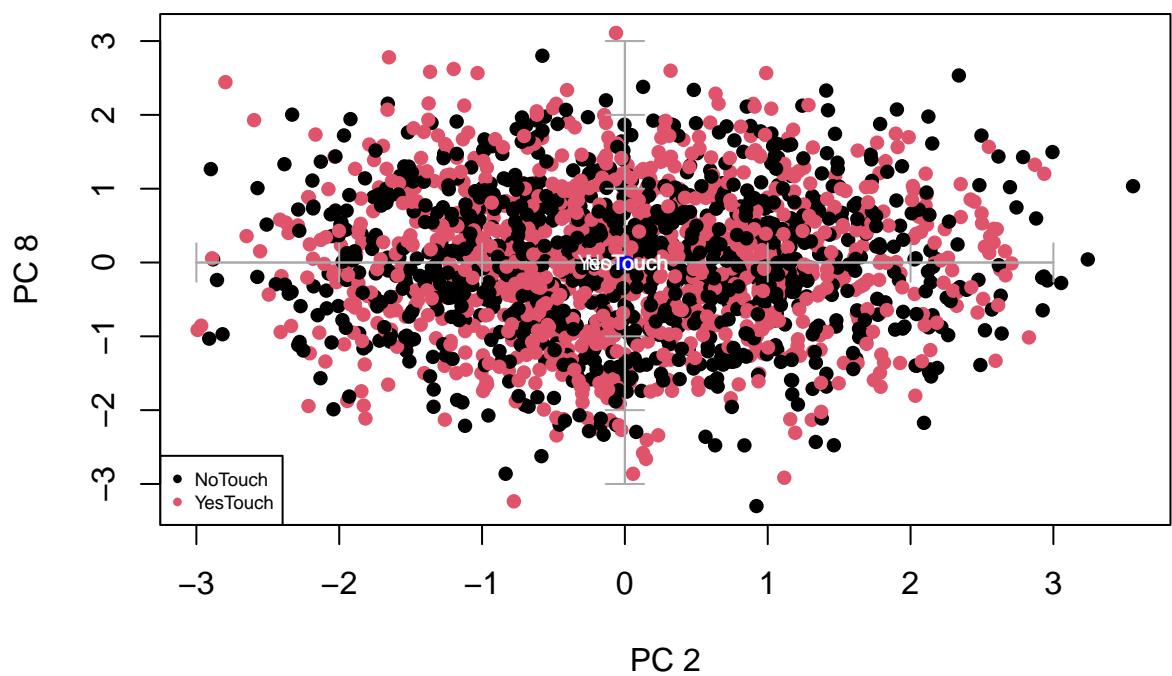


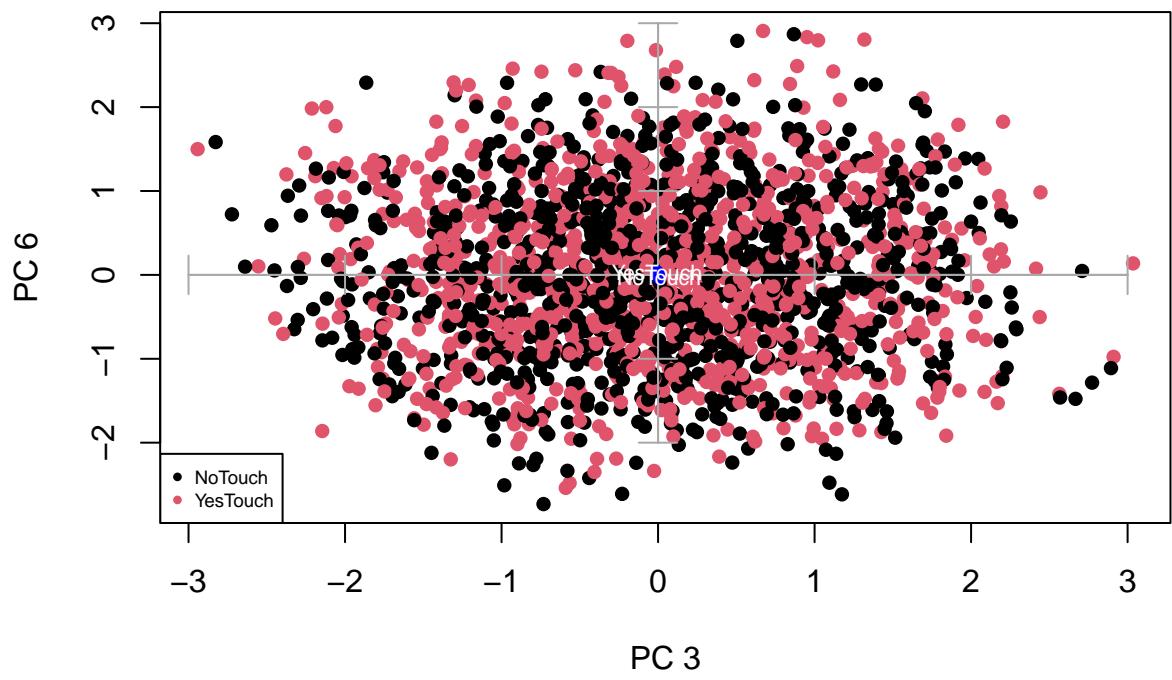
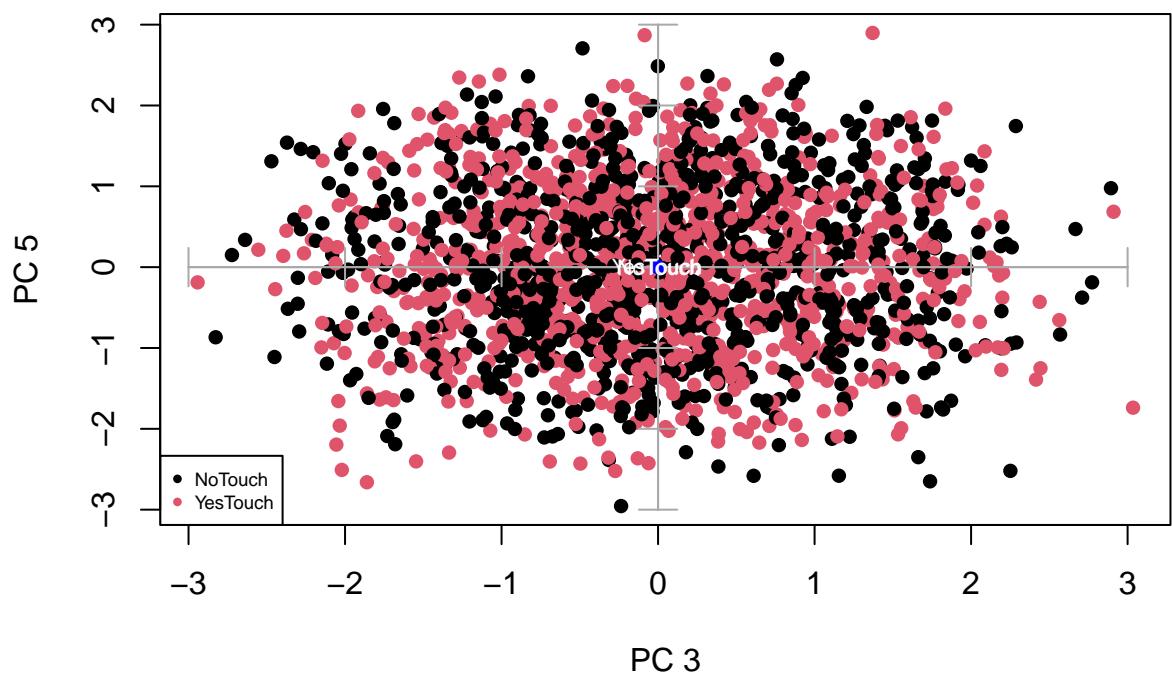


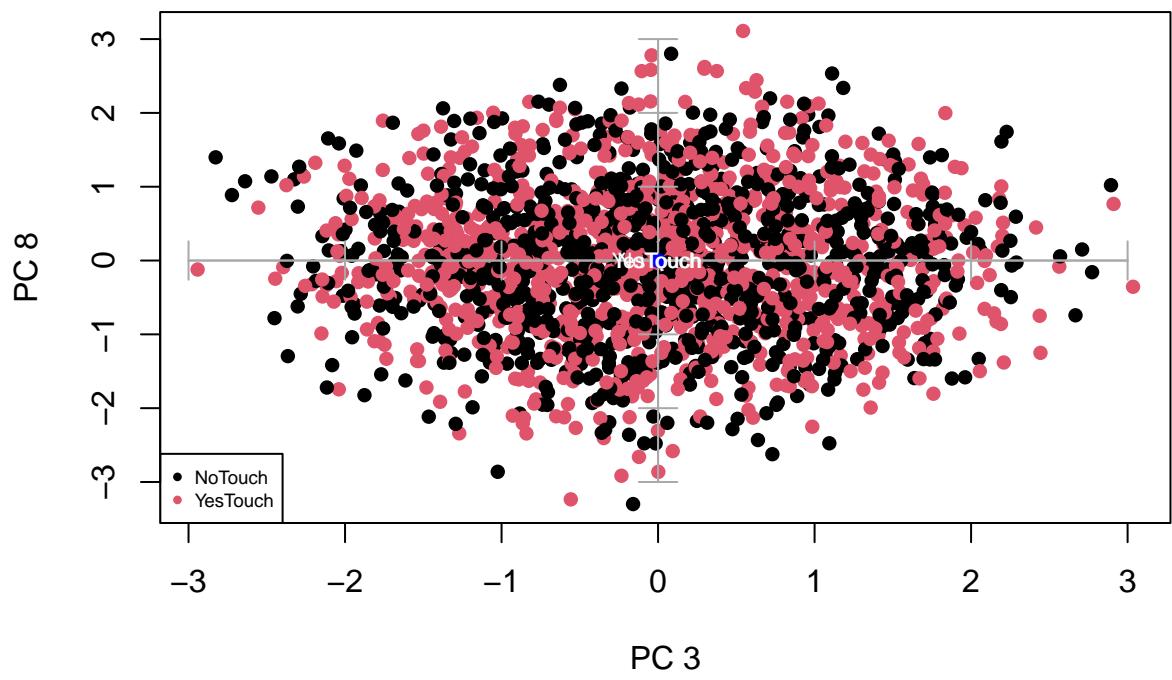
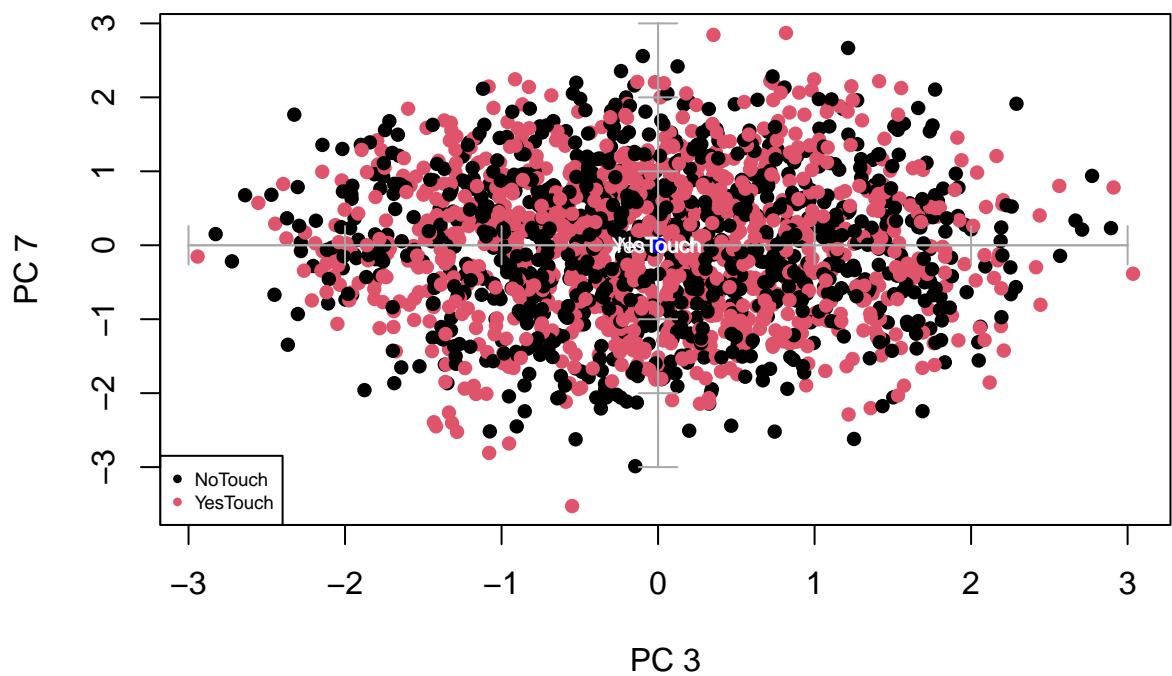


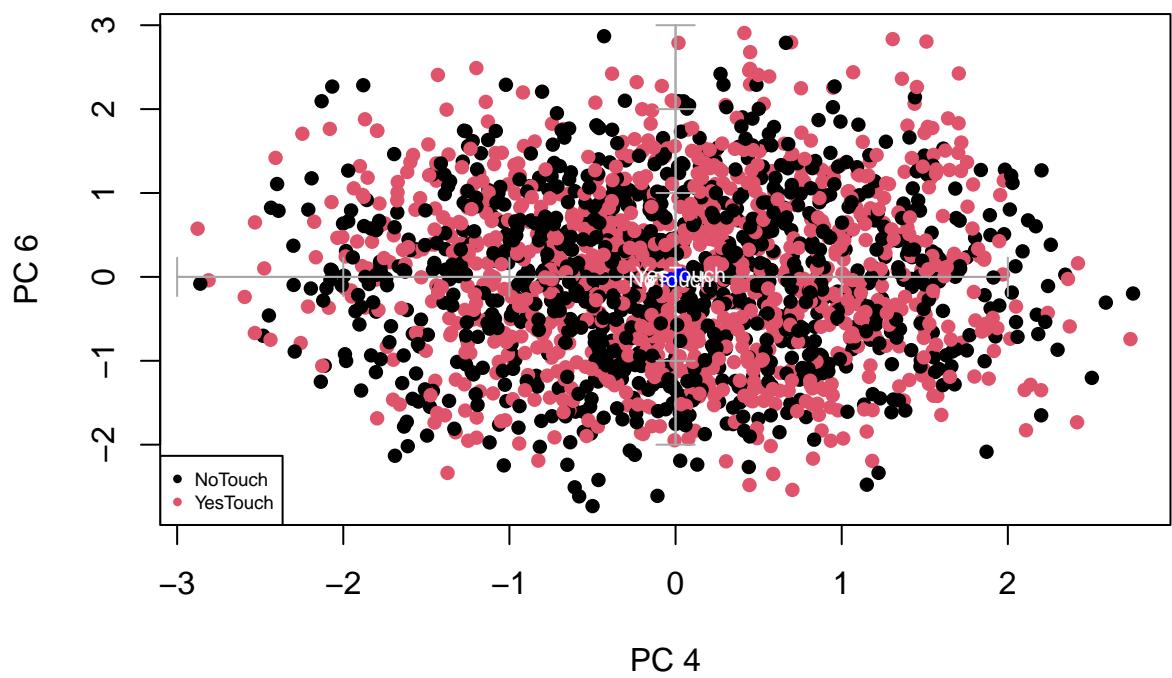
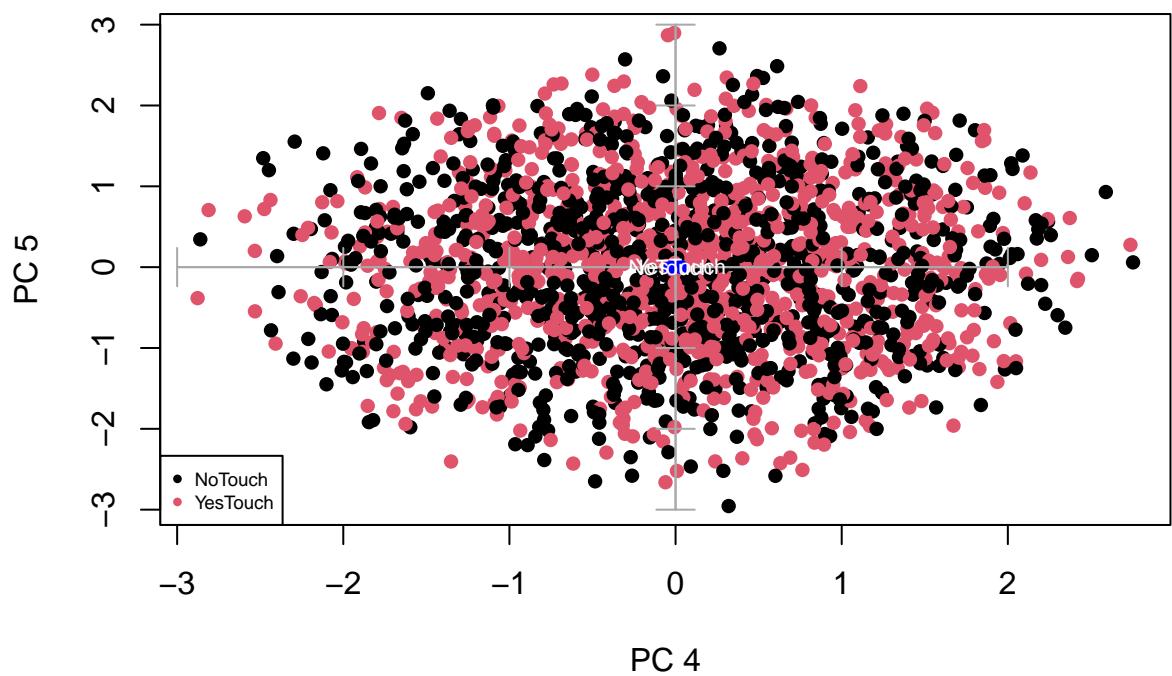


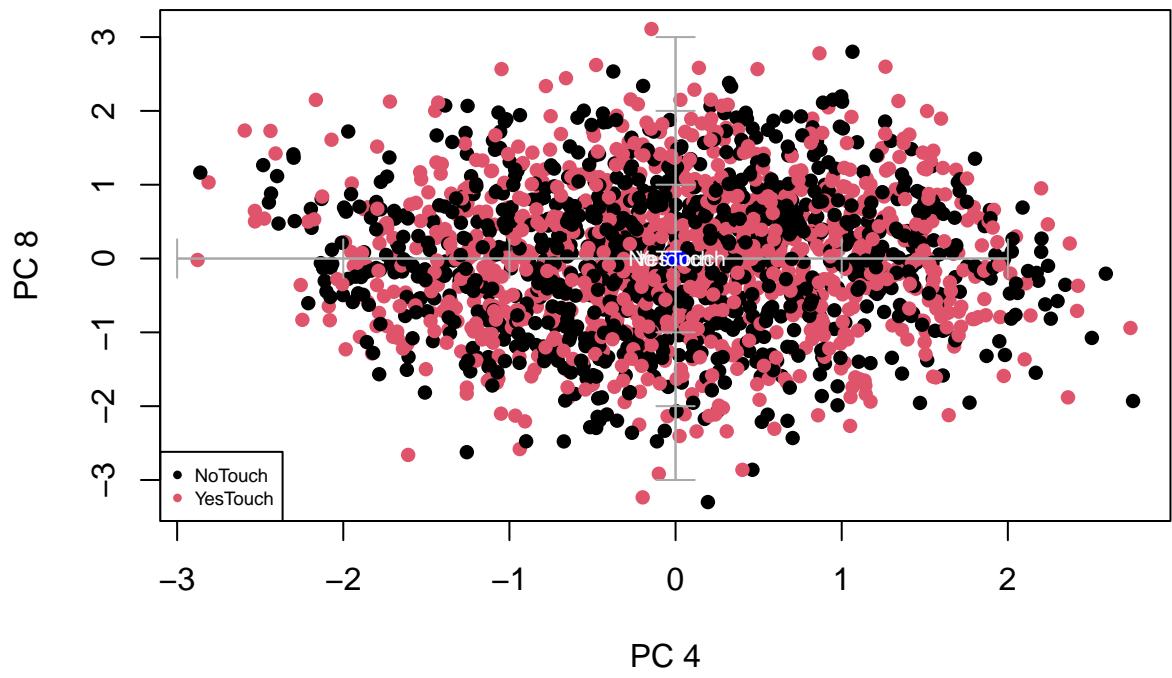
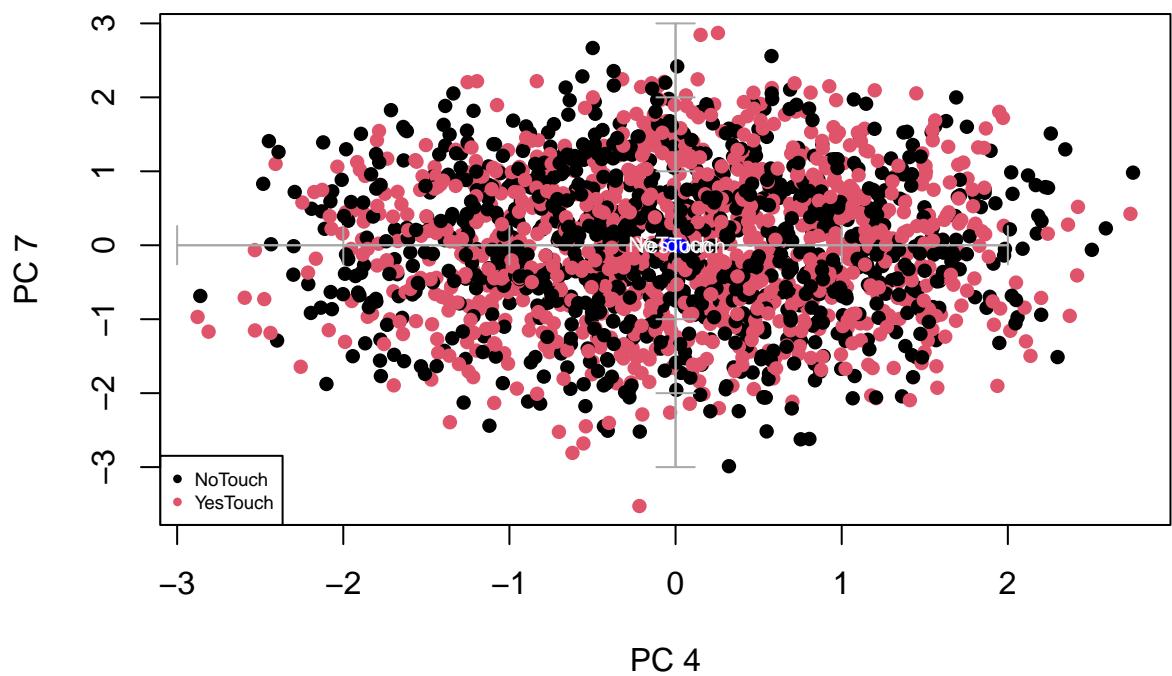


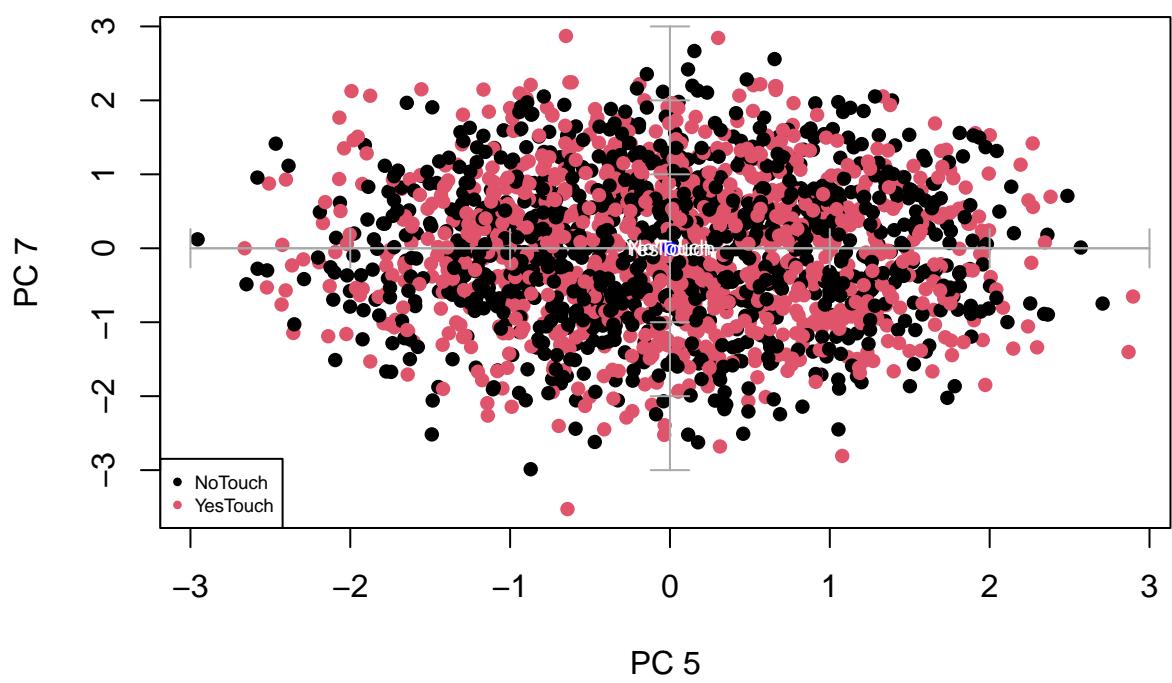
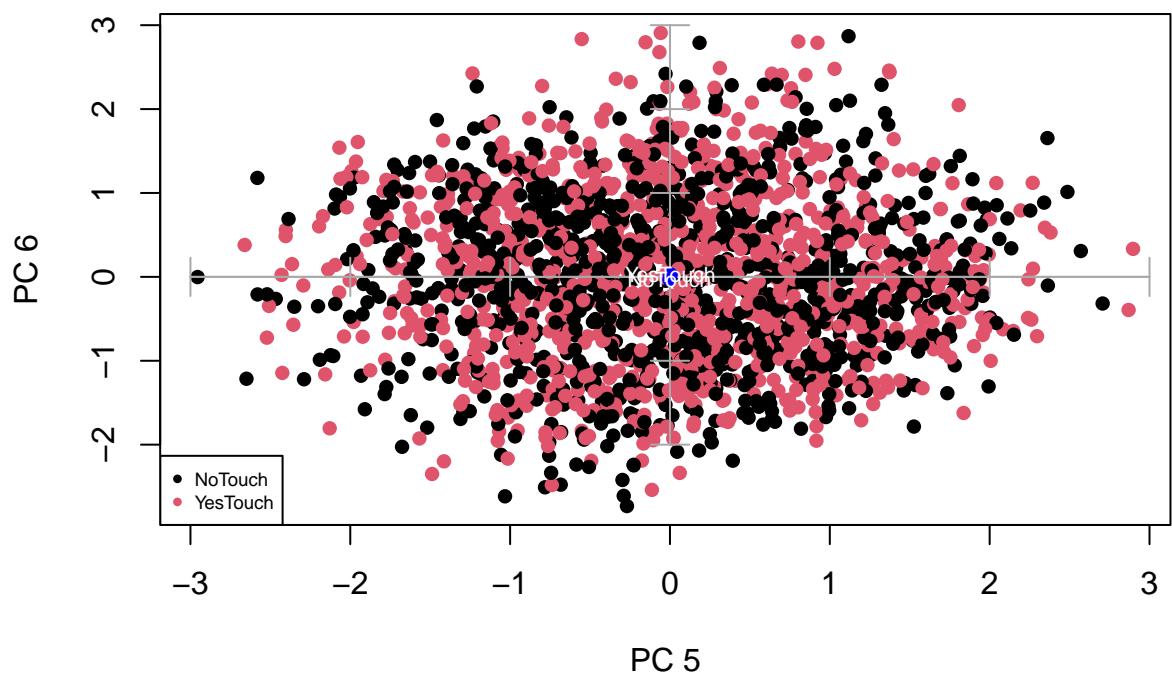


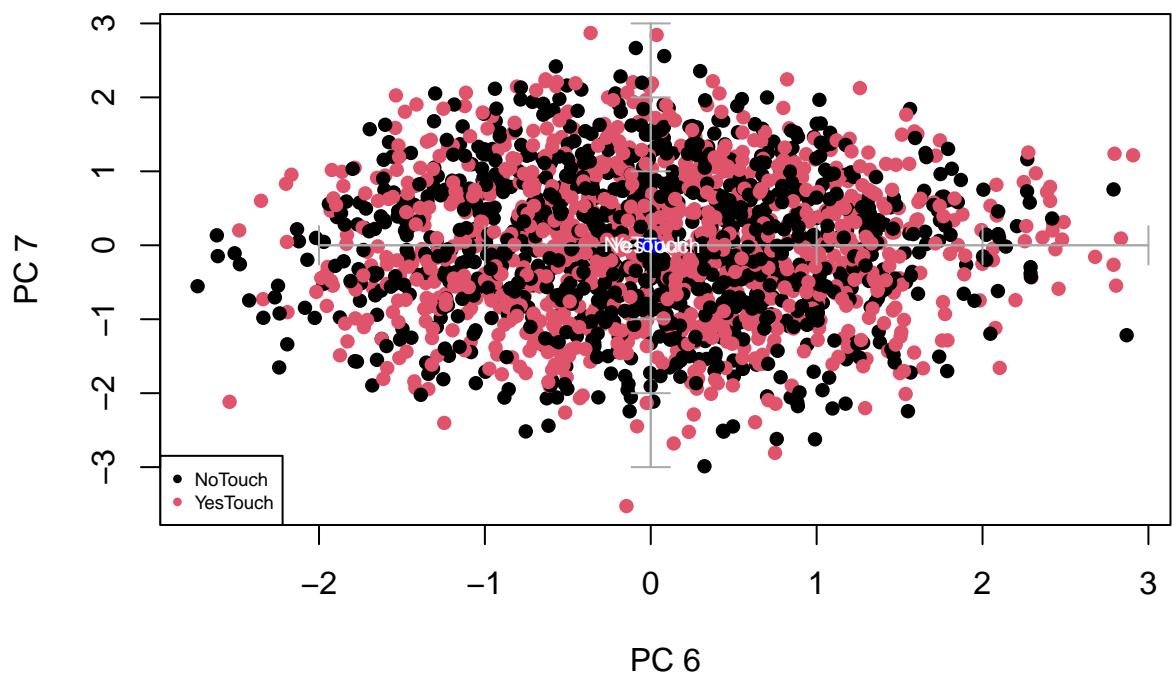
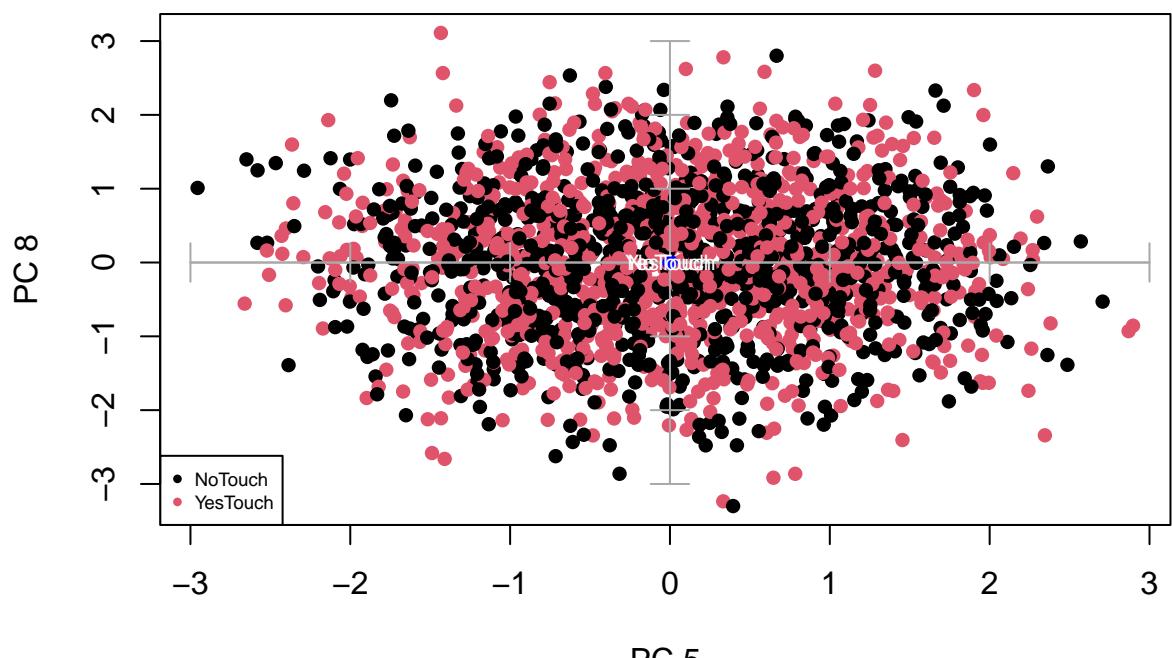


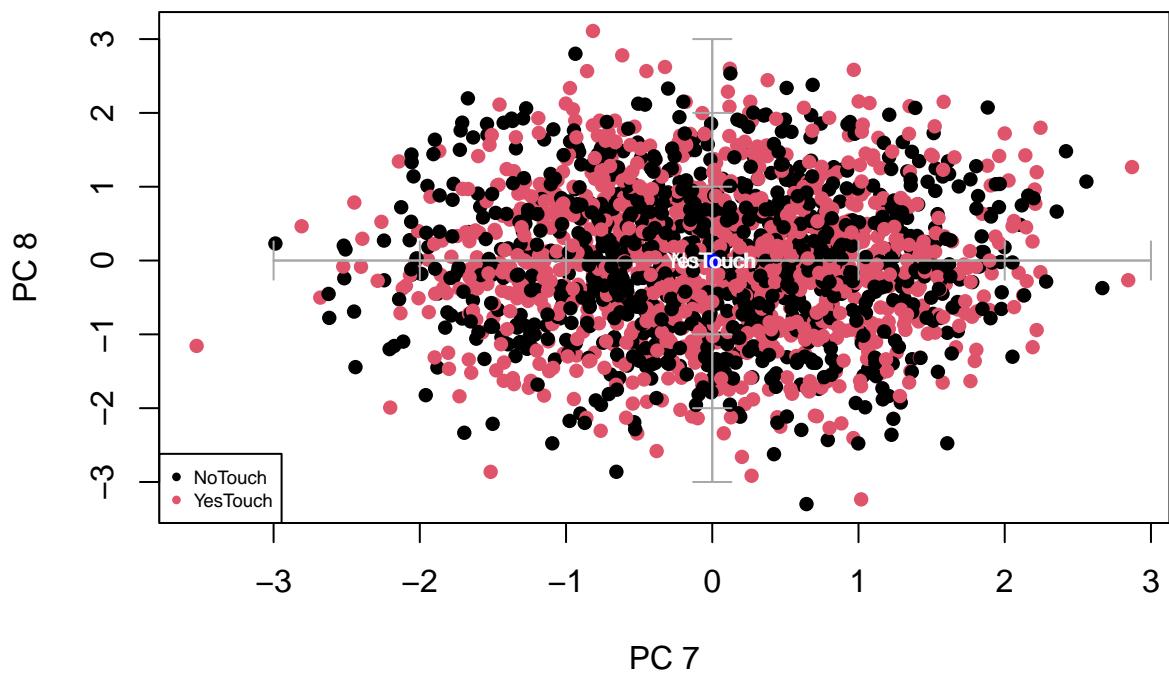
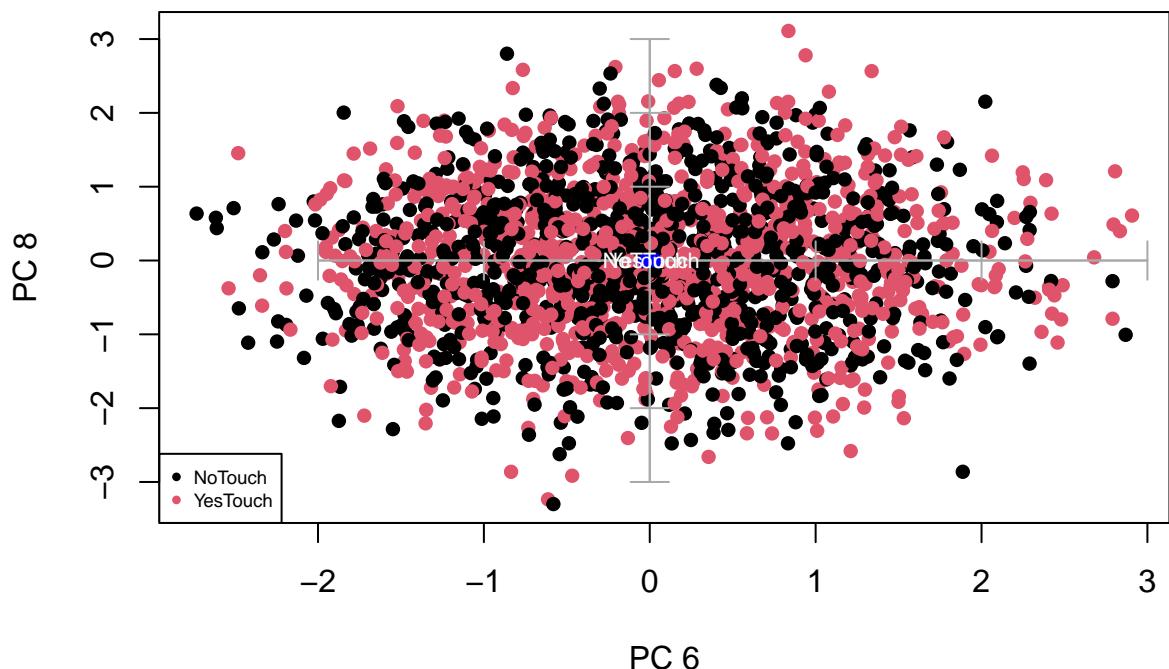












```

for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$three_g
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
  }
}
  
```

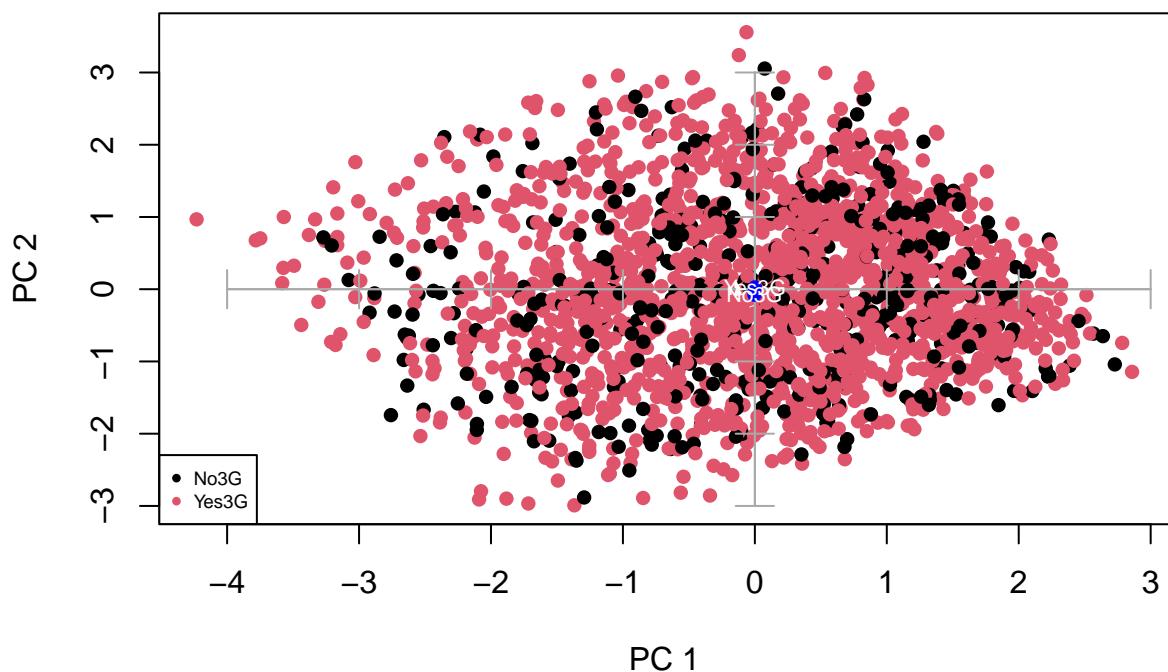
```

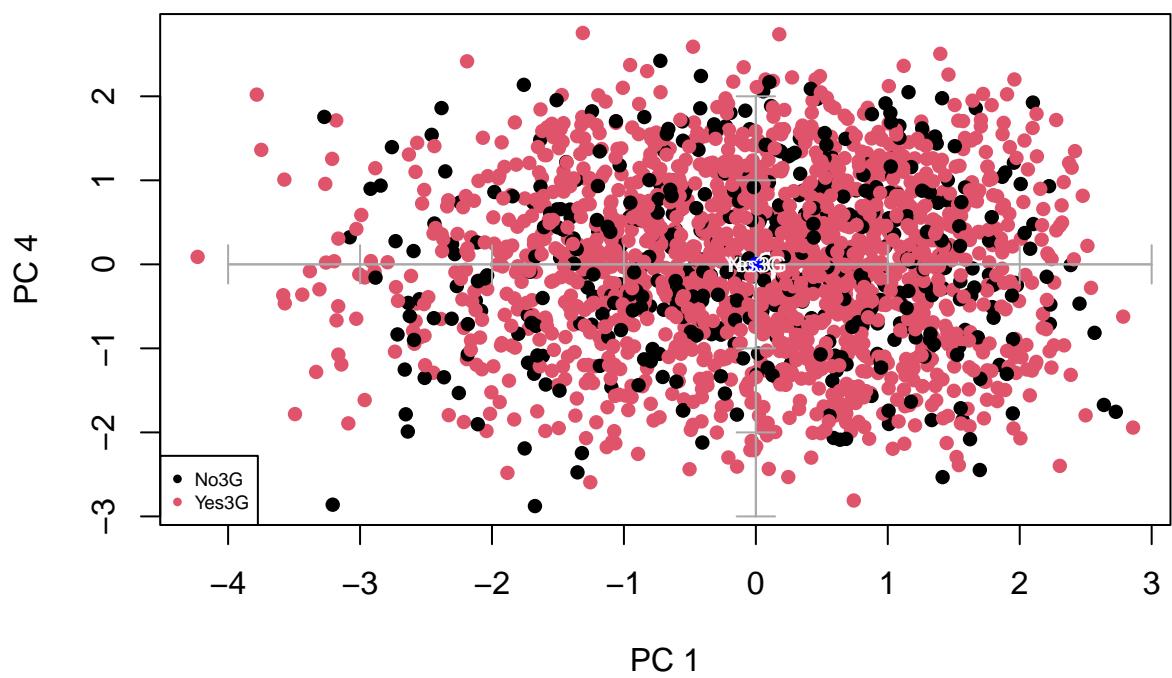
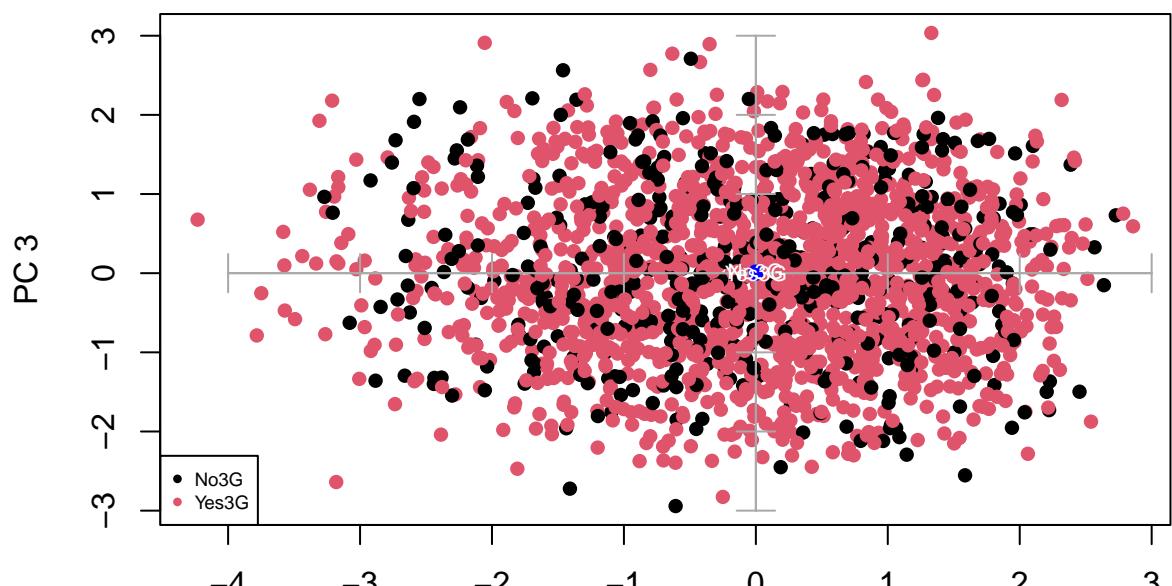
legend("bottomleft",levels(varcat),pch=16,col=c(1:2), cex=0.6)

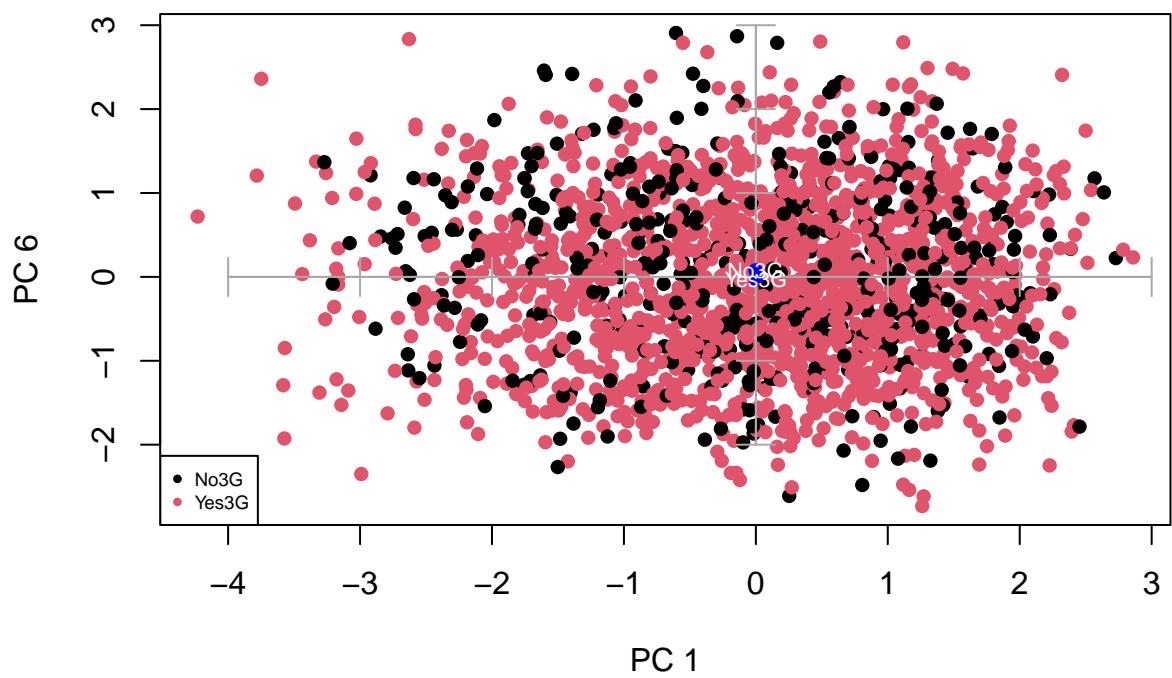
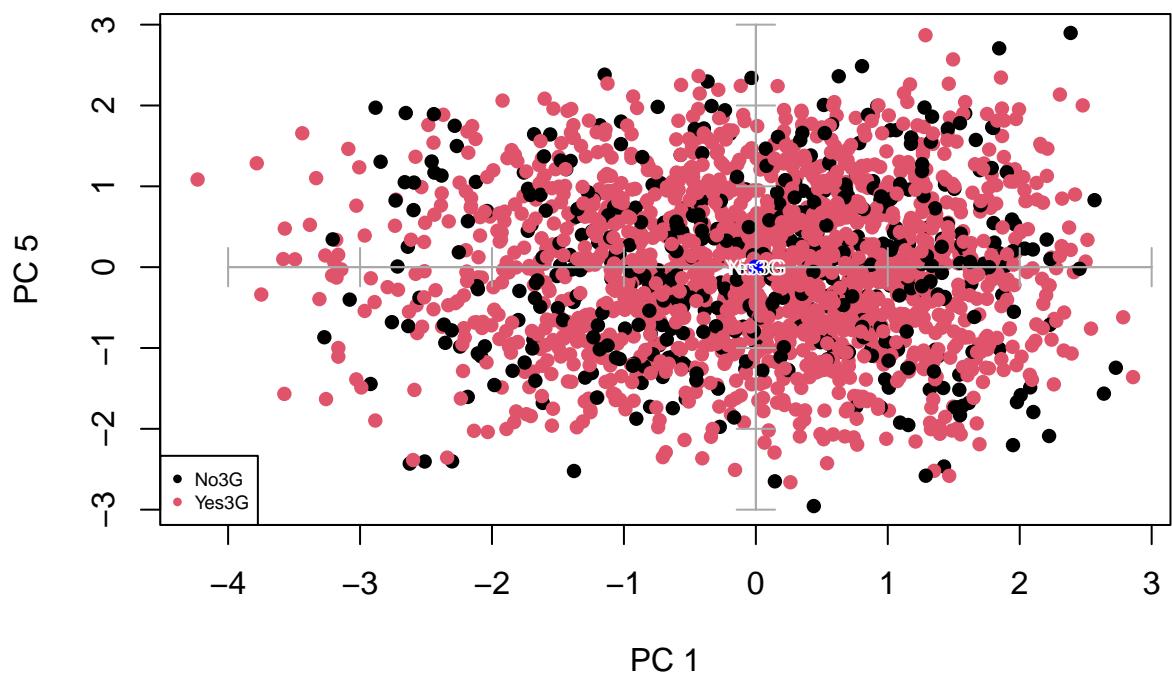
#select your qualitative variable
fdic1 = tapply(Psi[,i],varcat,mean)
fdic2 = tapply(Psi[,j],varcat,mean)
points(fdic1,fdic2,pch=16,col="blue")
text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
}

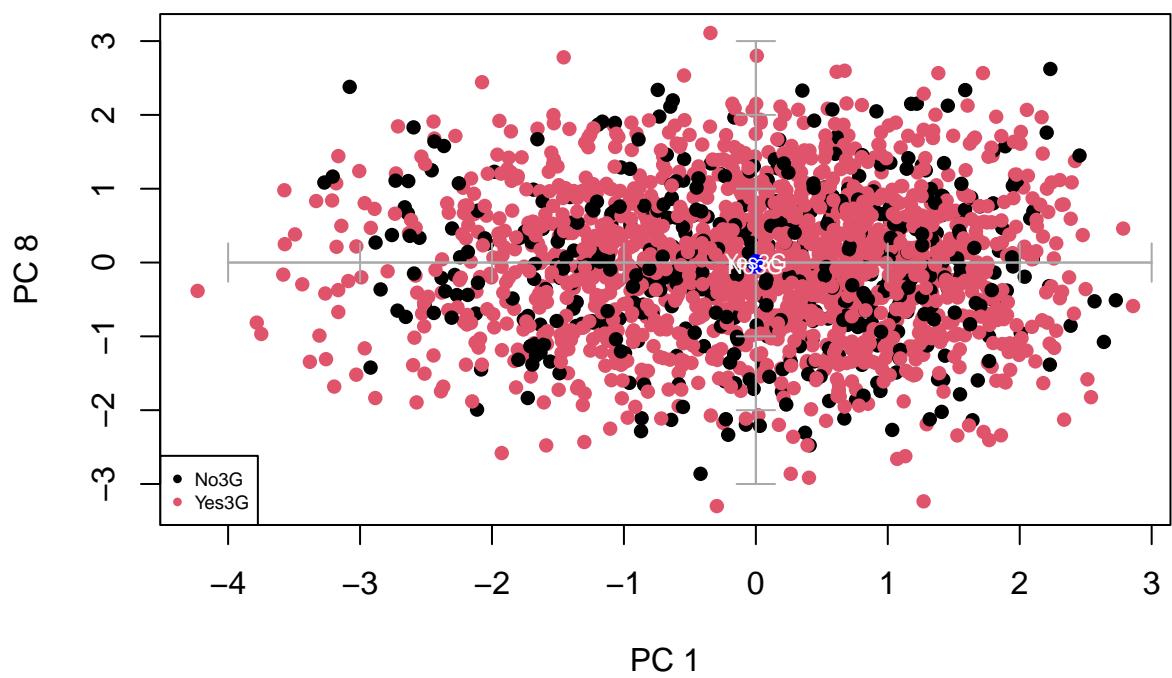
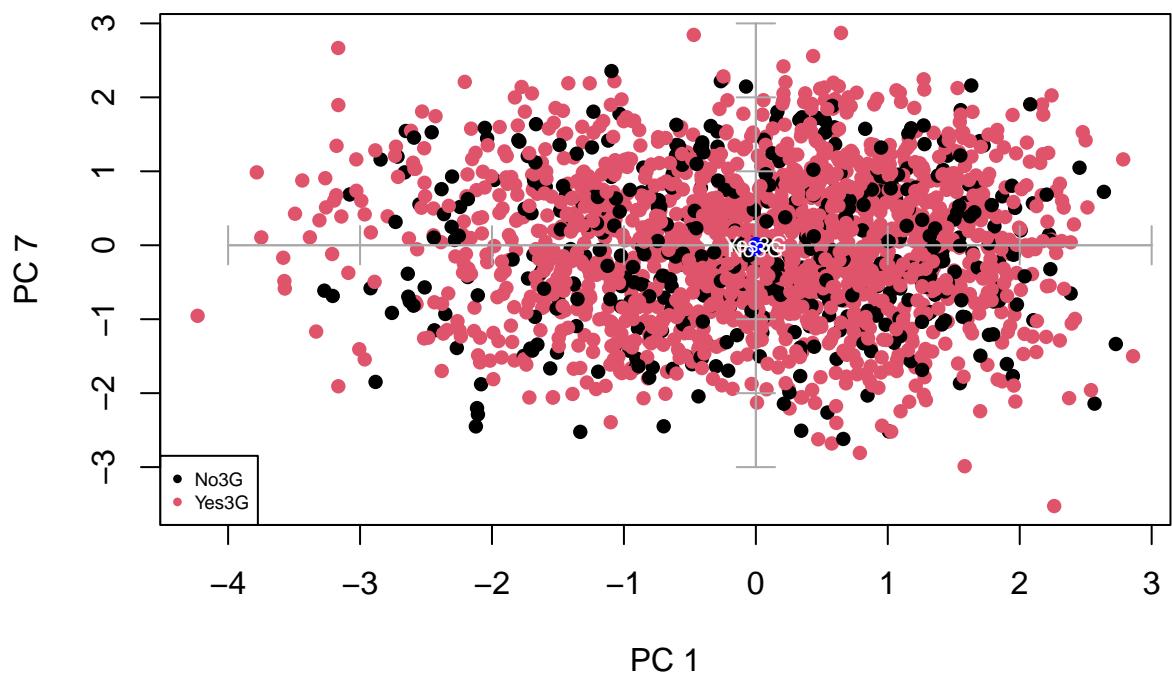
}

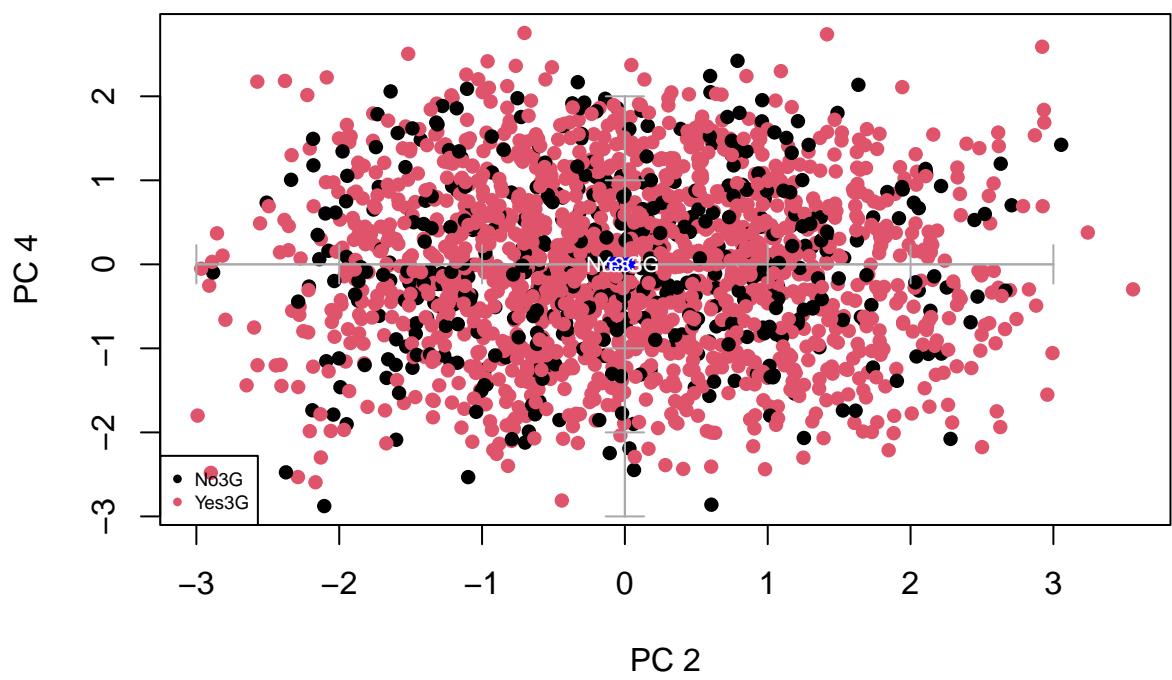
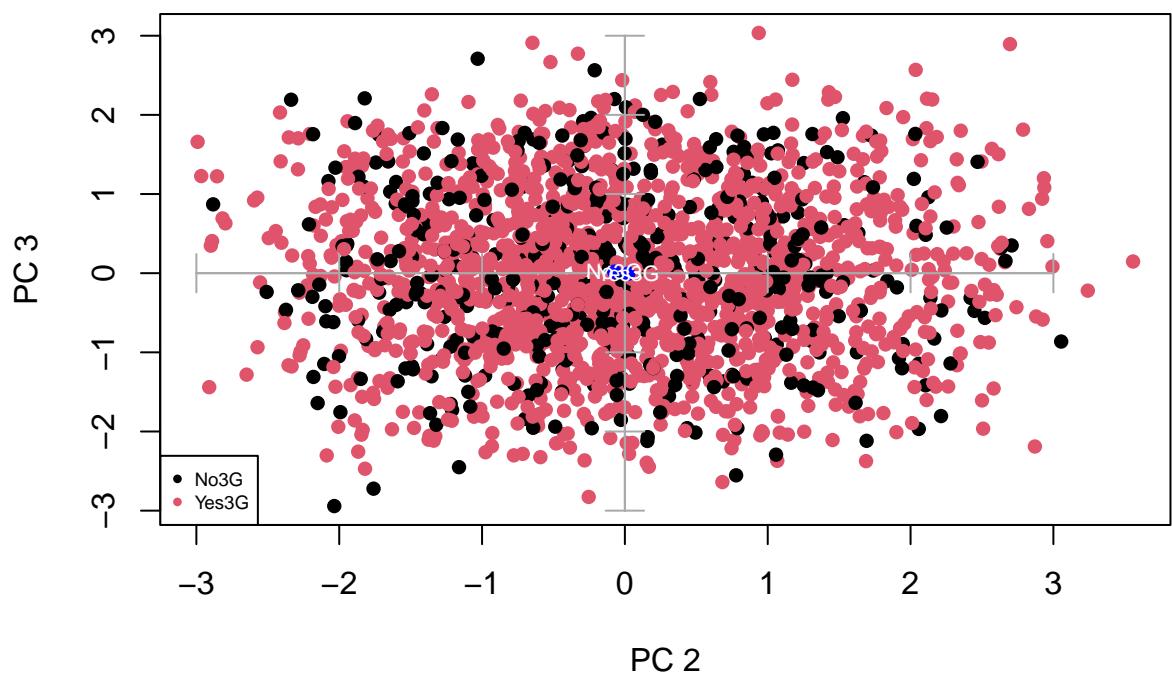
```

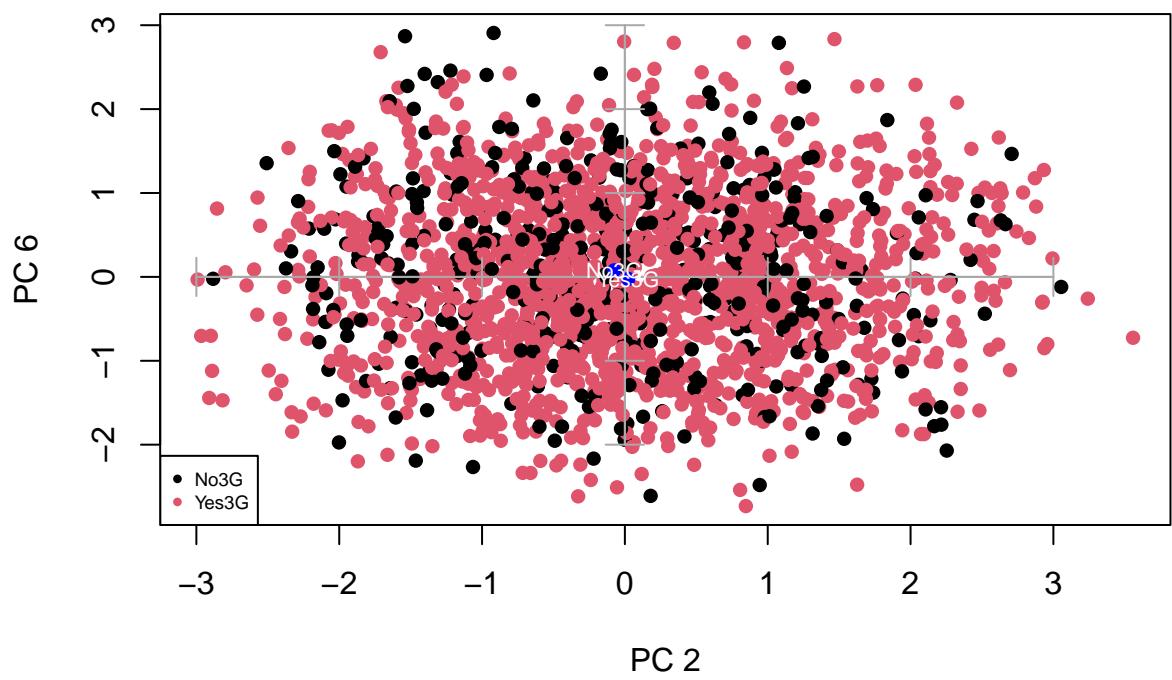
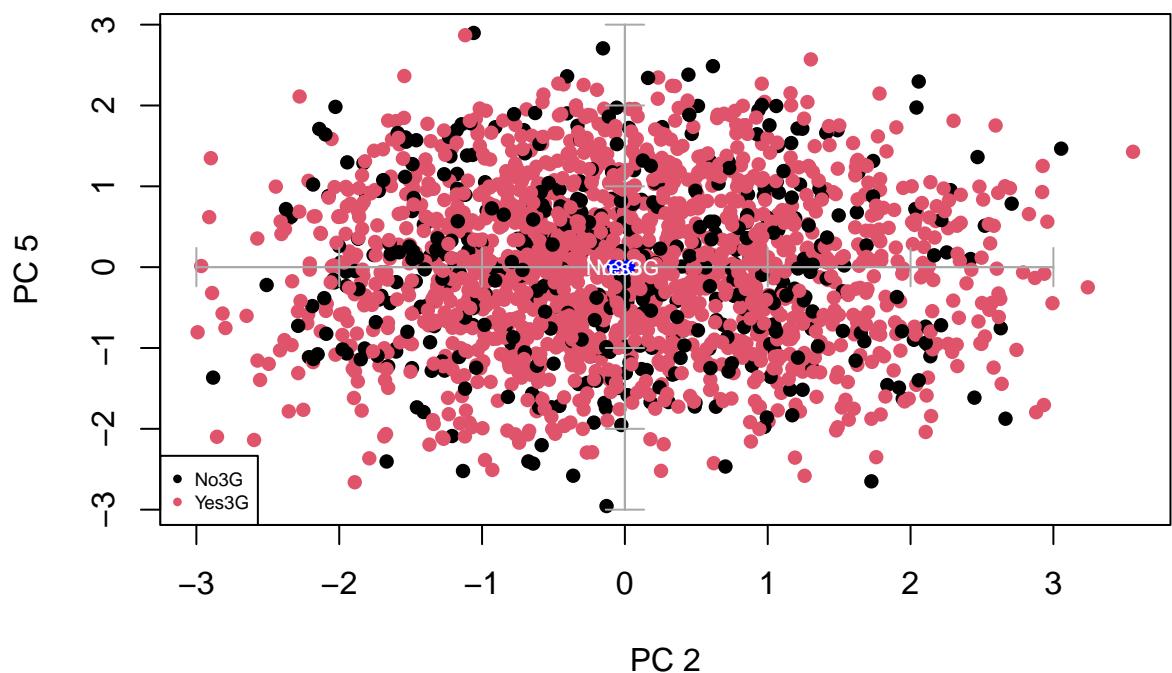


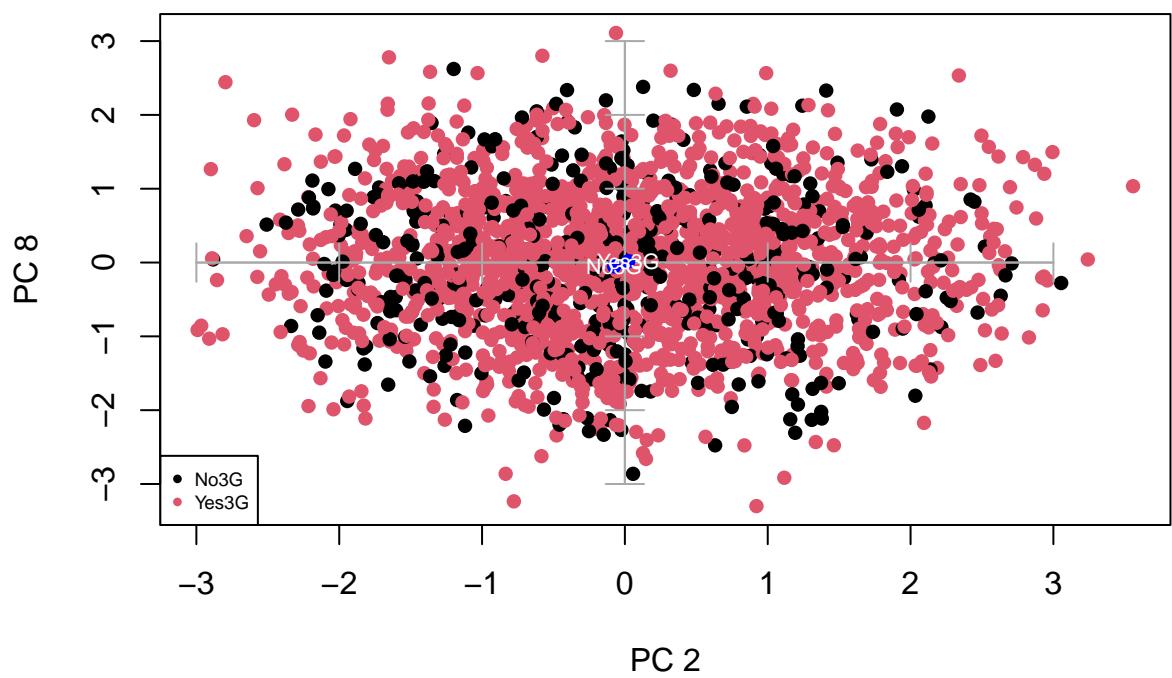
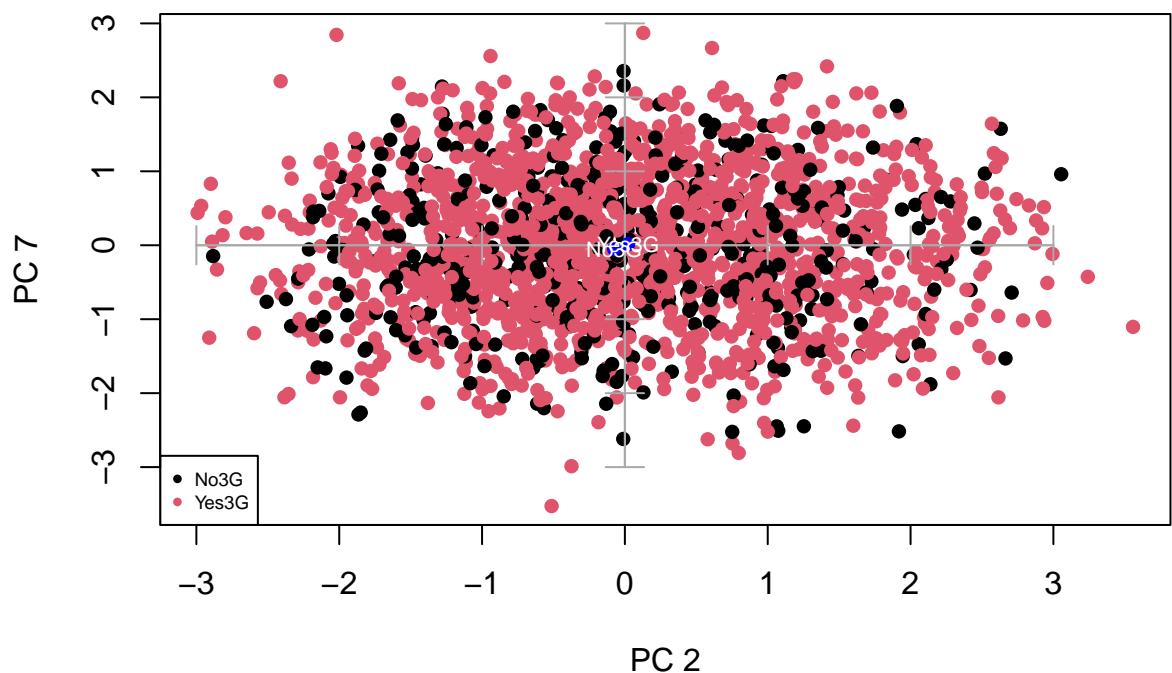


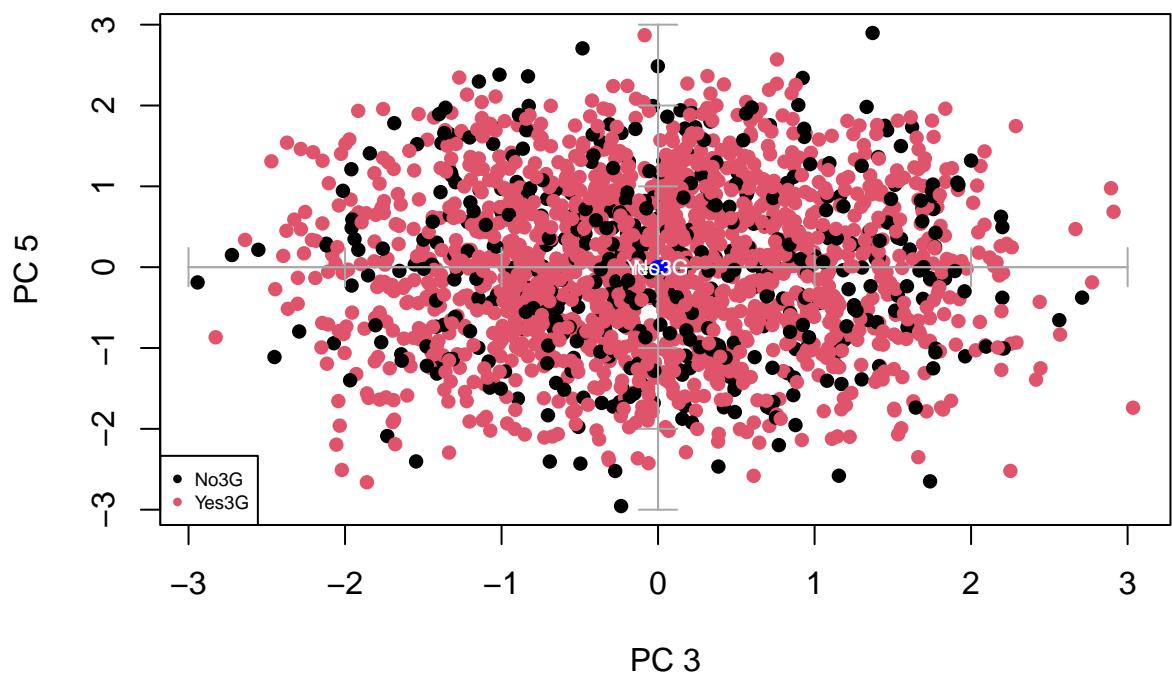
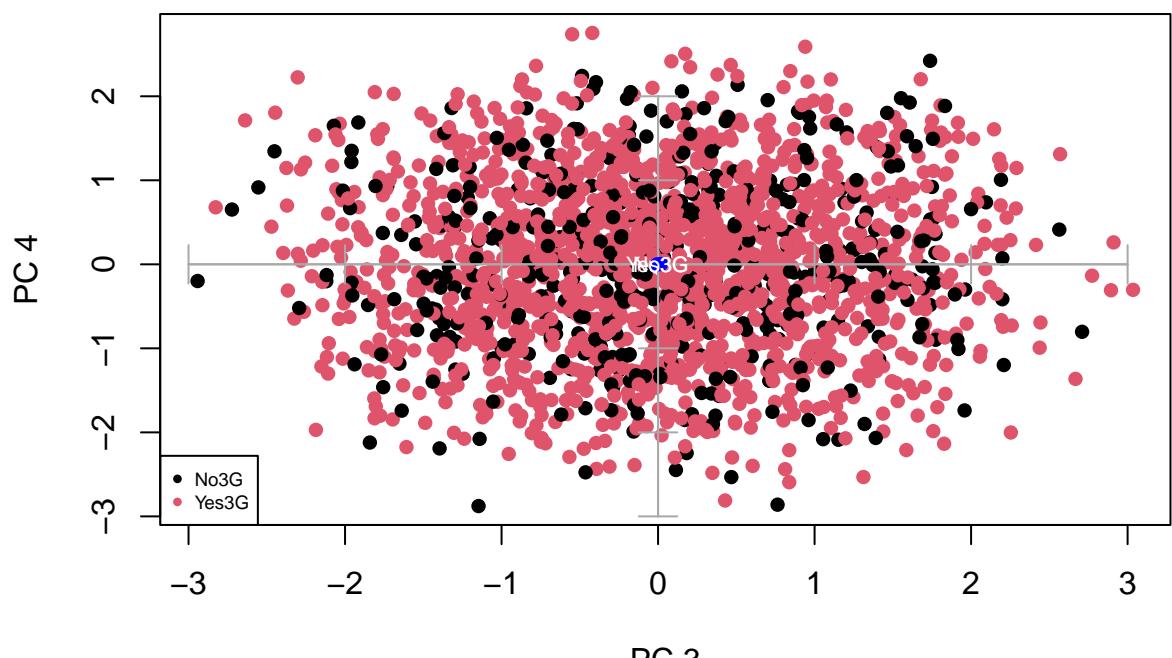


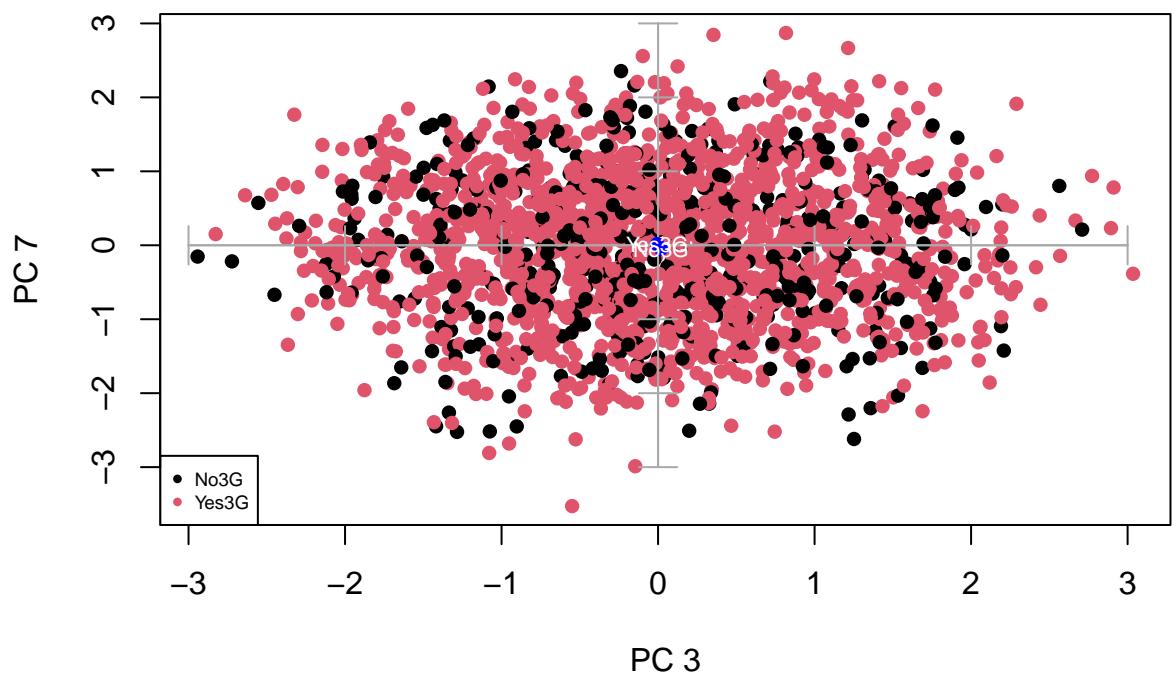
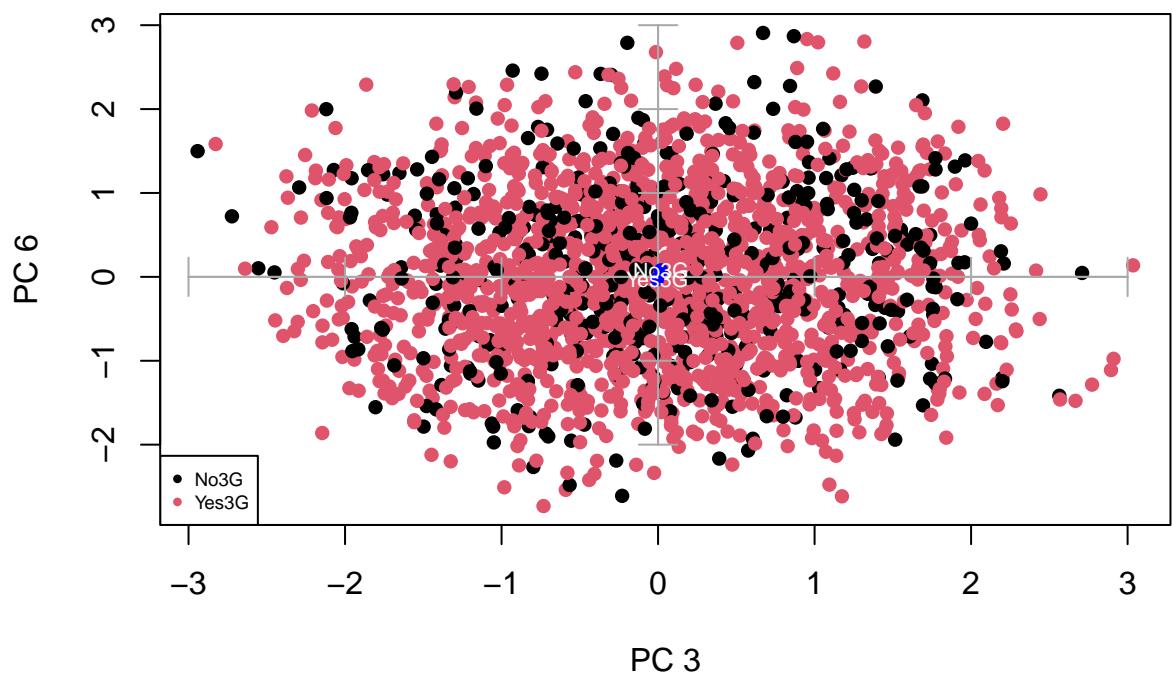


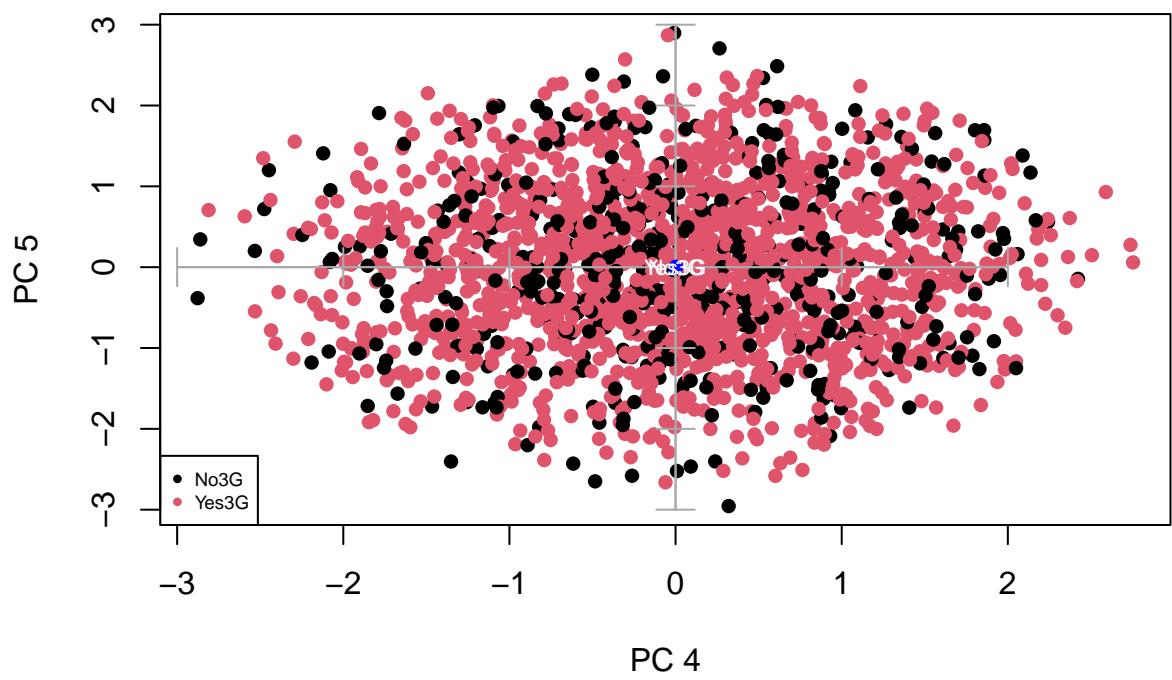
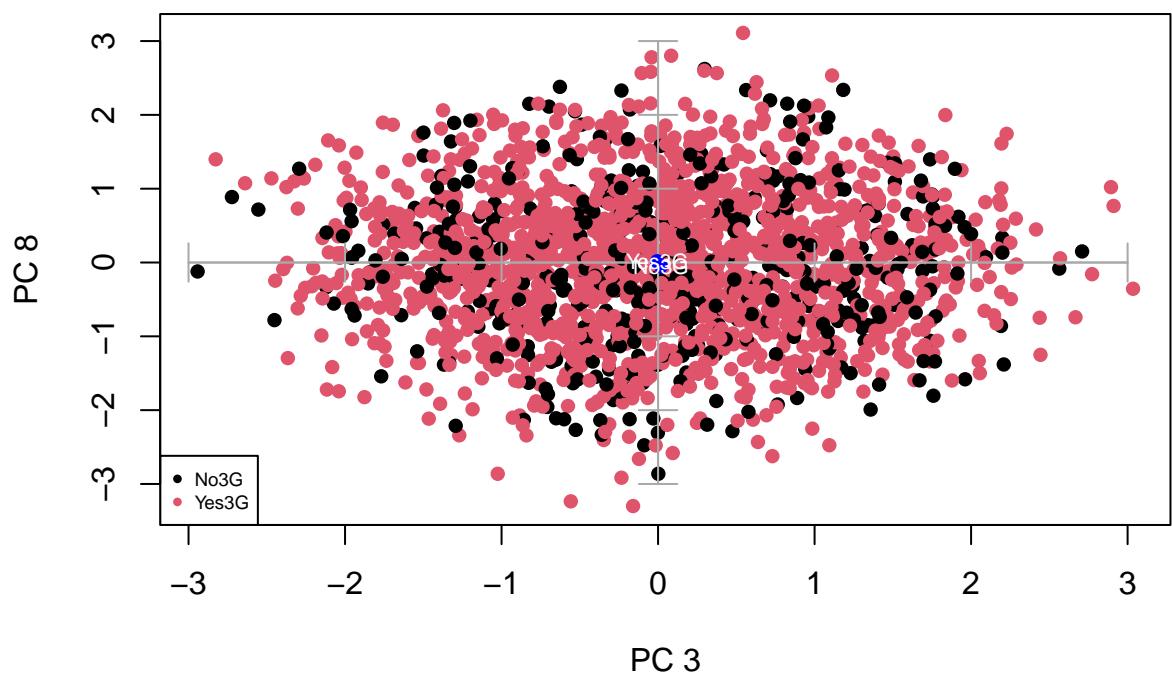


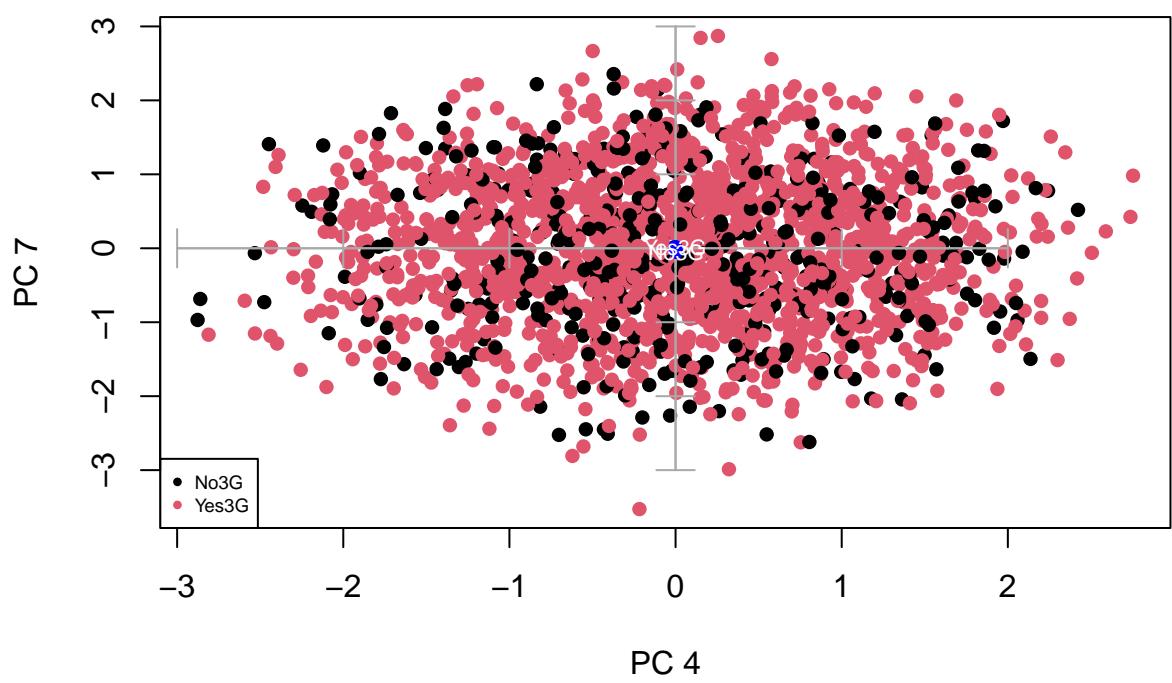
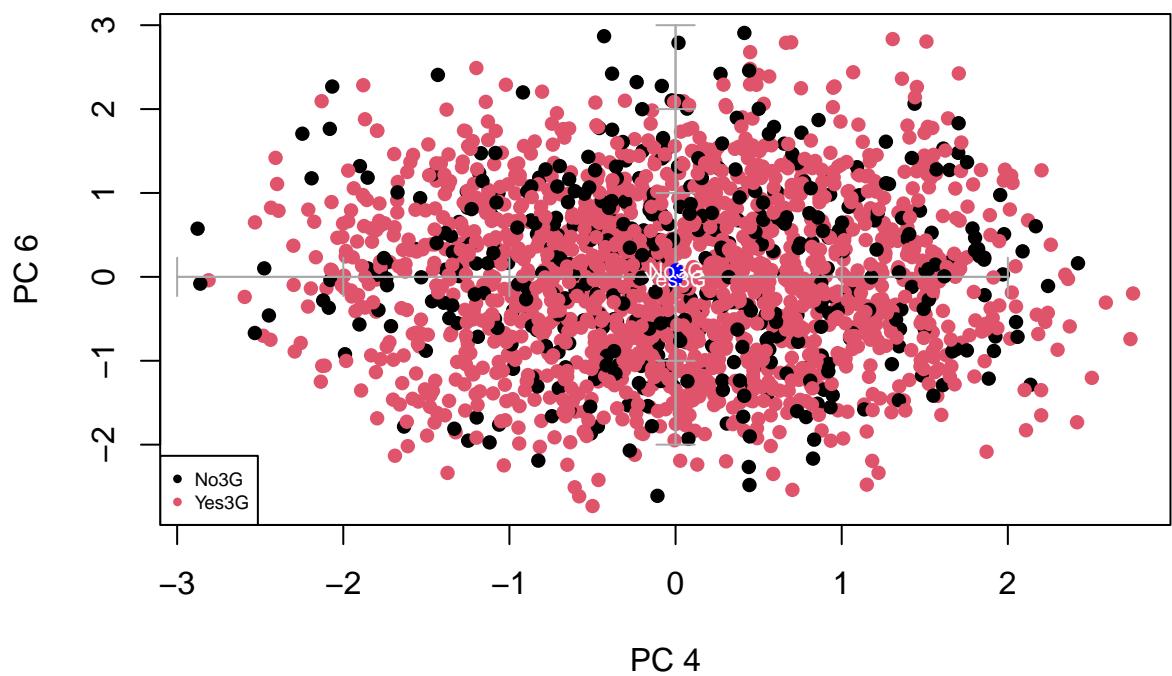


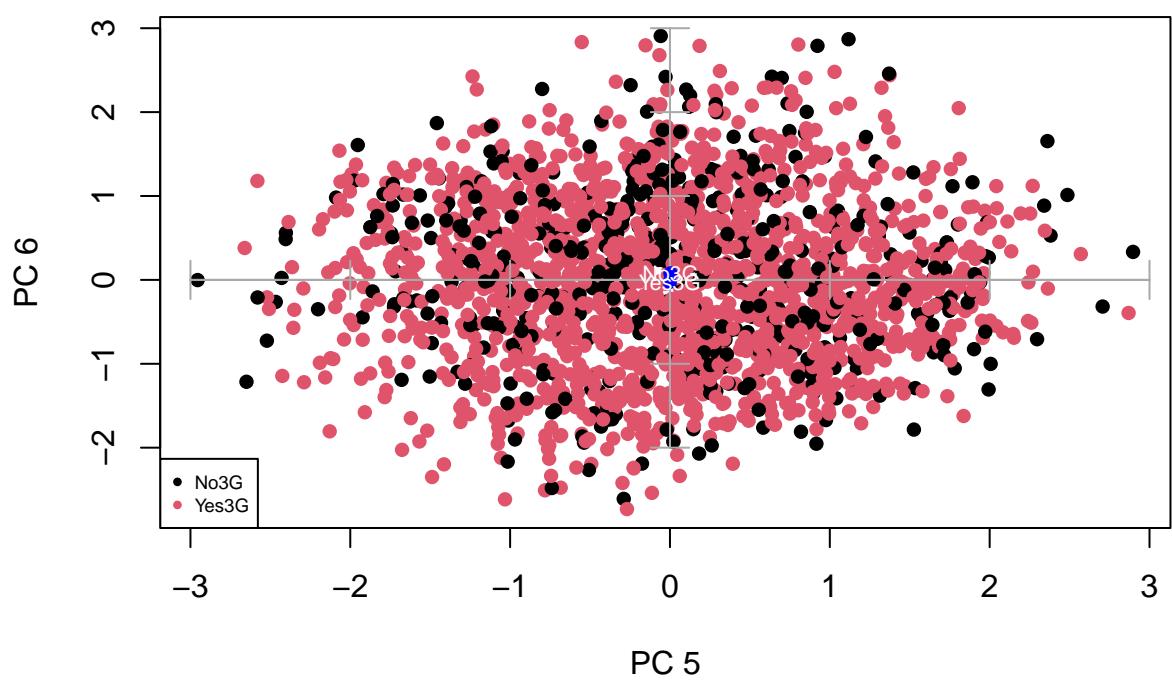
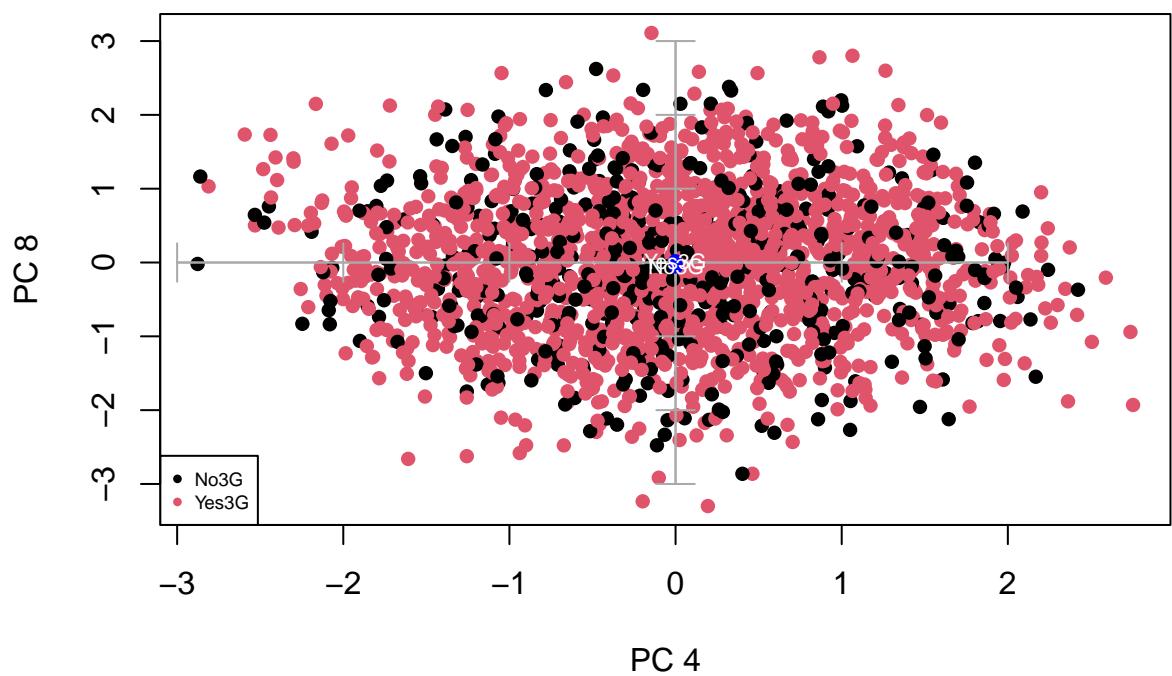


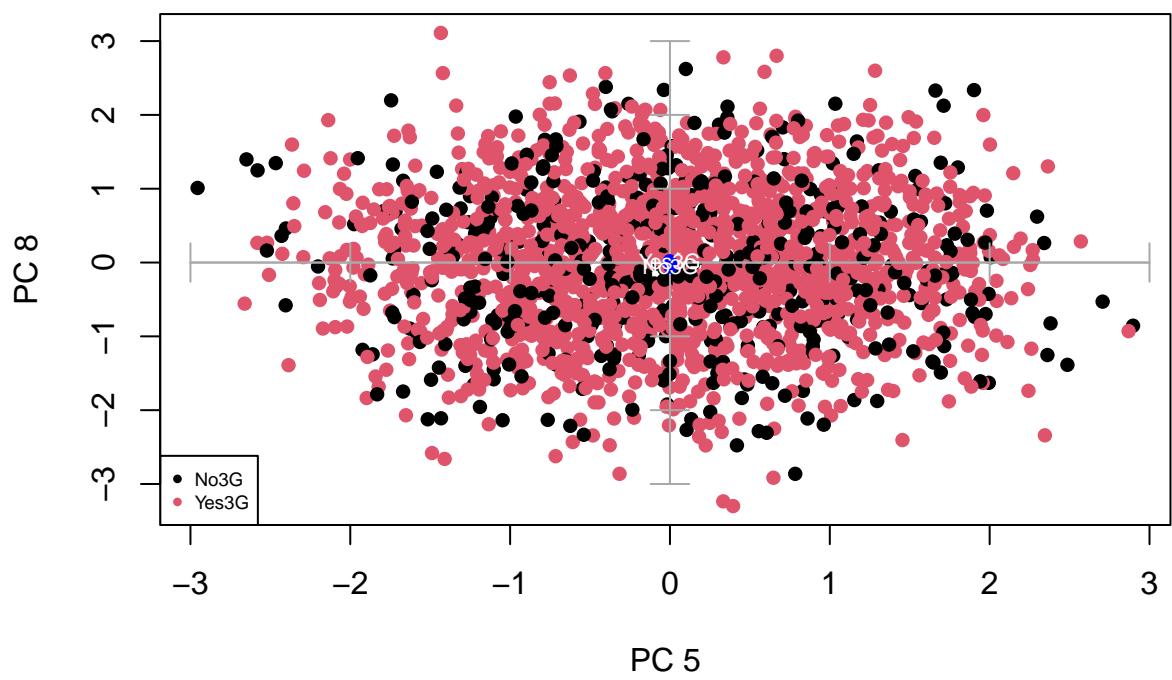
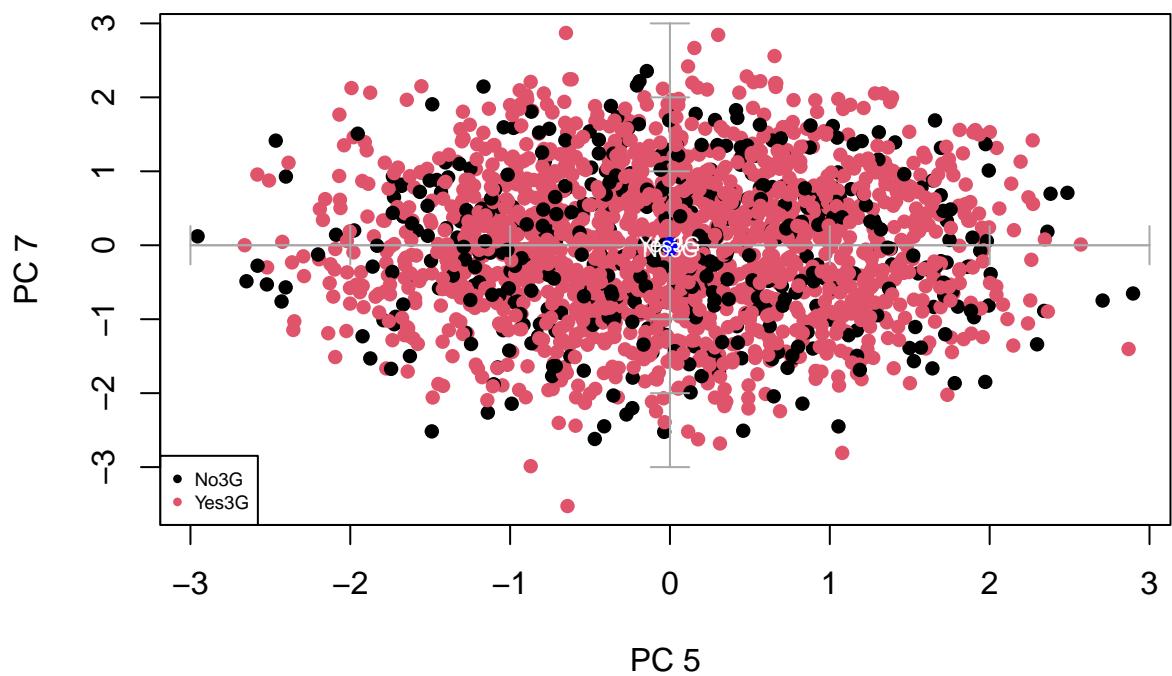


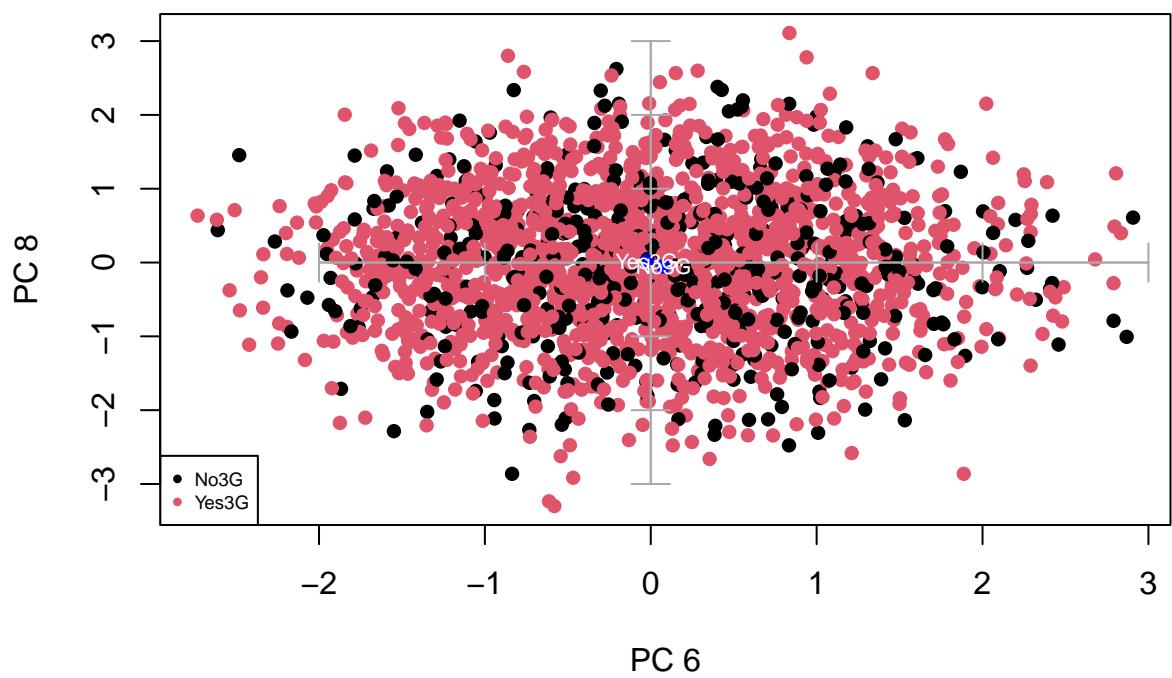
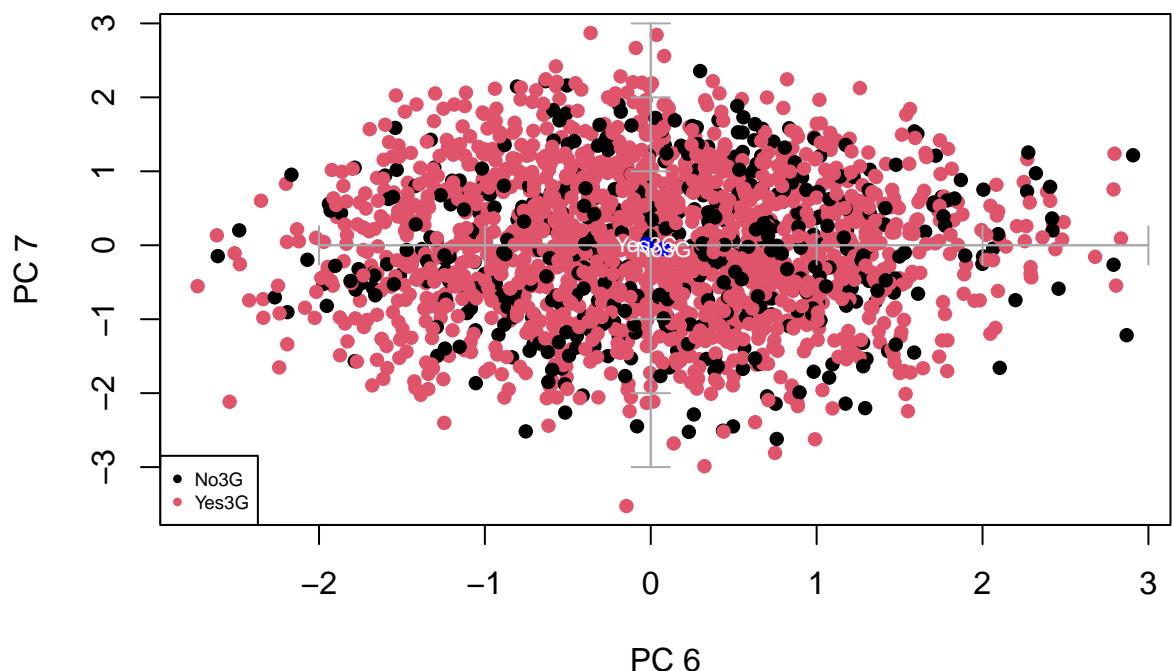


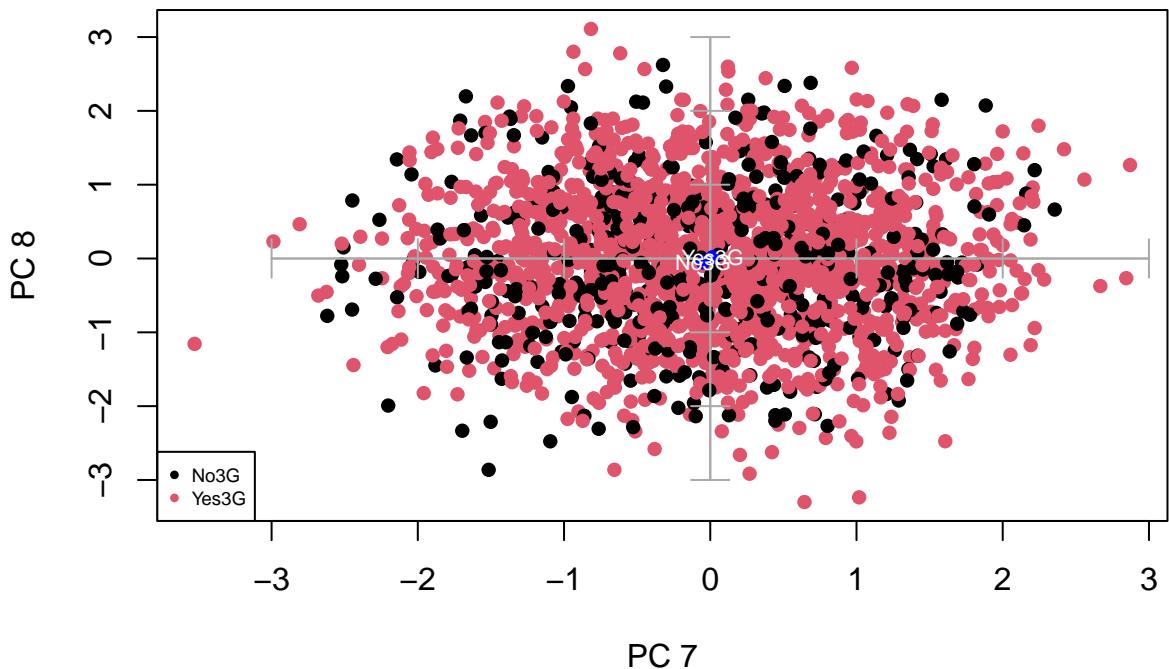








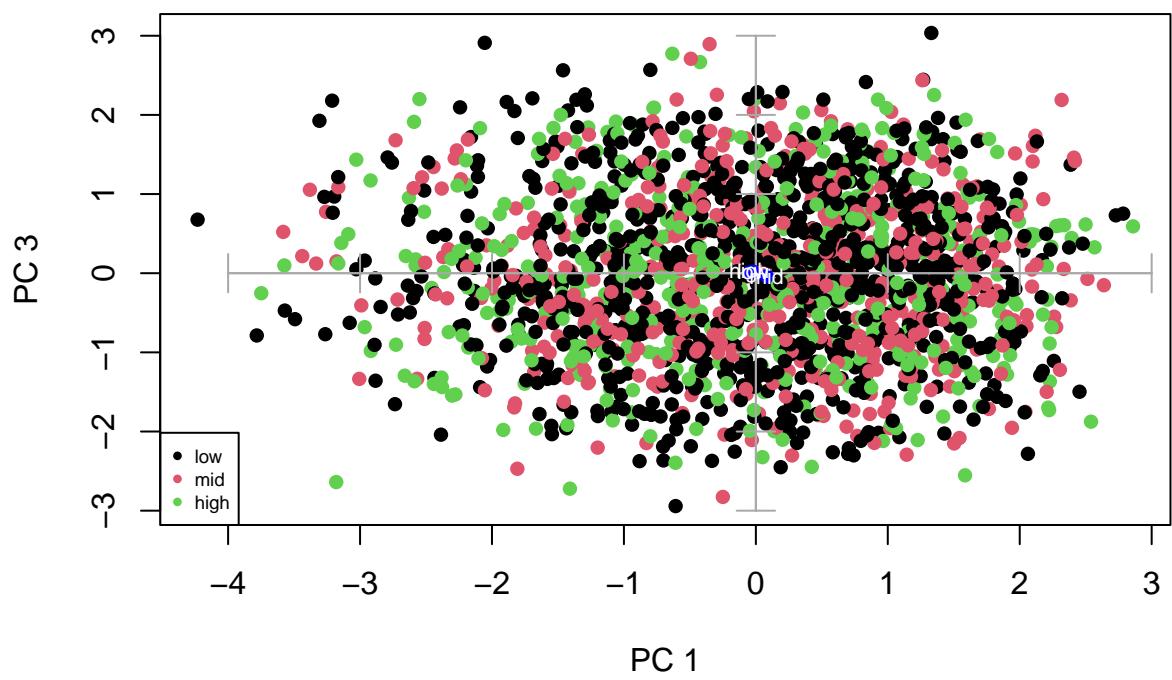
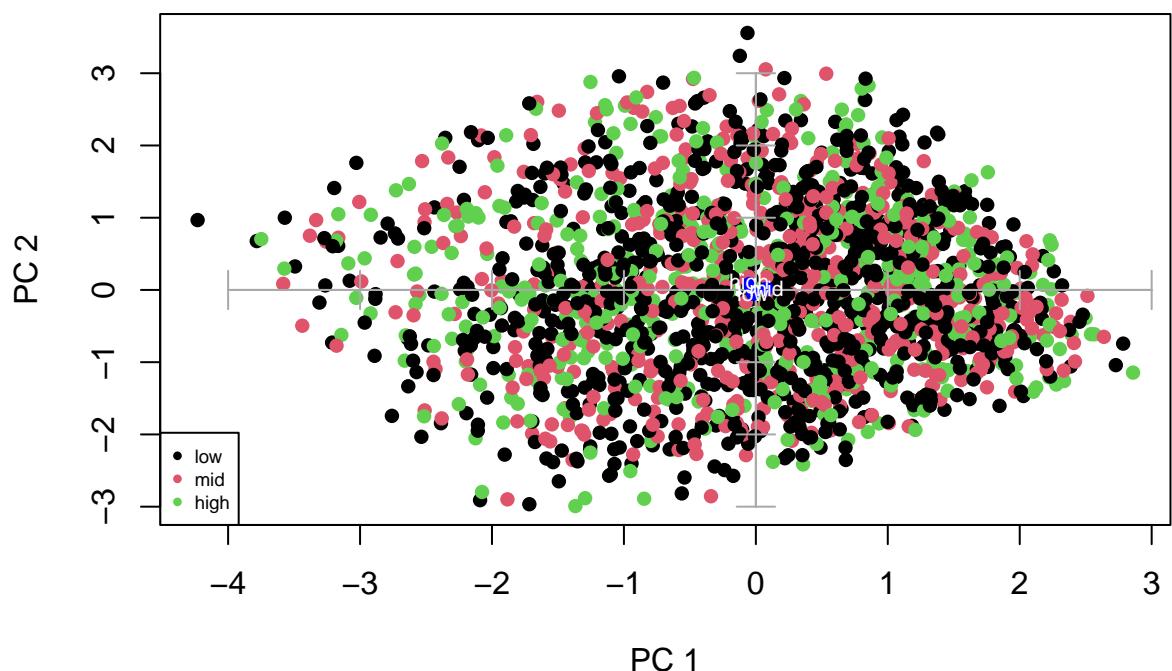


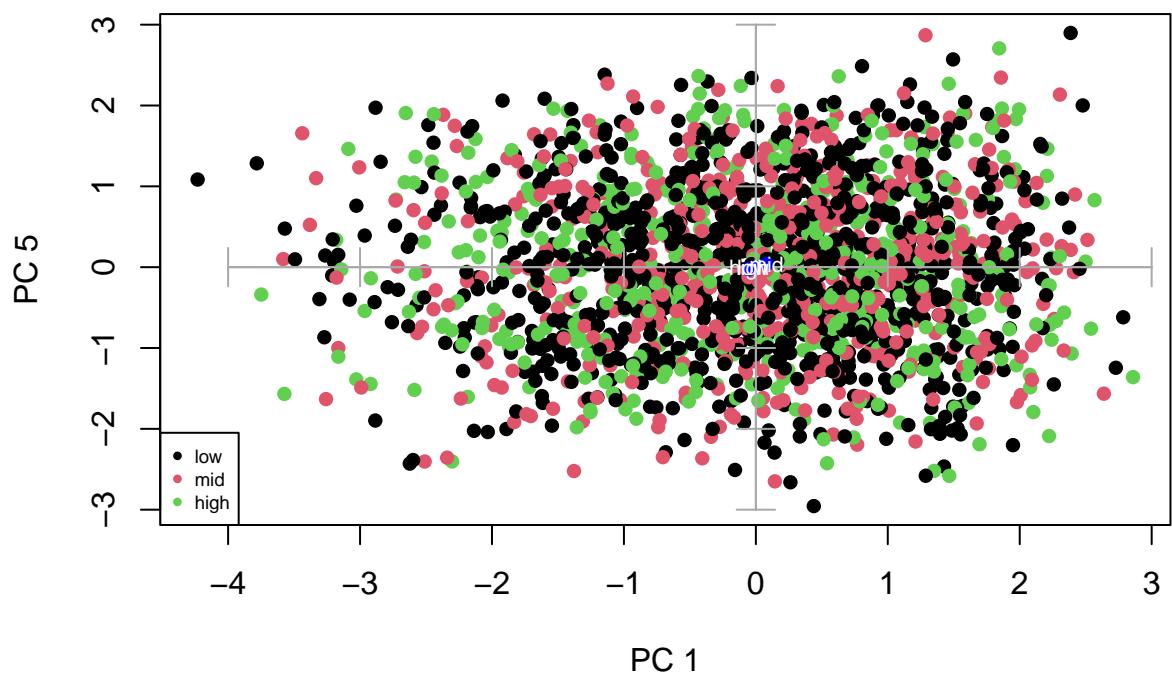
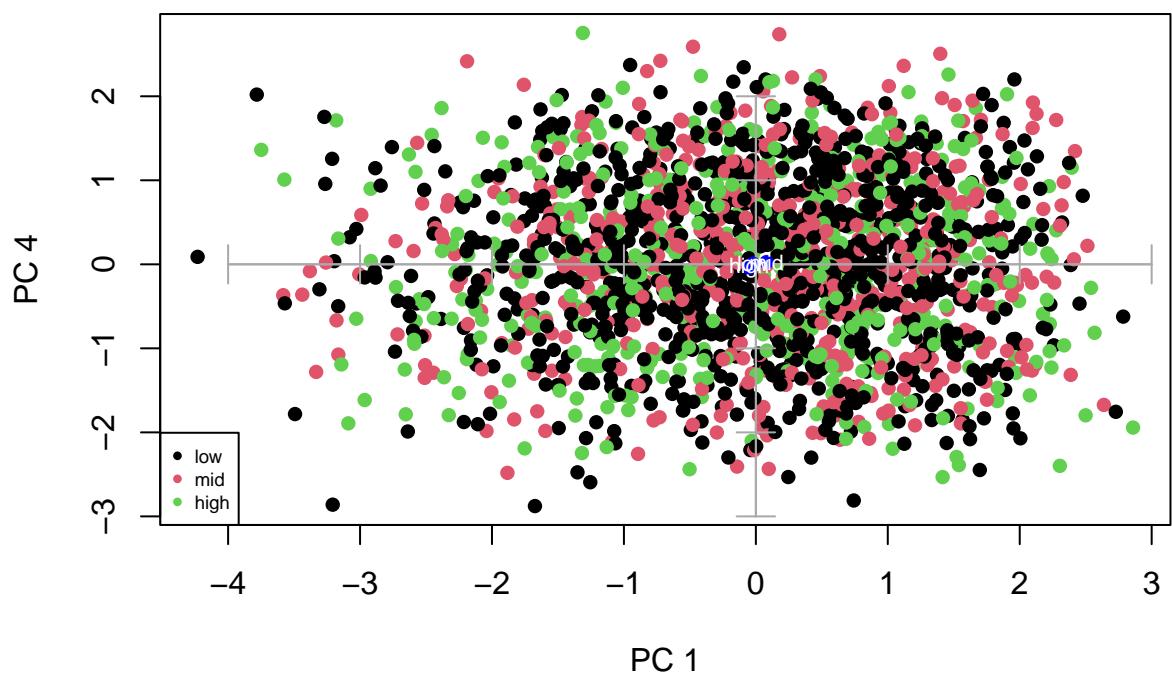


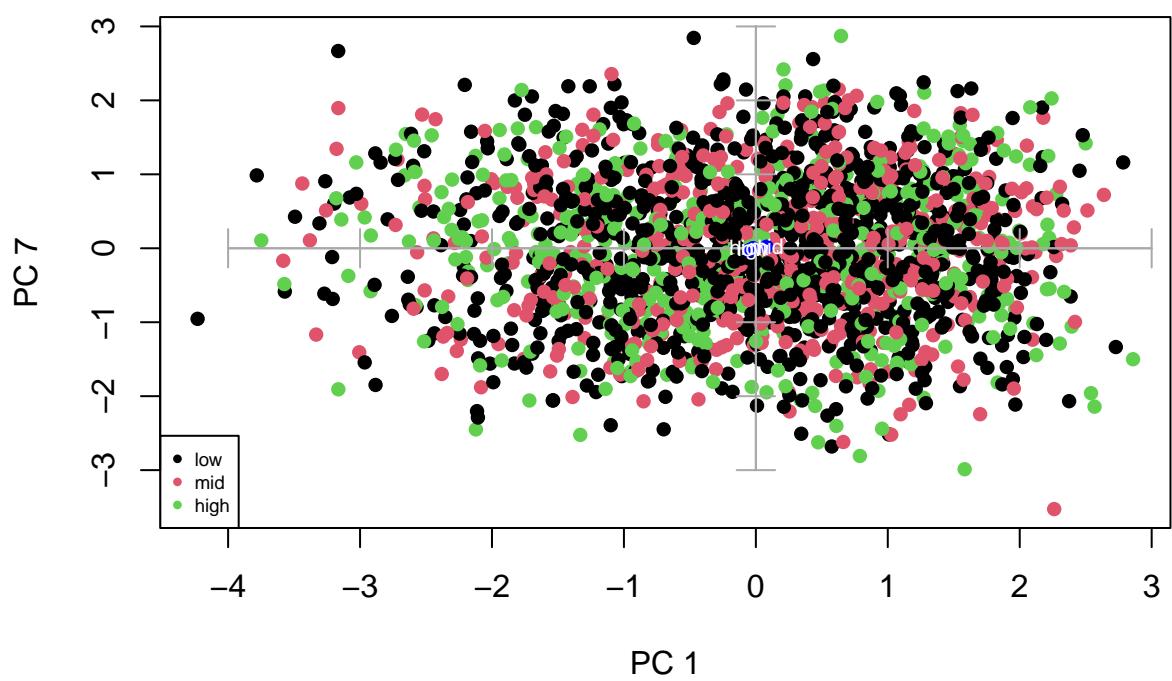
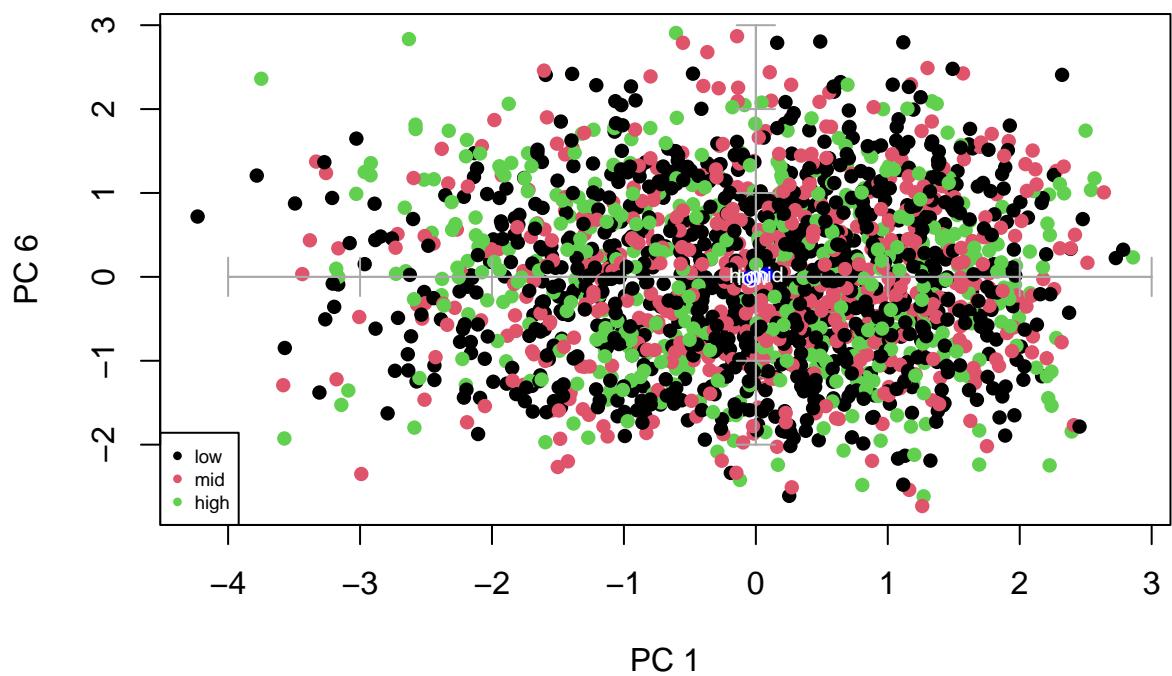
```

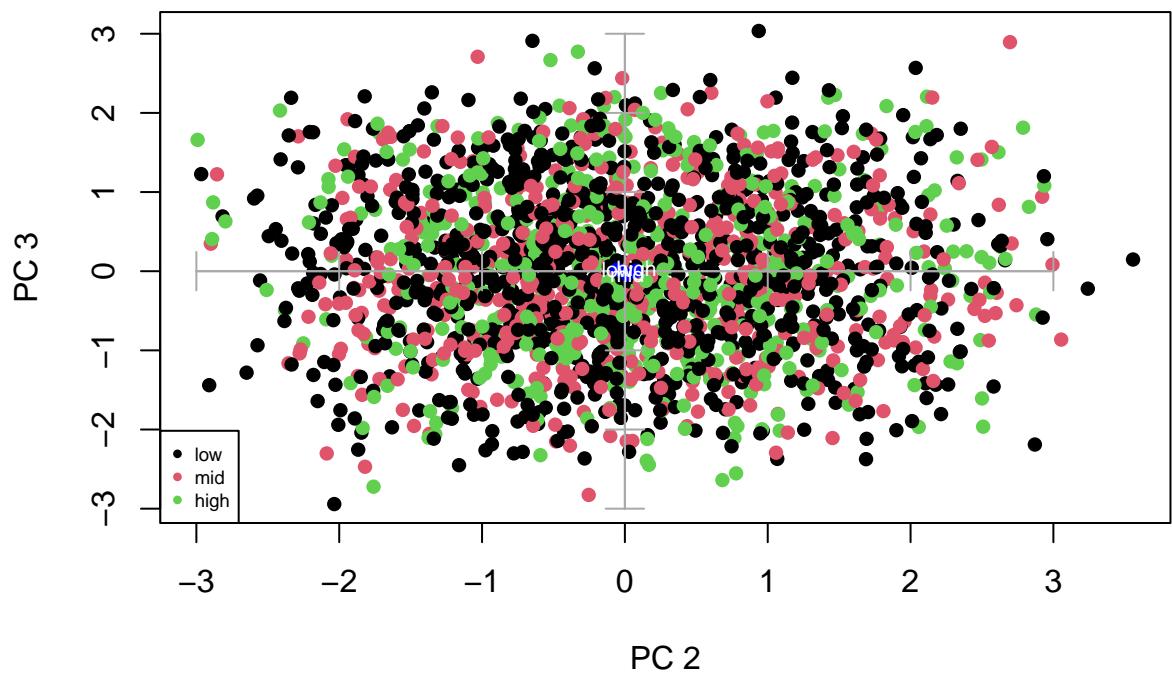
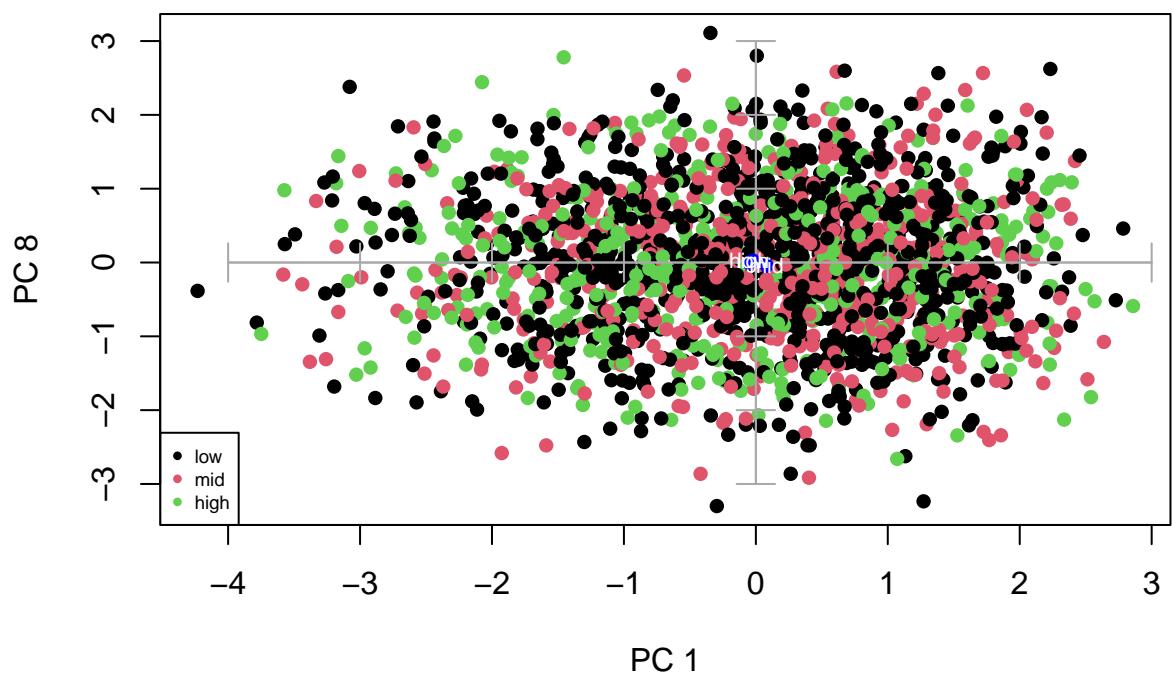
for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$ram
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
    legend("bottomleft",levels(varcat),pch=16,col=c(1:3), cex=0.6)

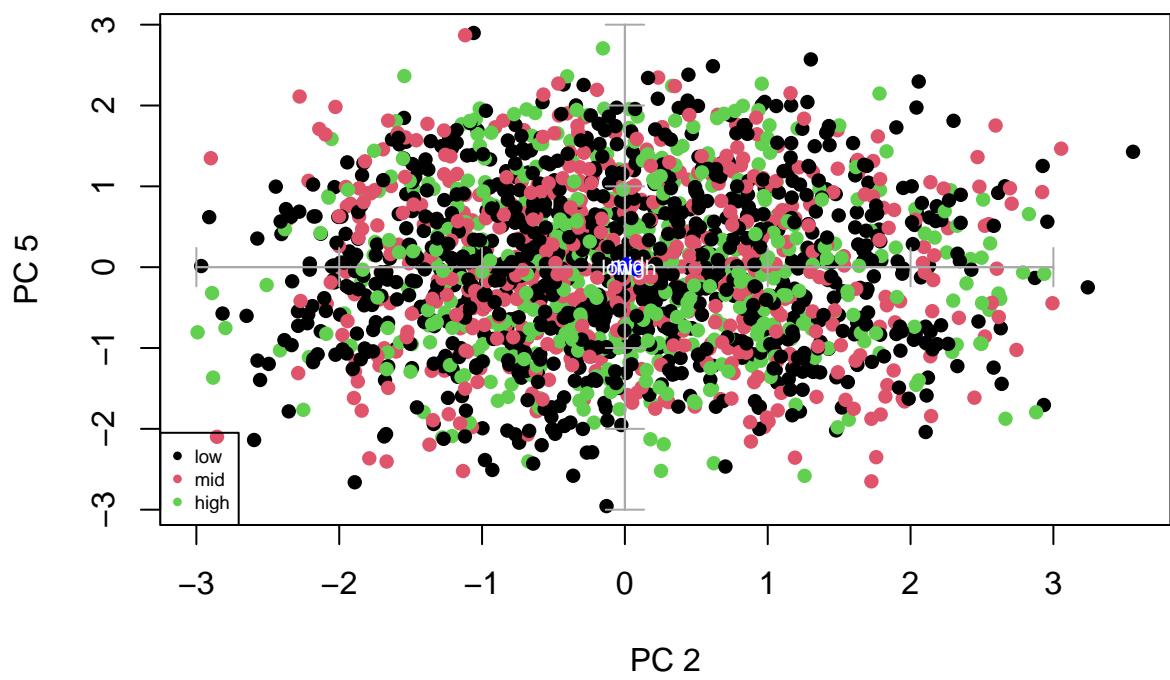
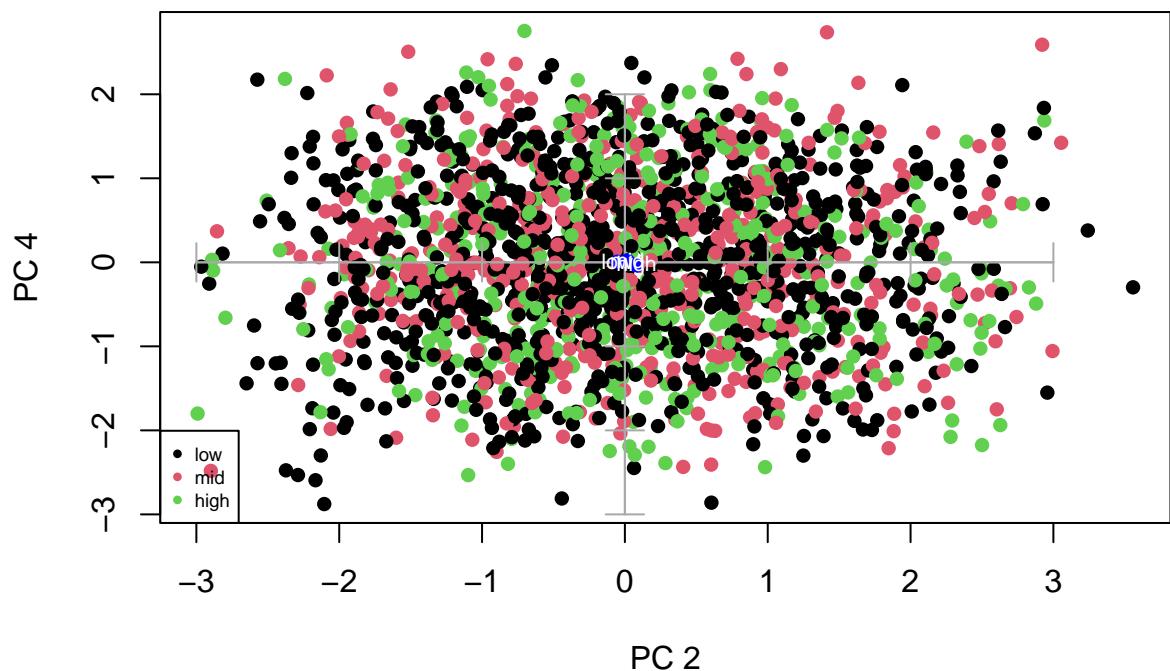
    #select your qualitative variable
    fdic1 = tapply(Psi[,i],varcat,mean)
    fdic2 = tapply(Psi[,j],varcat,mean)
    points(fdic1,fdic2,pch=16,col="blue")
    text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
  }
}
  
```

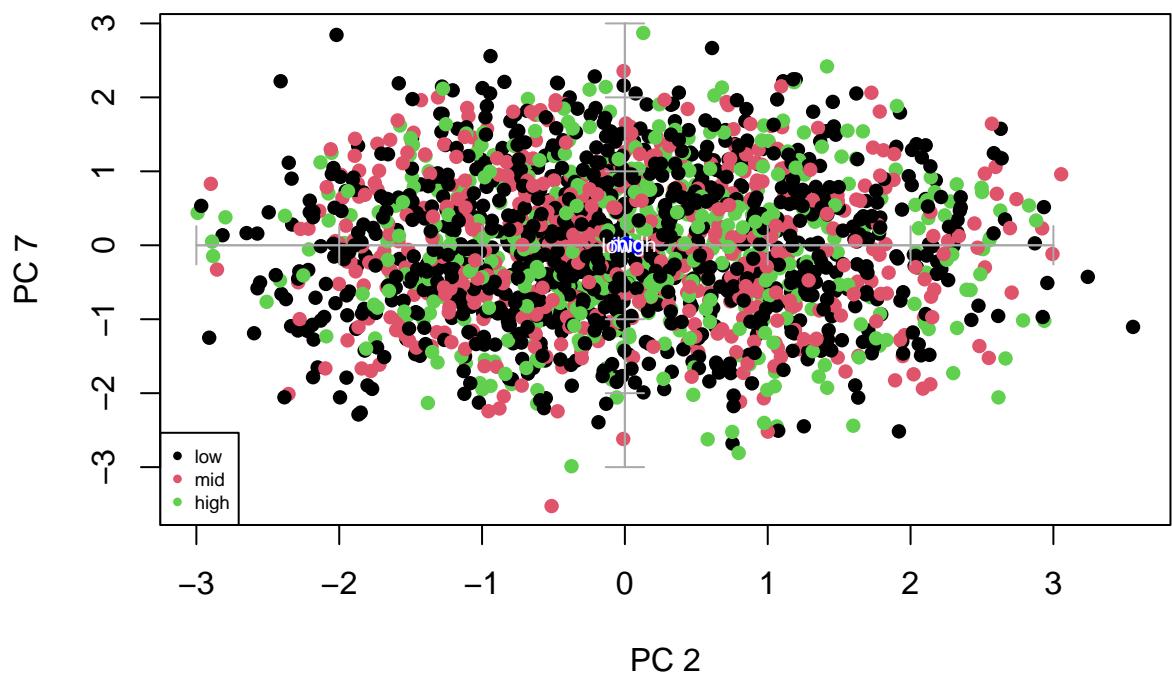
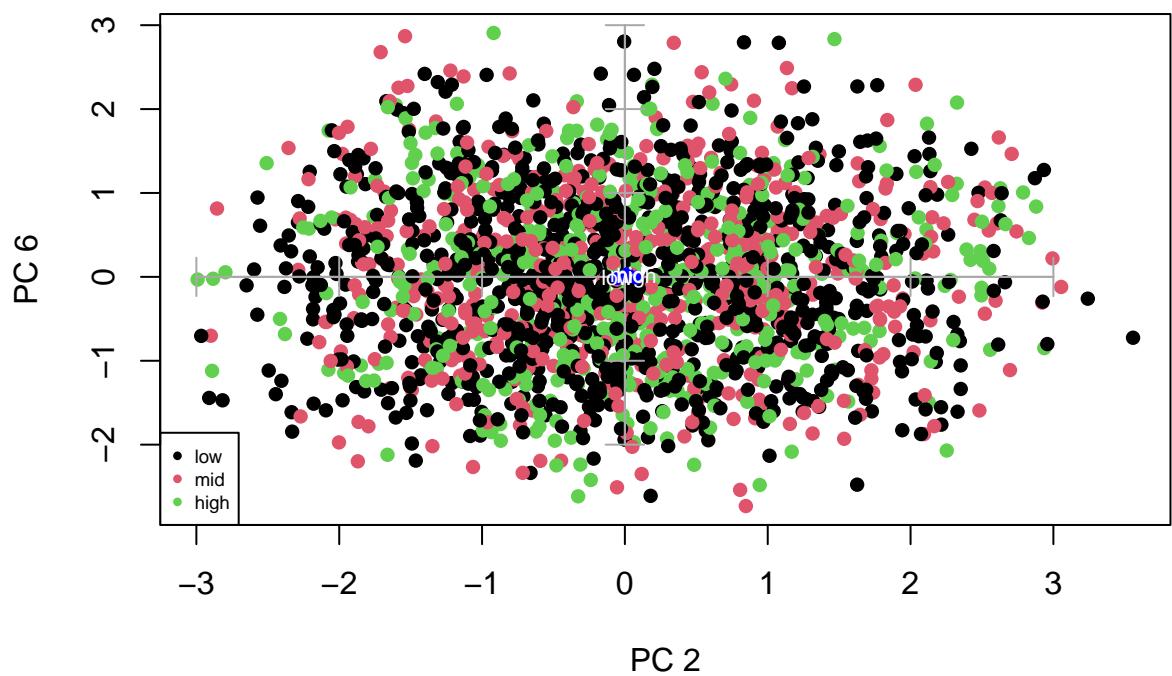


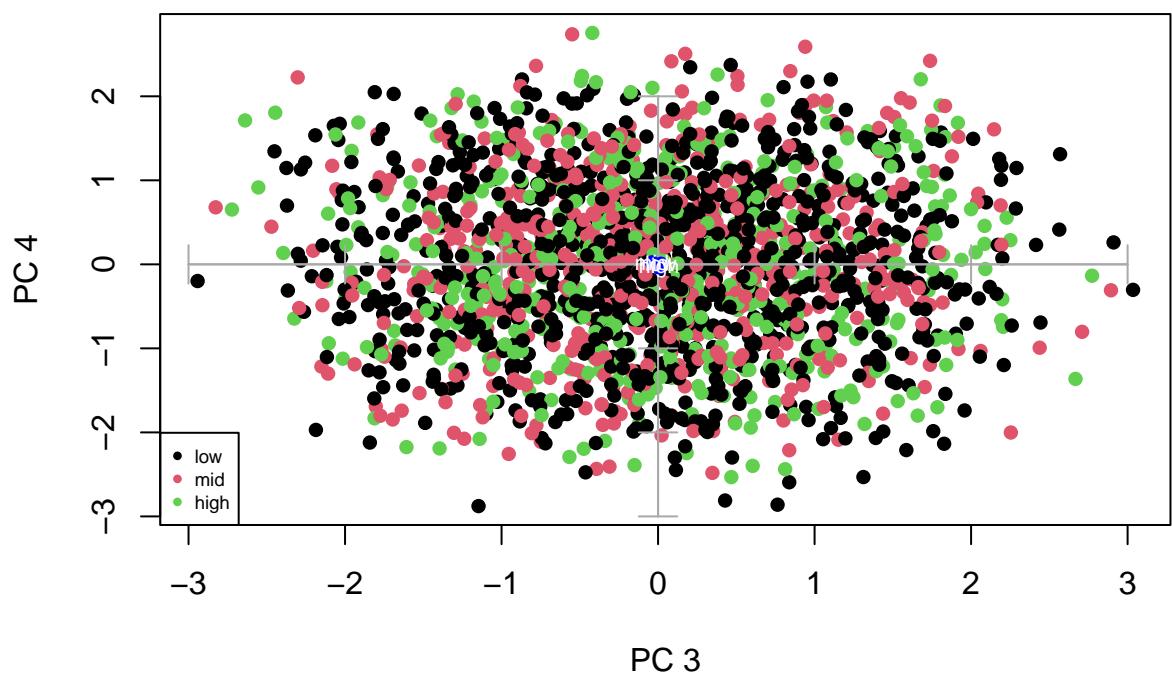
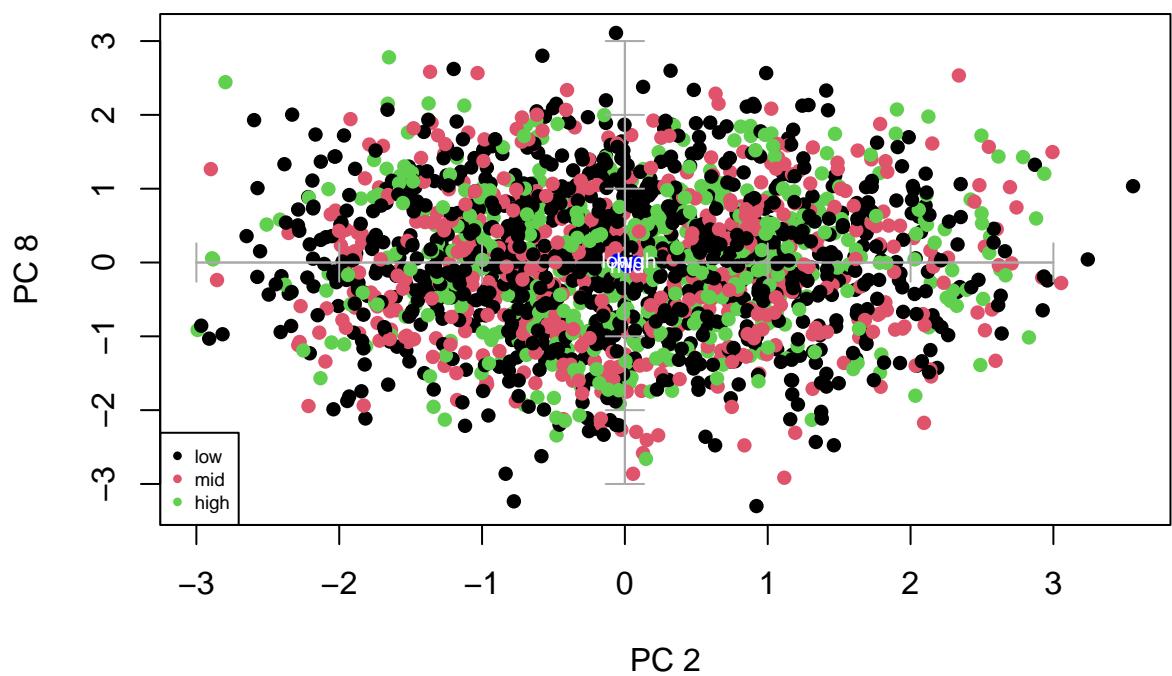


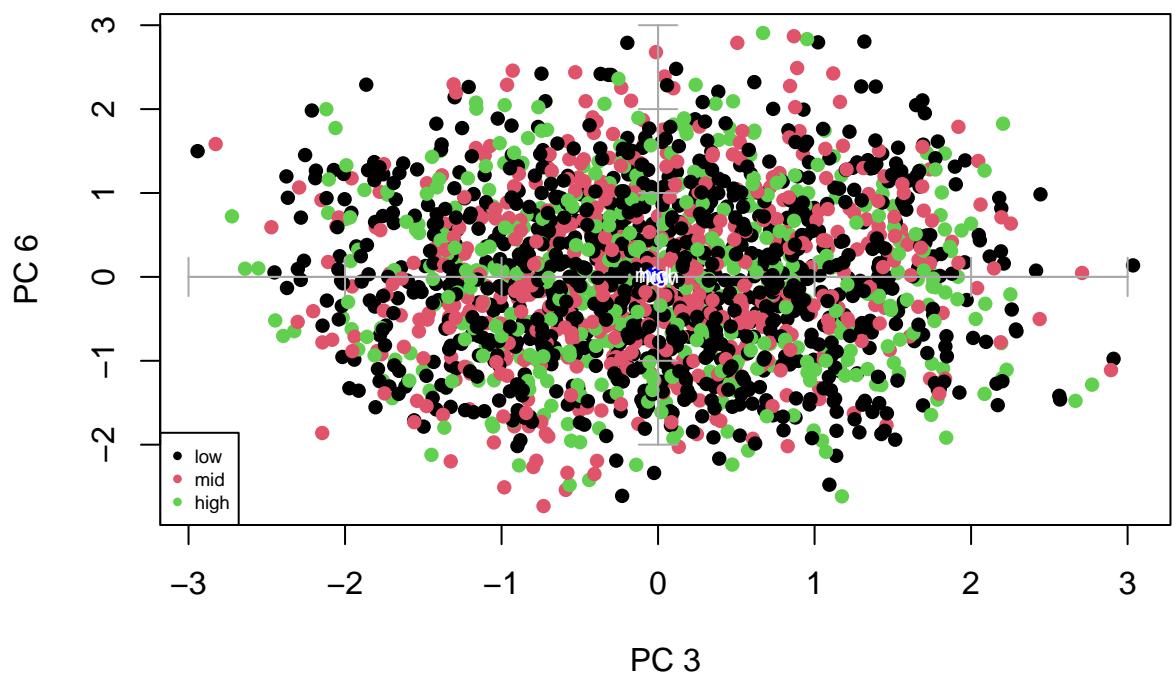
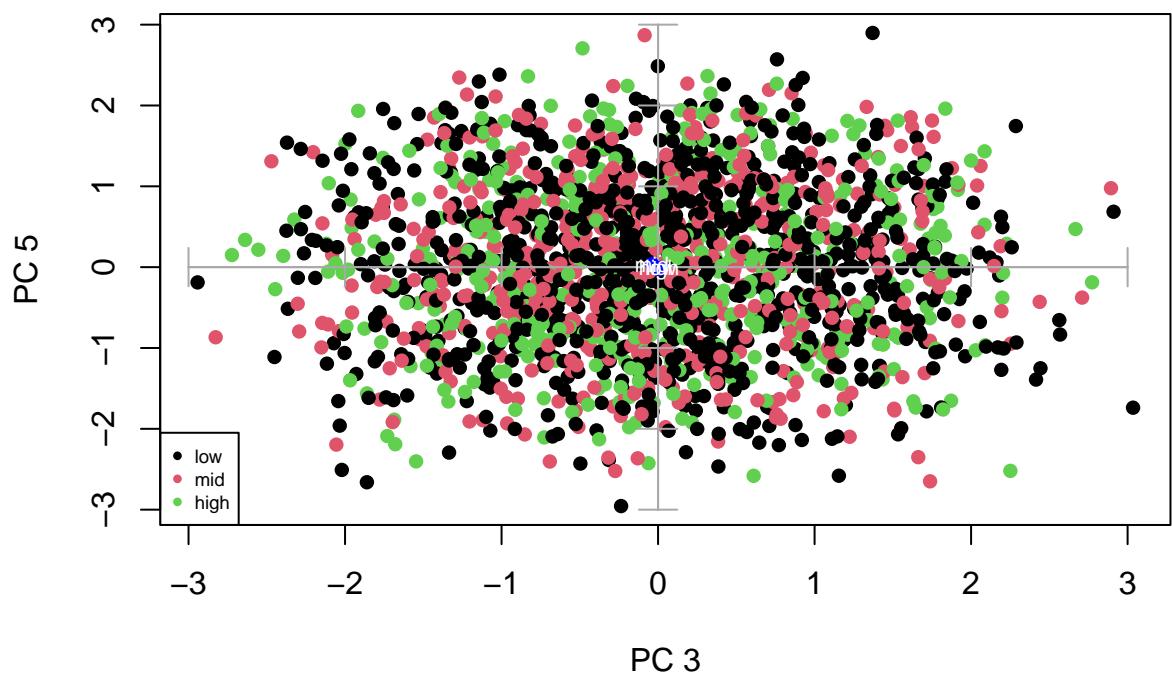


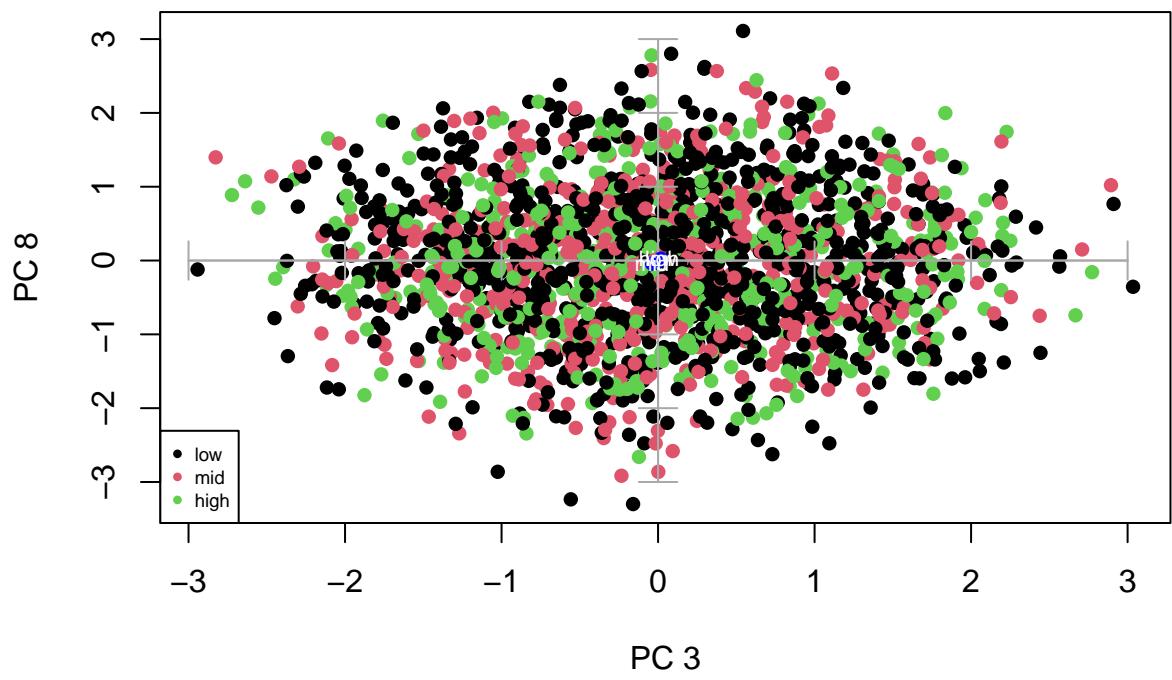
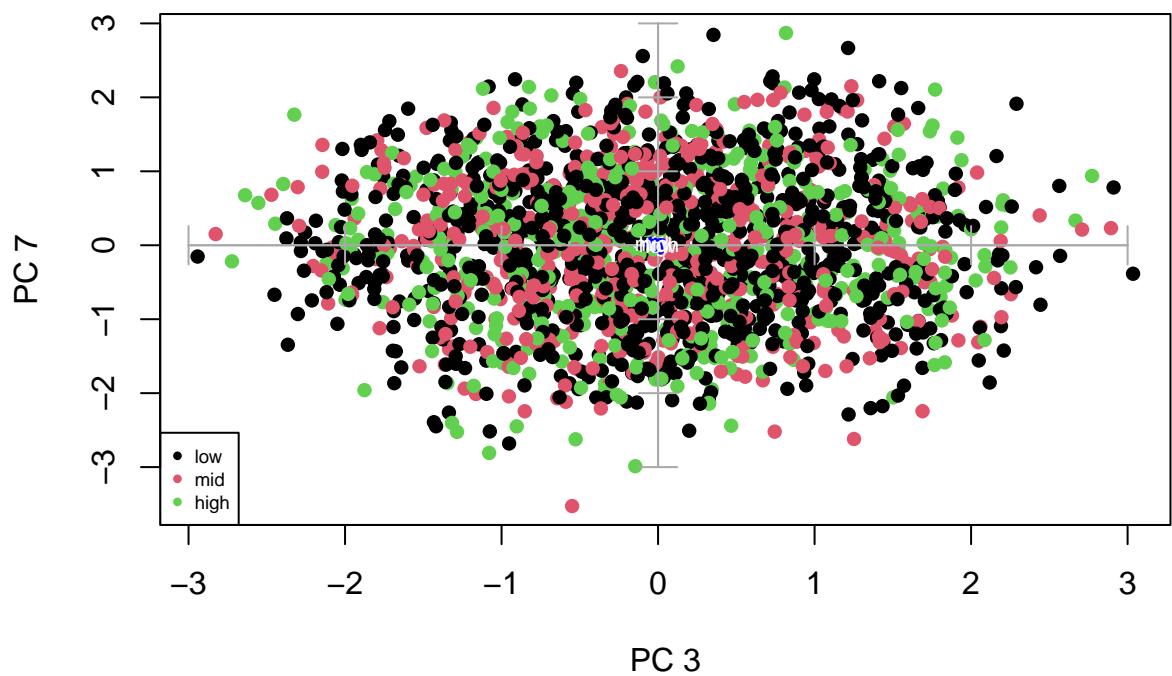


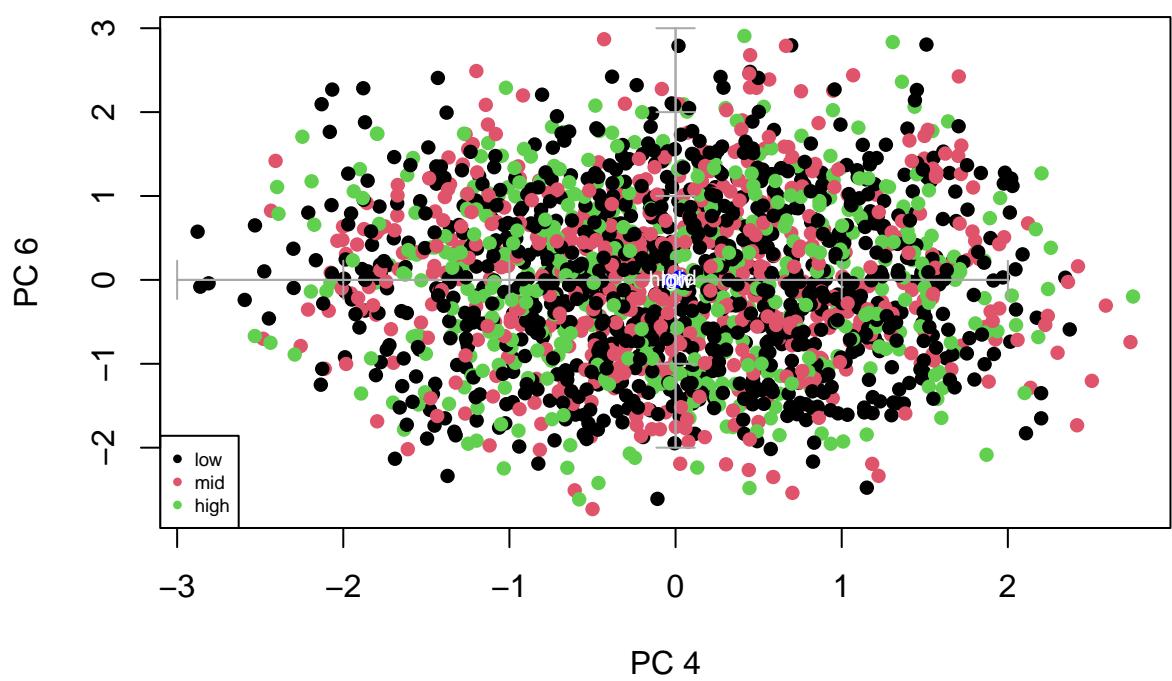
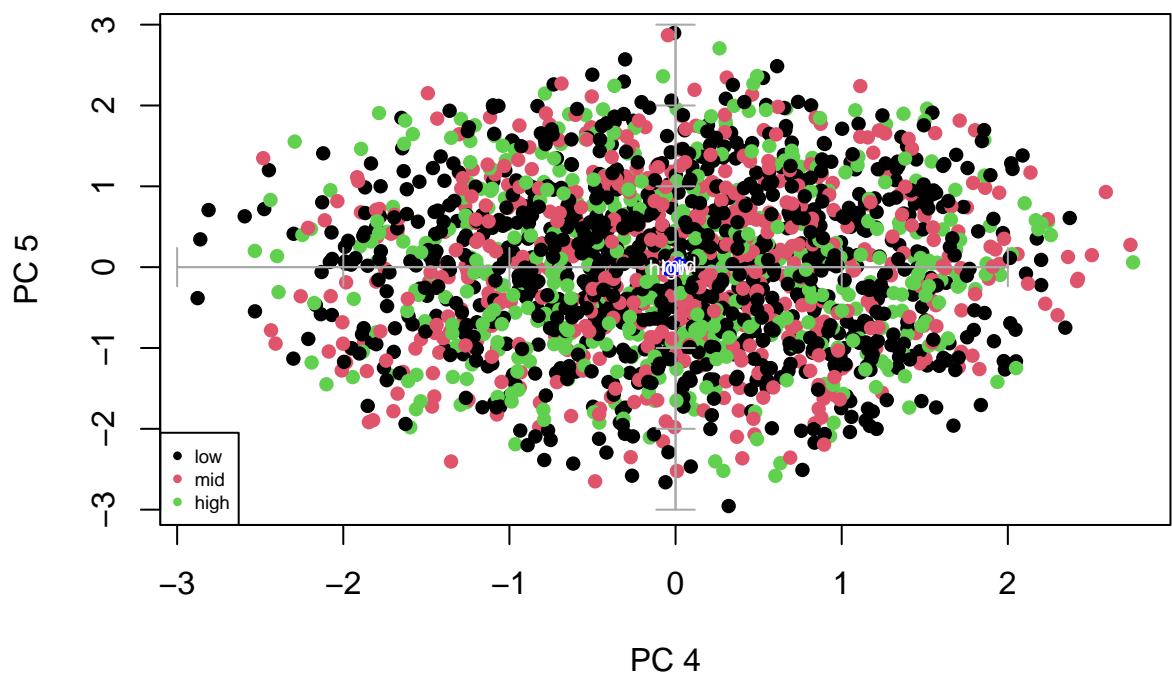


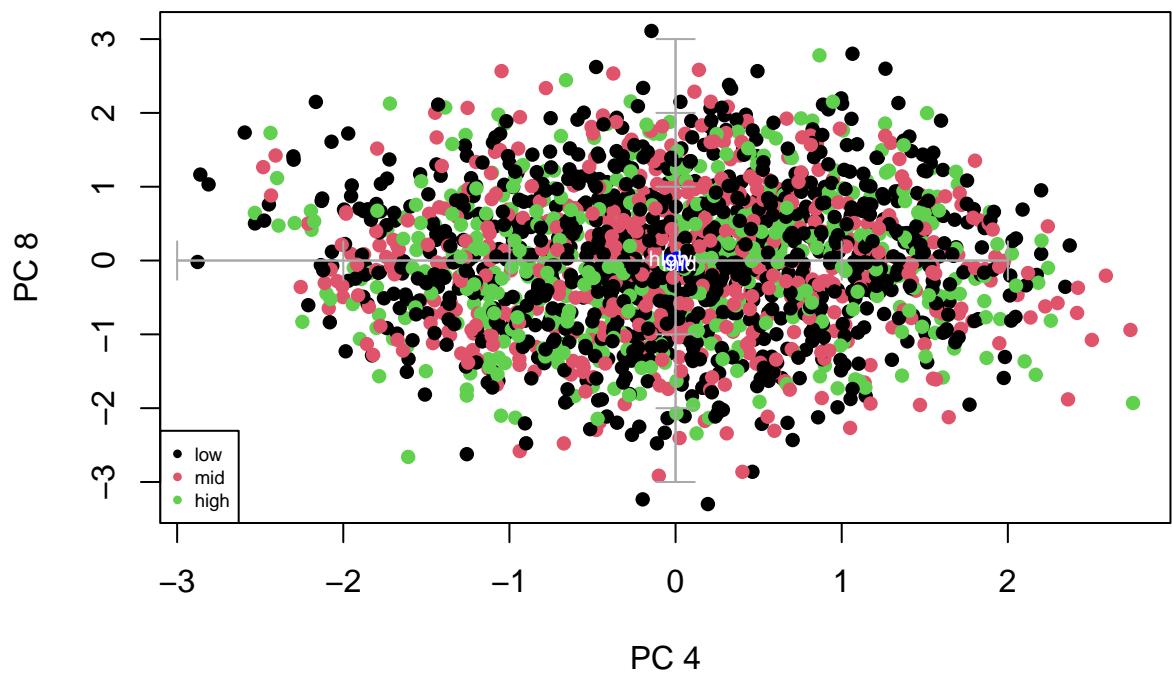
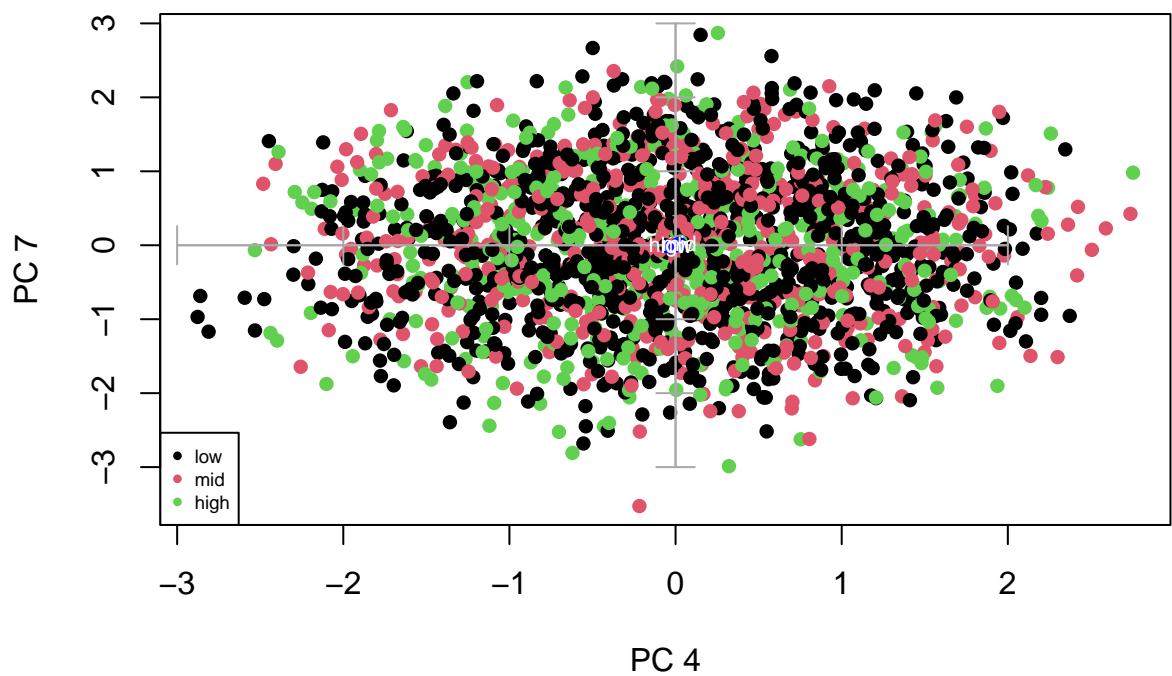


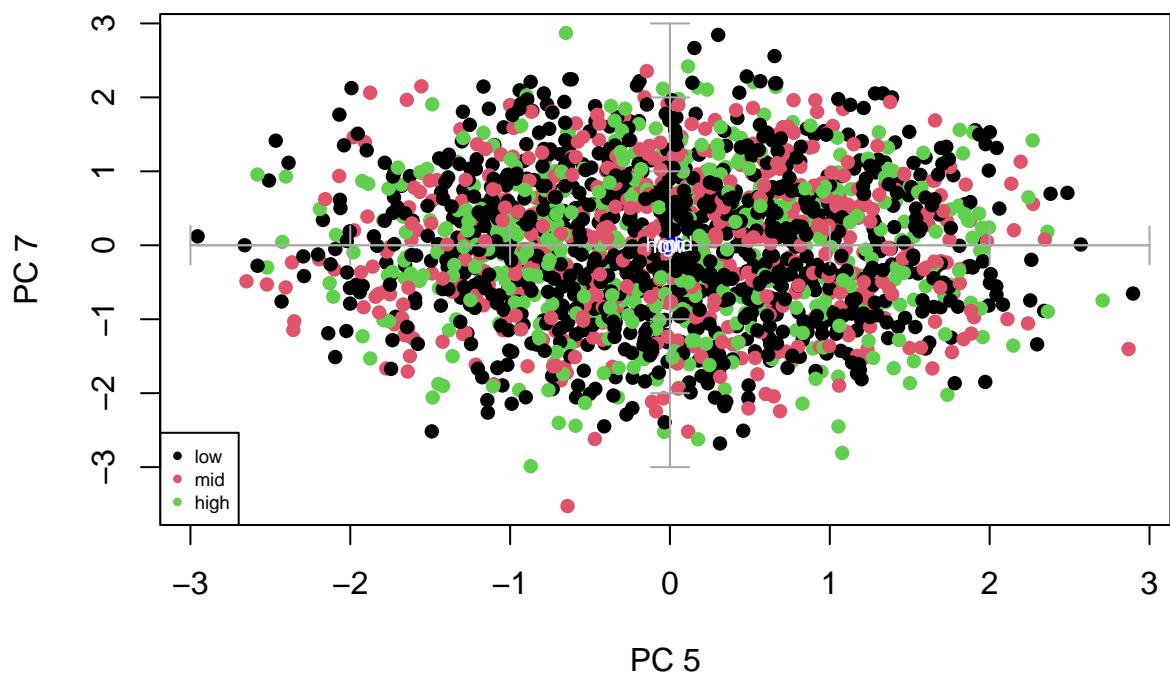
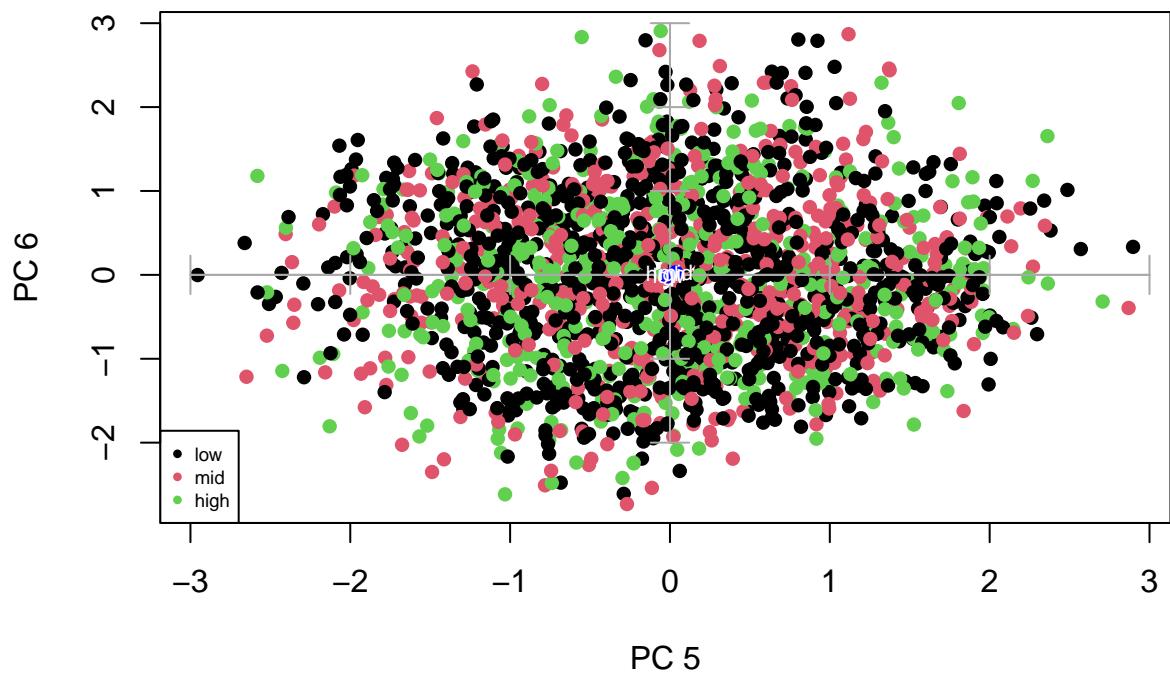


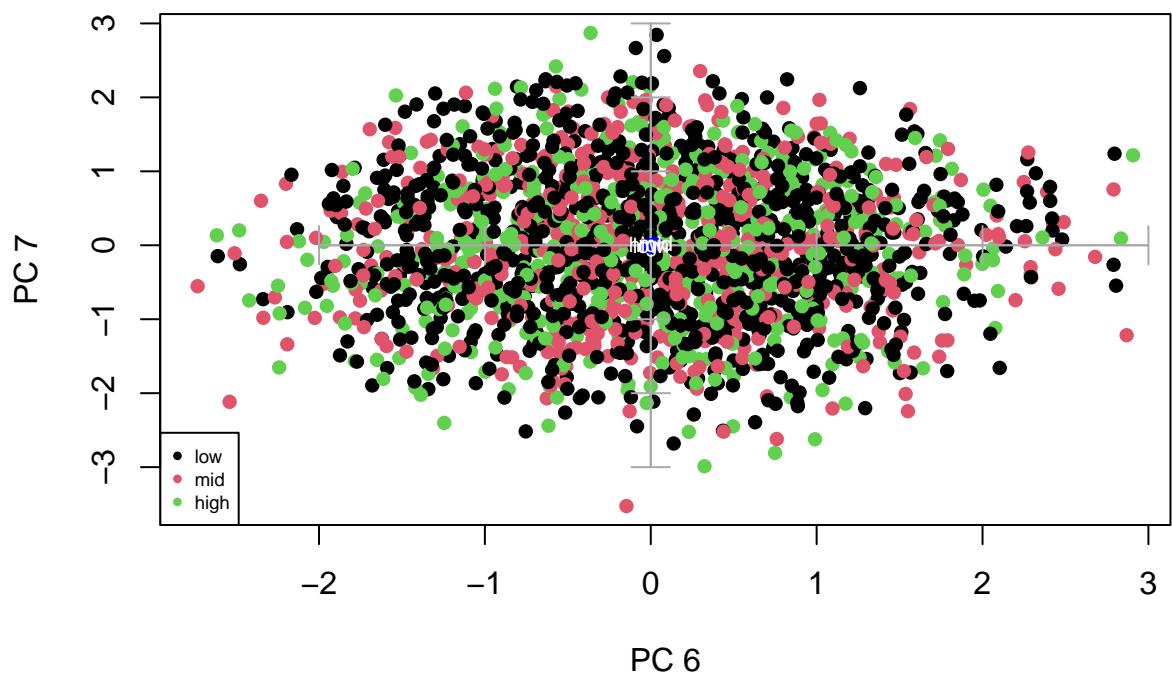
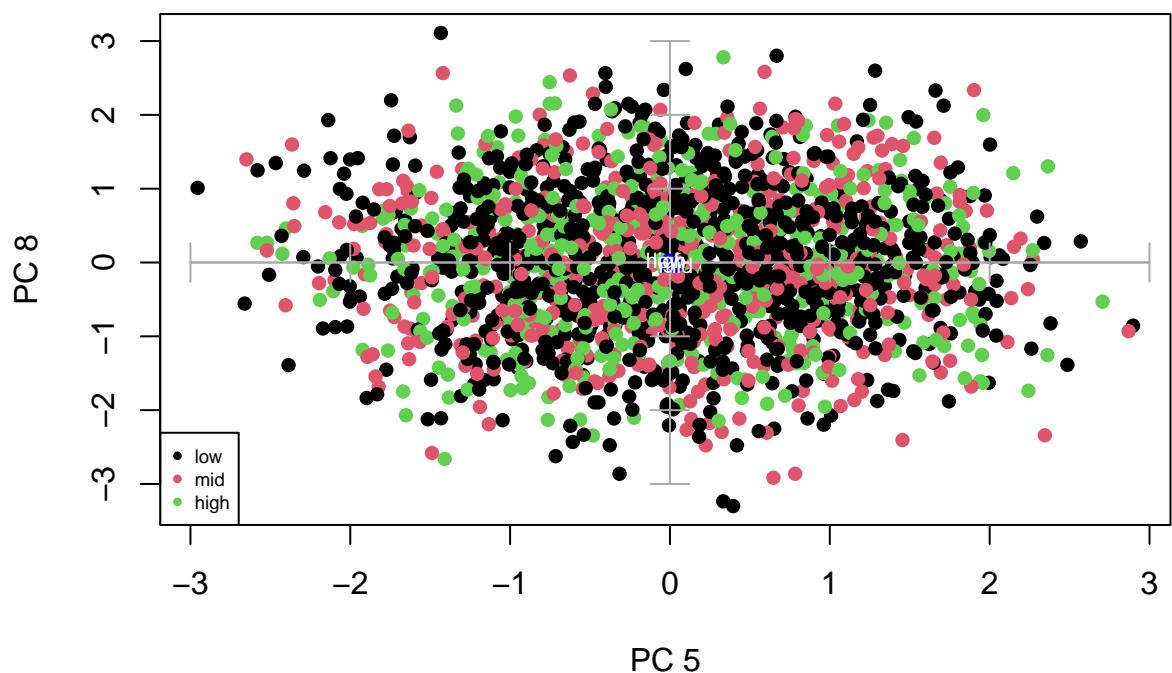


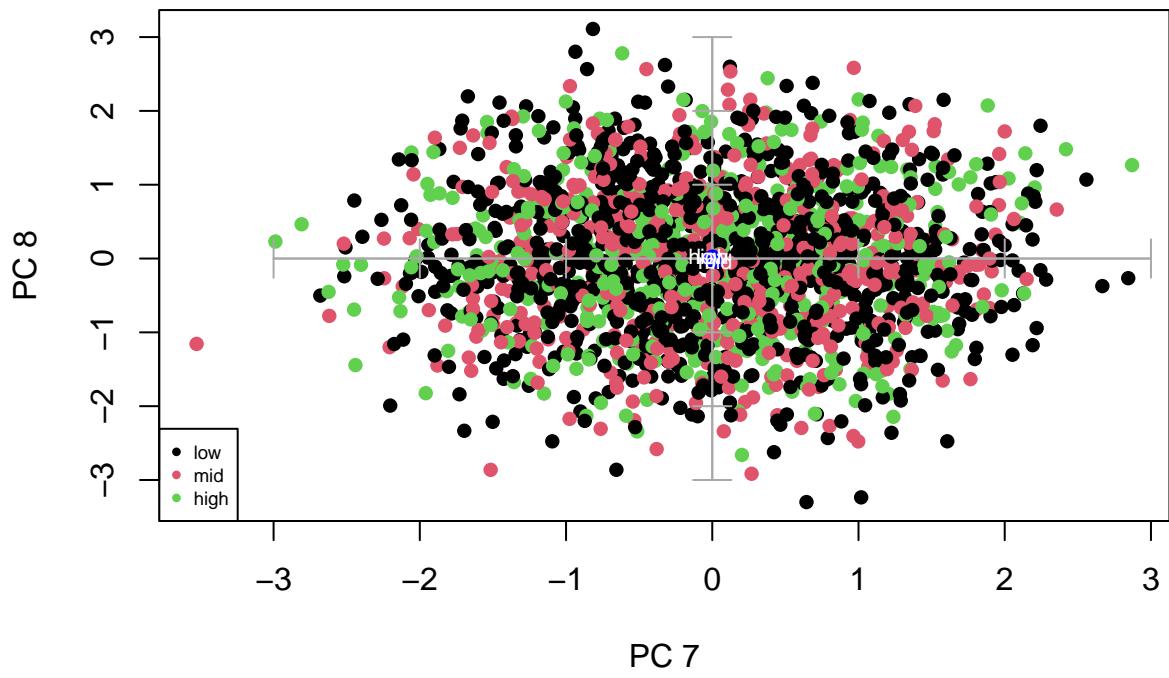
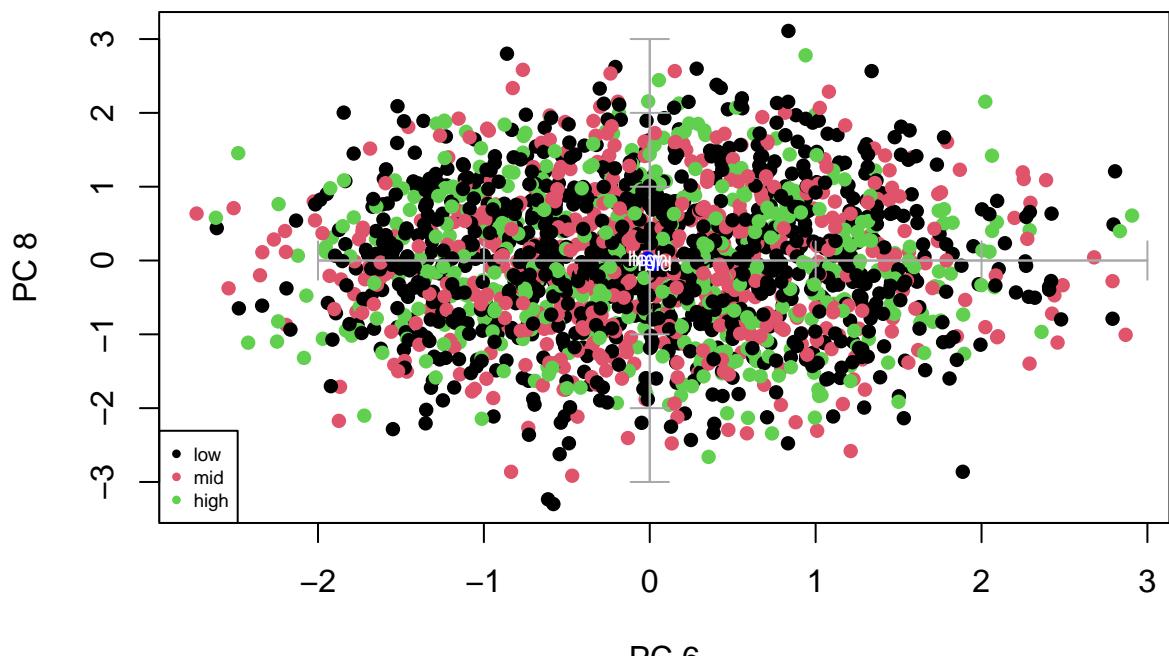












```

for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$pc
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
  }
}
  
```

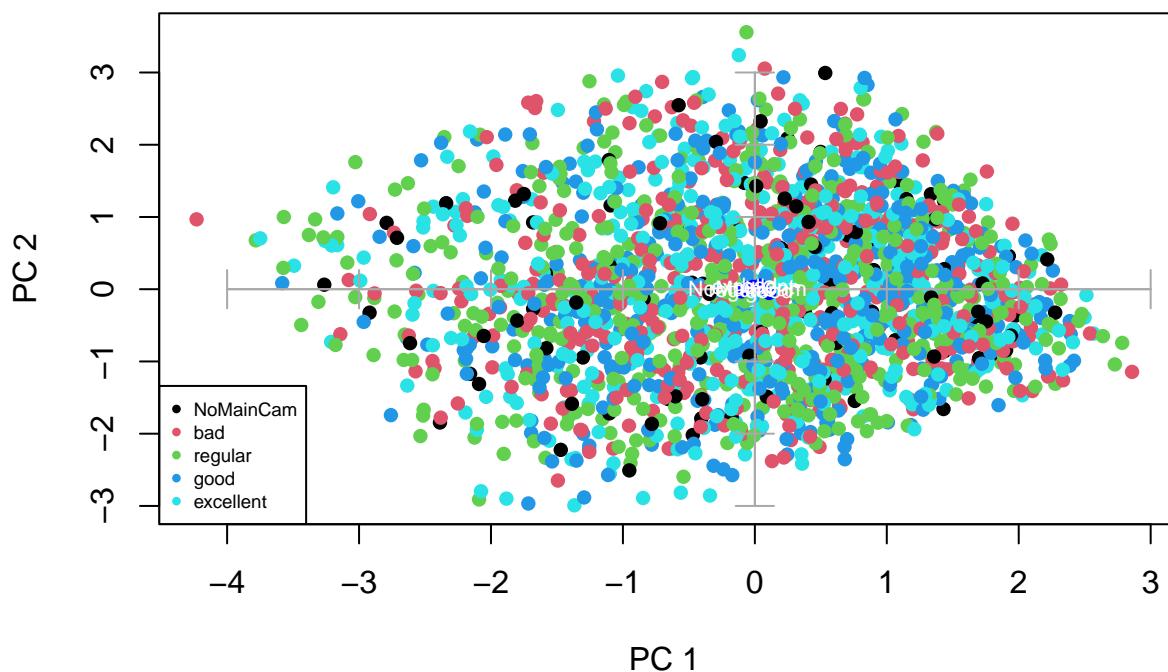
```

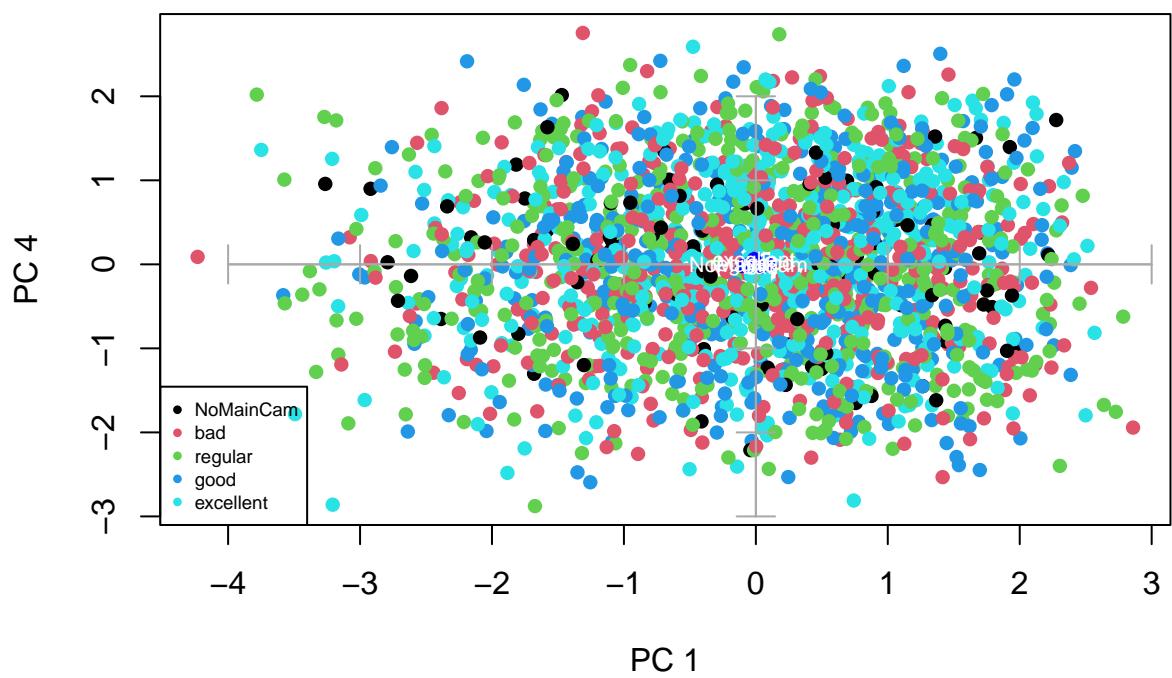
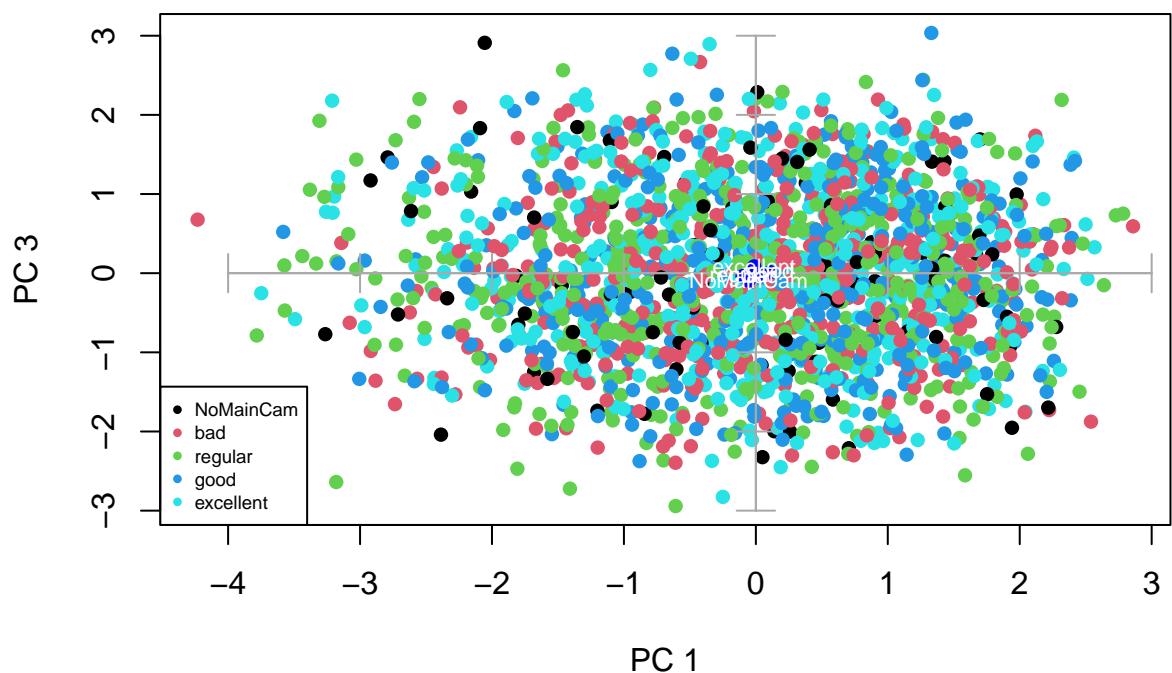
legend("bottomleft",levels(varcat),pch=16,col=c(1:5), cex=0.6)

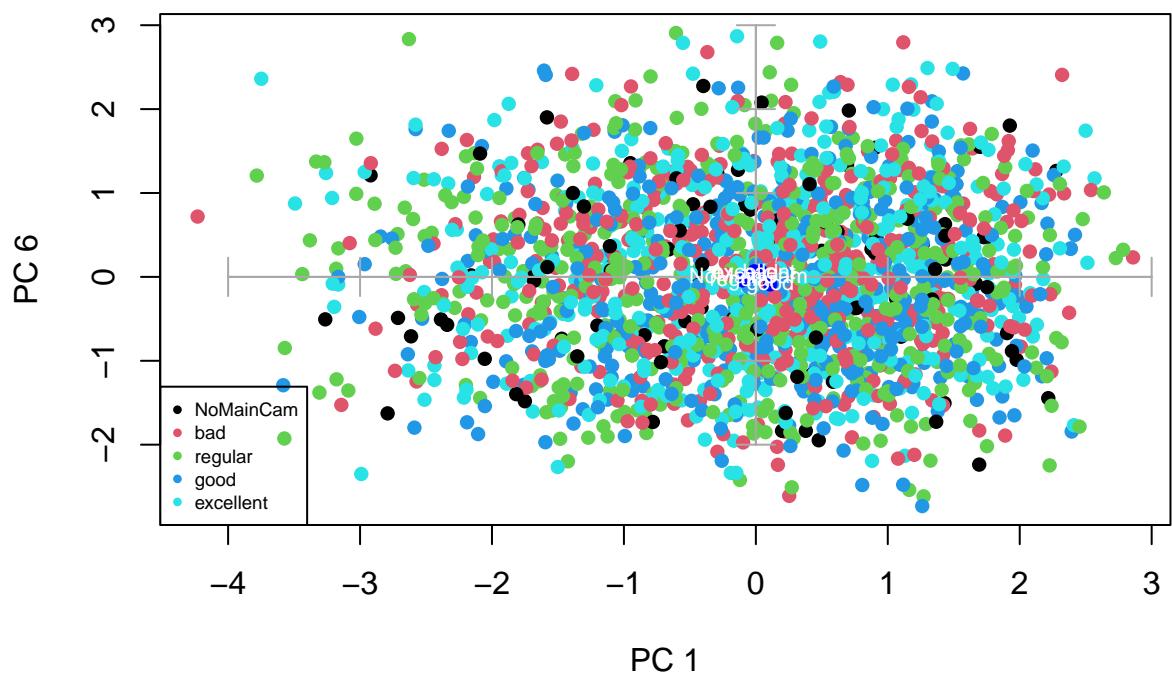
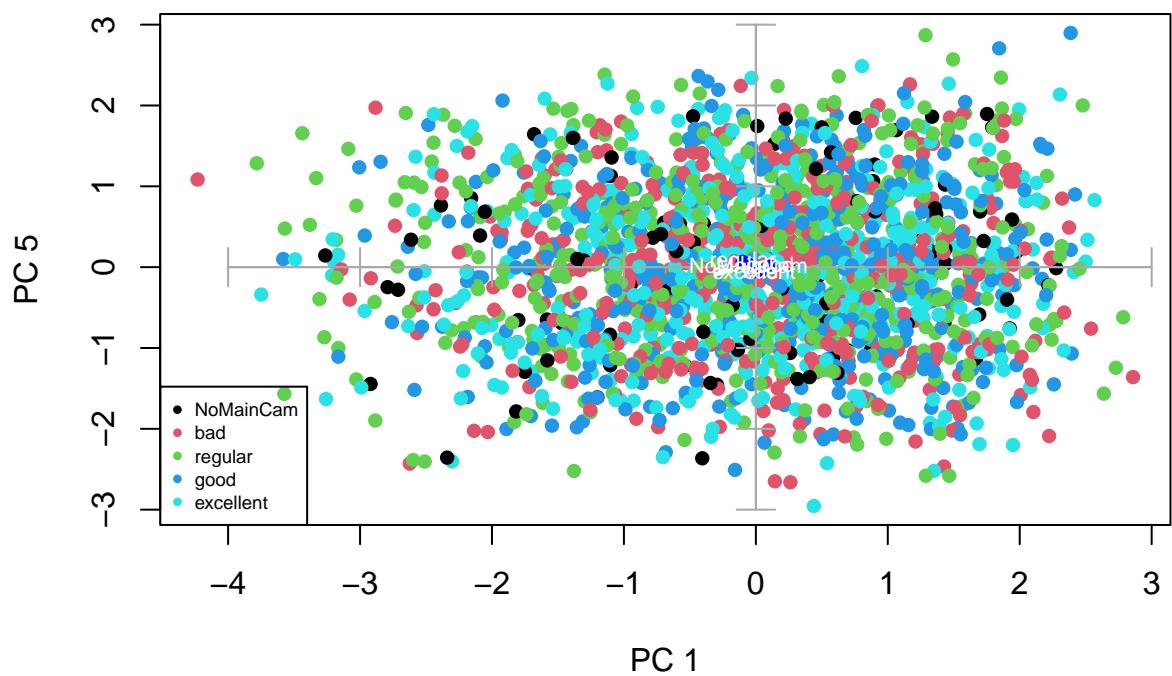
#select your qualitative variable
fdic1 = tapply(Psi[,i],varcat,mean)
fdic2 = tapply(Psi[,j],varcat,mean)
points(fdic1,fdic2,pch=16,col="blue")
text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
}

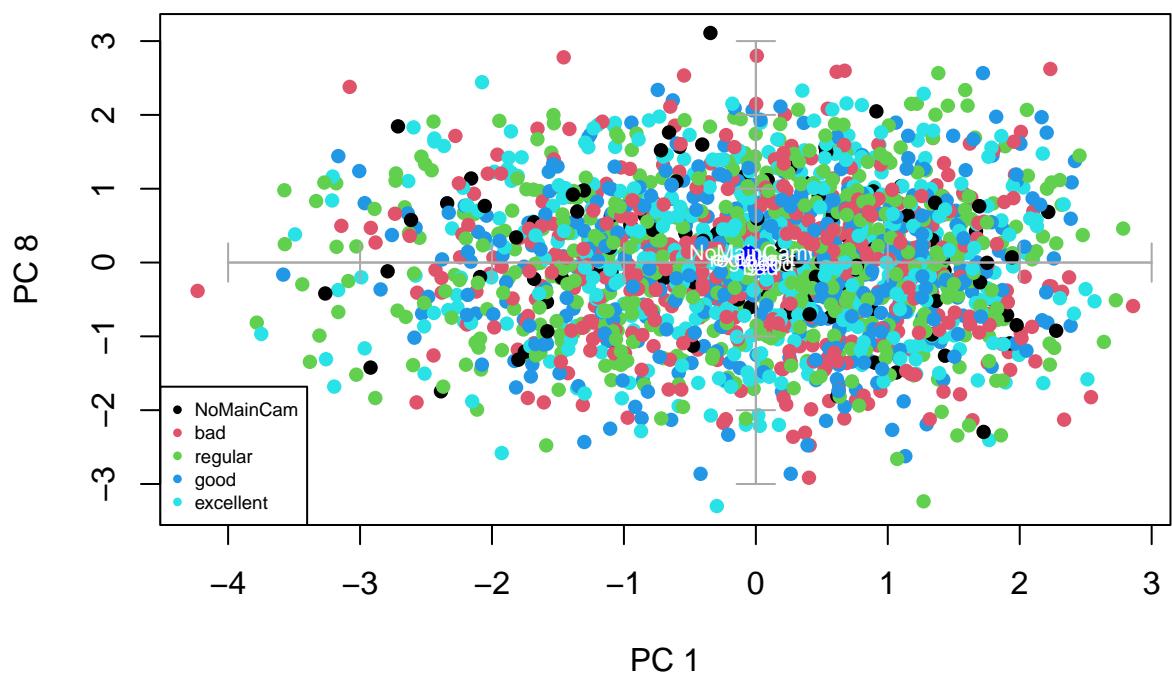
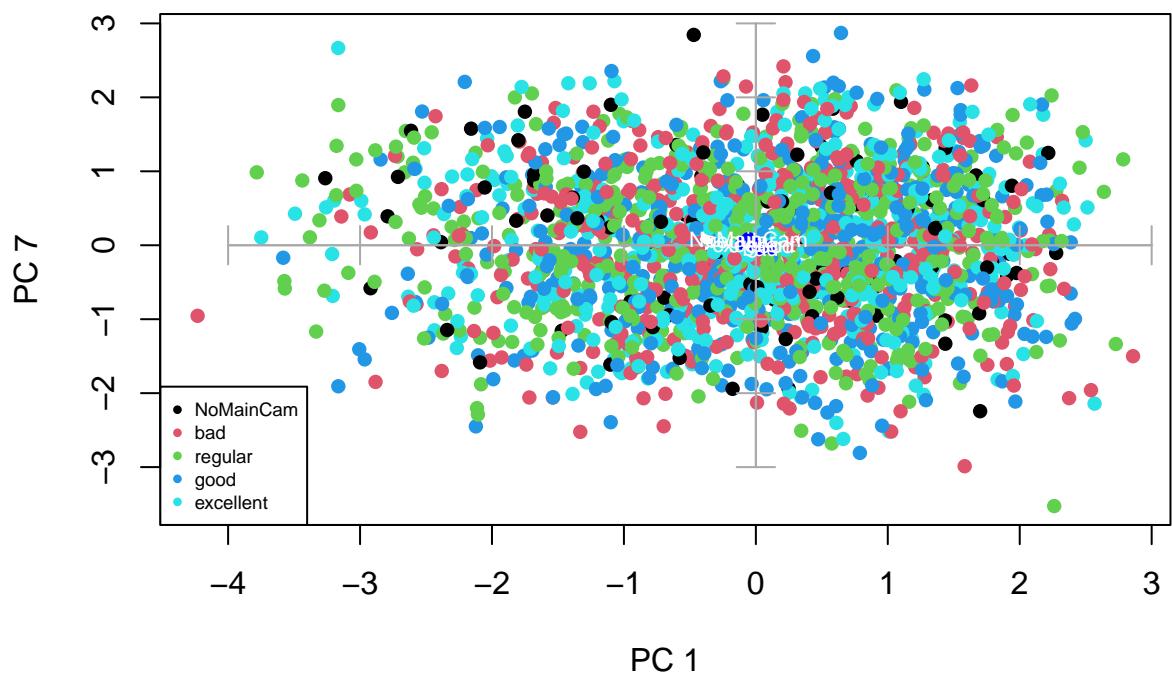
}

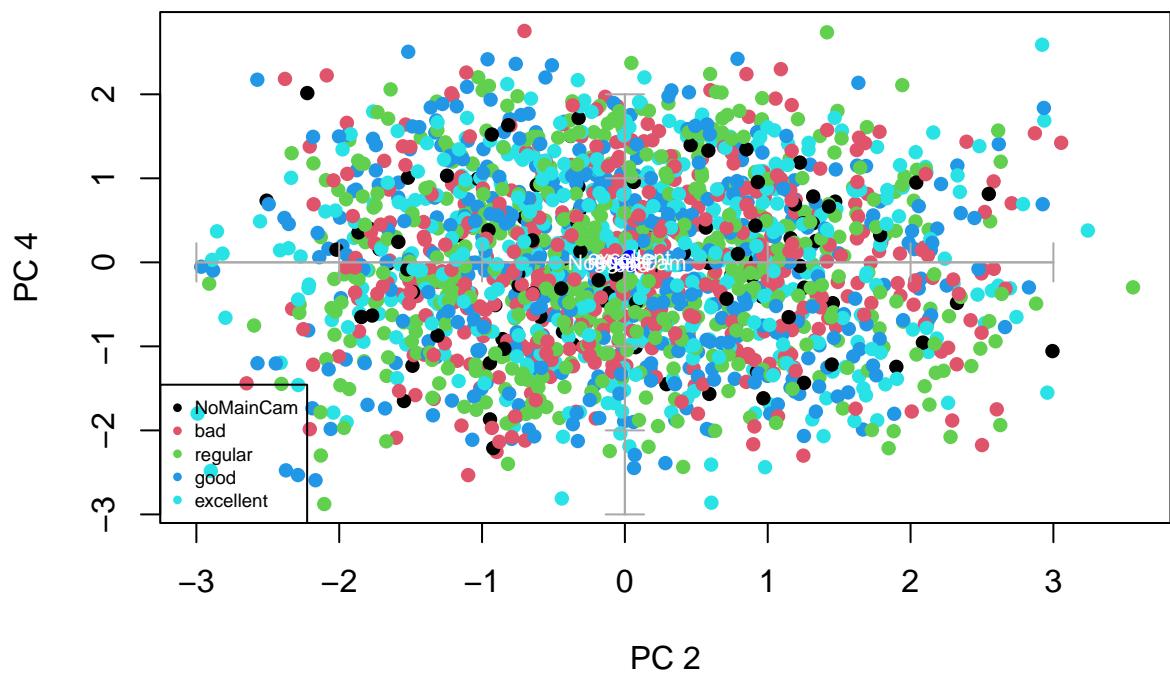
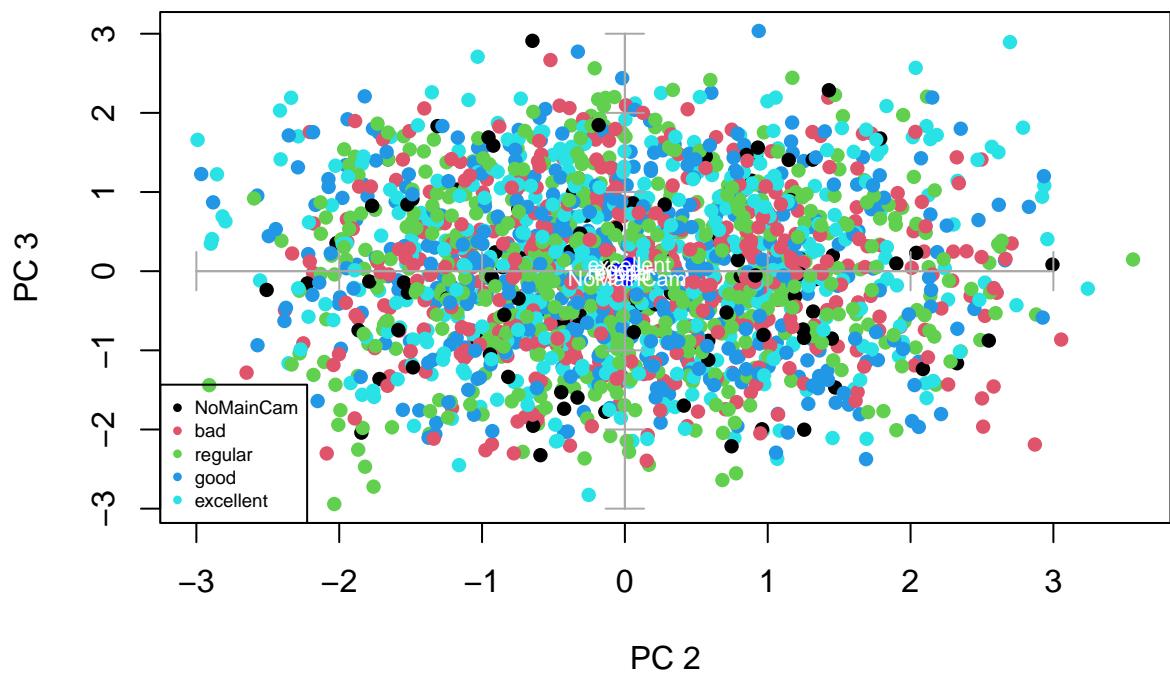
```

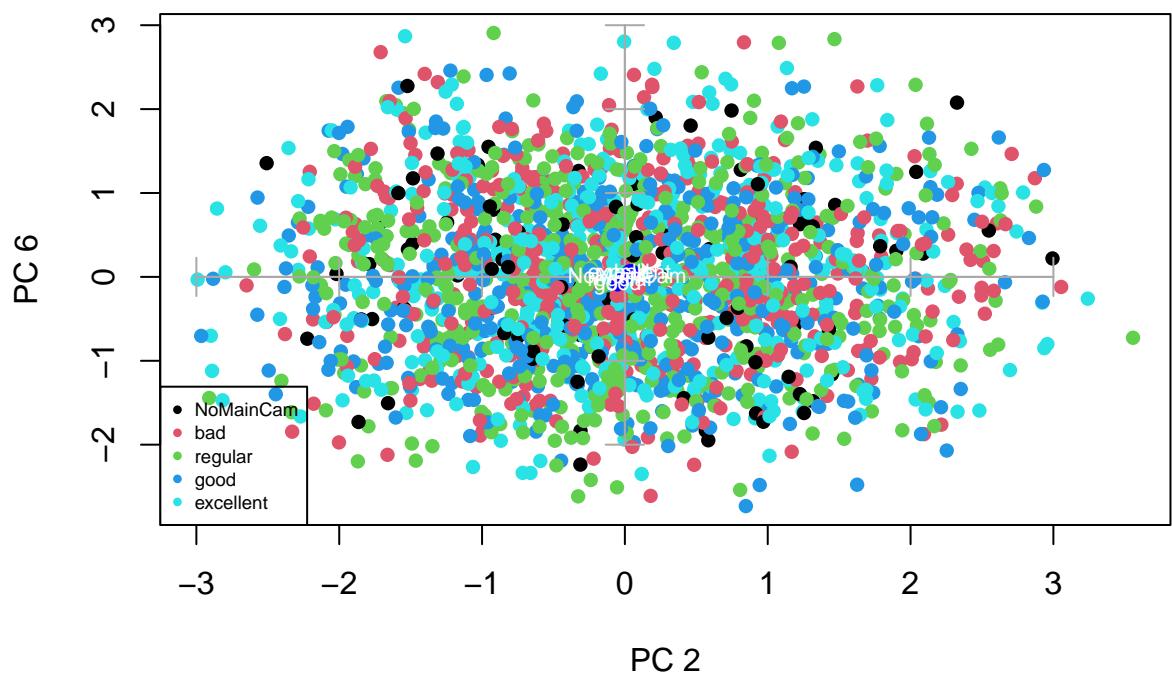
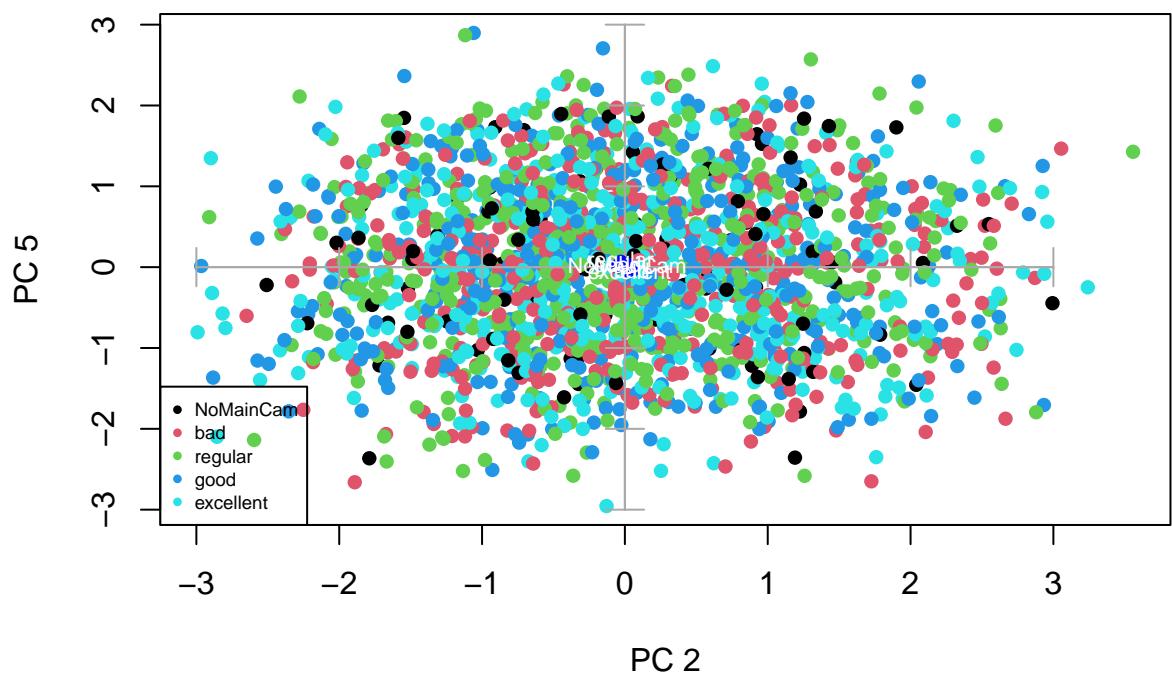


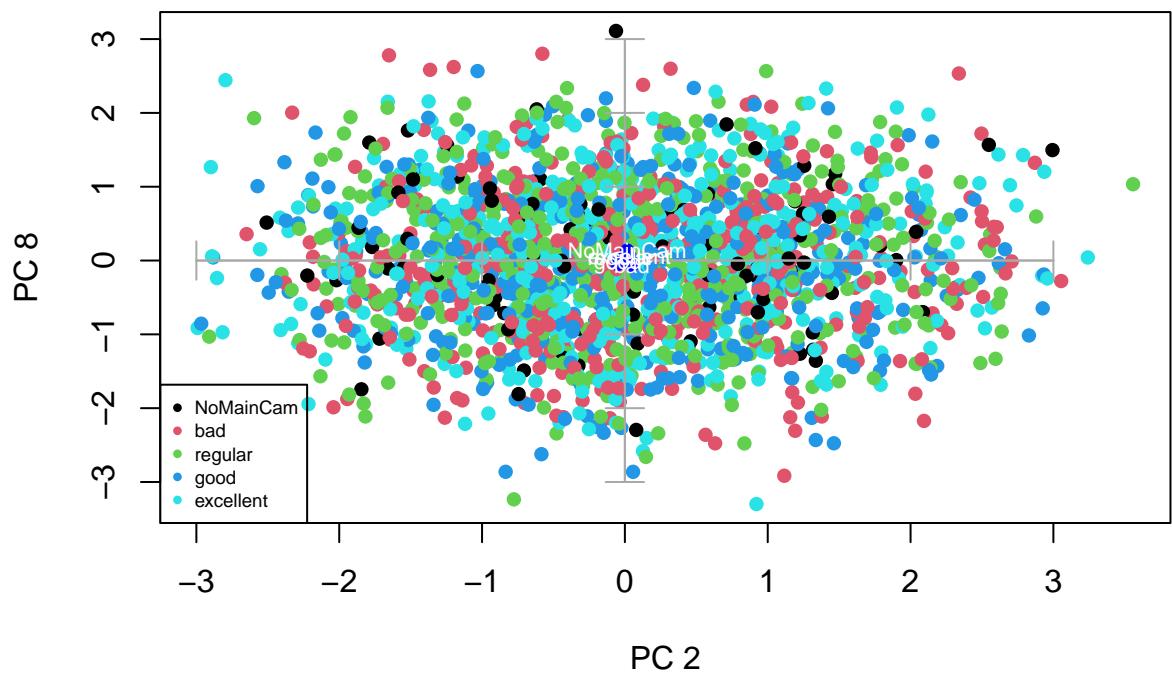
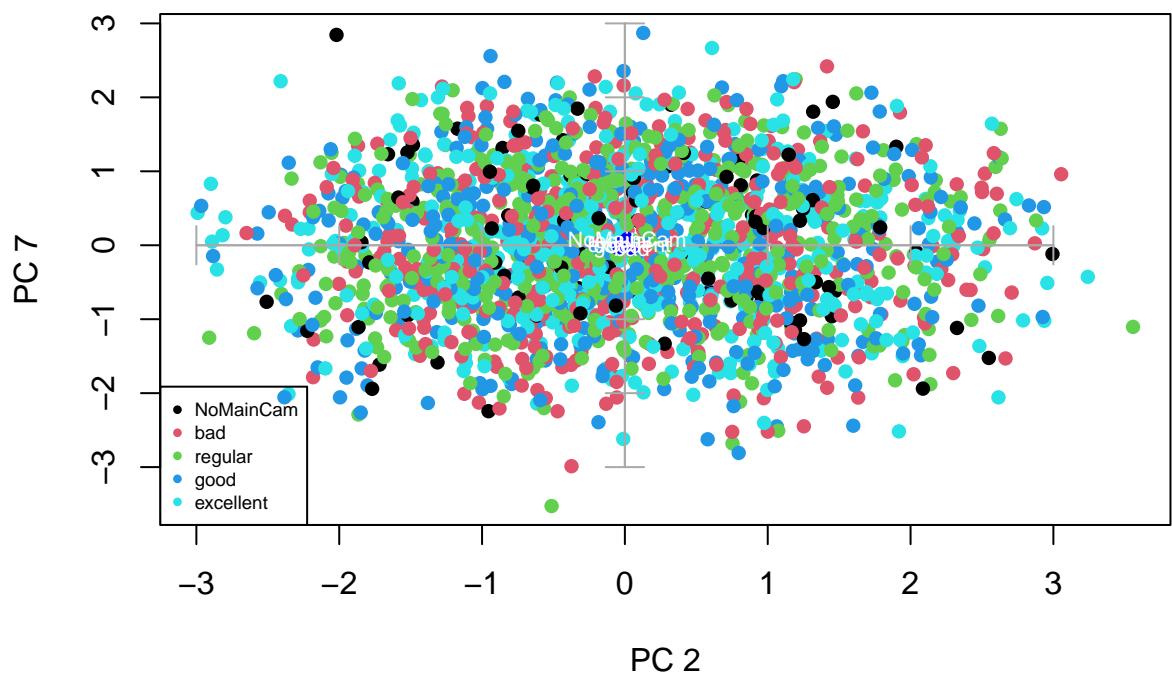


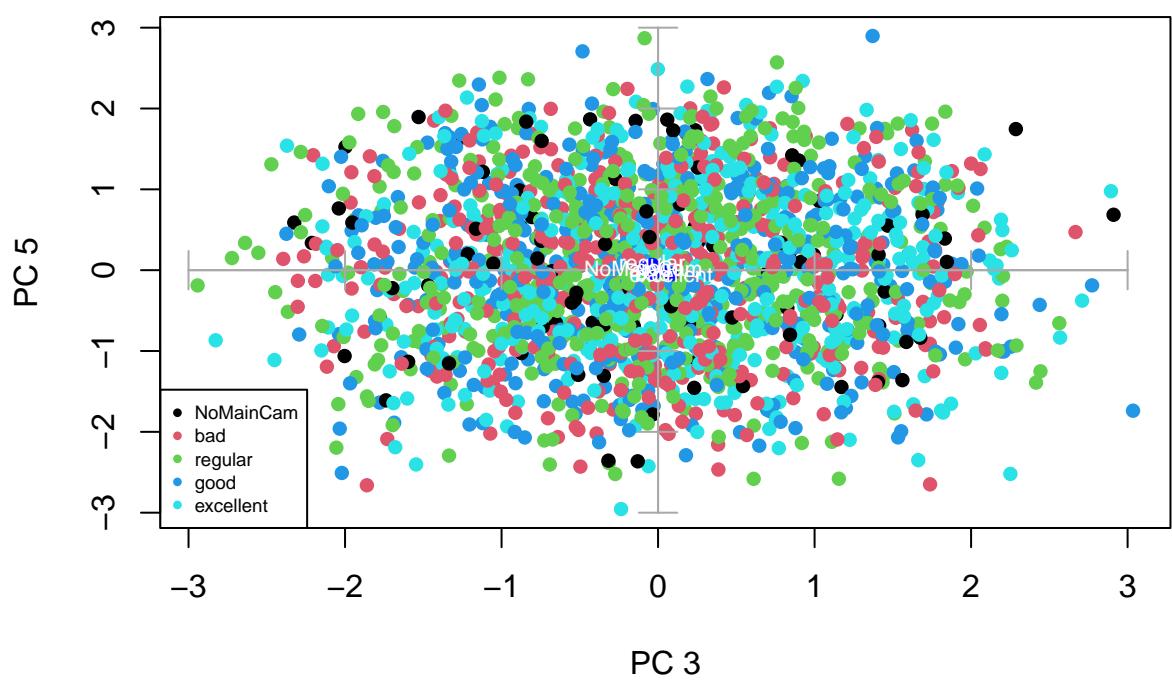
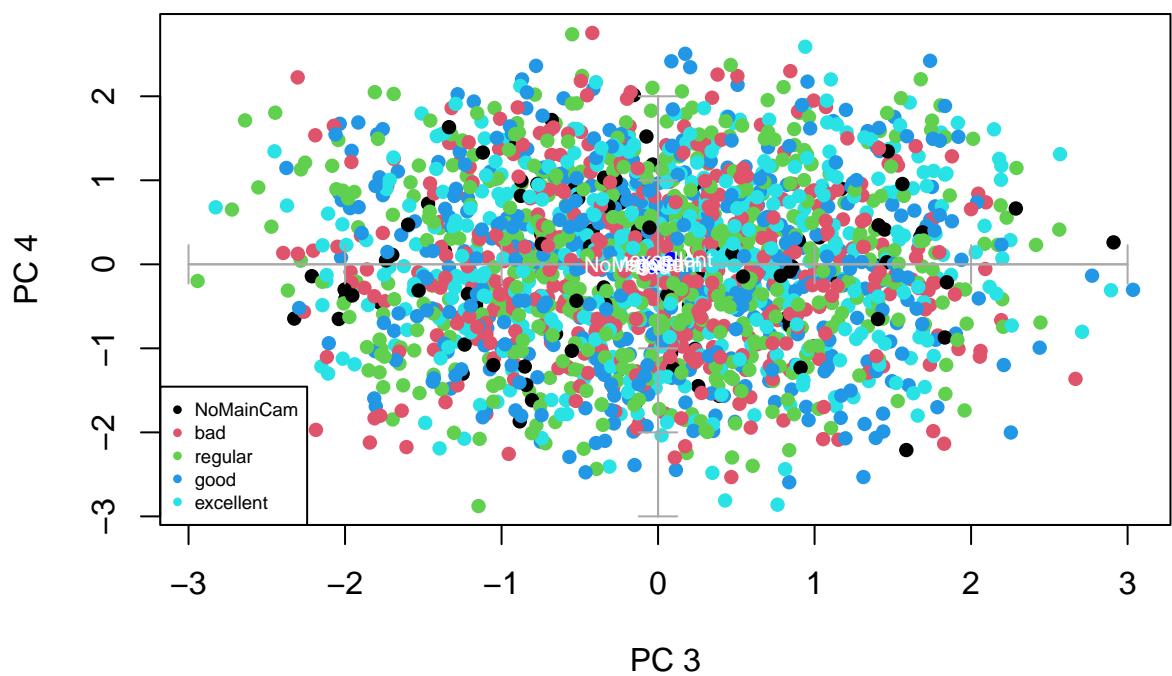


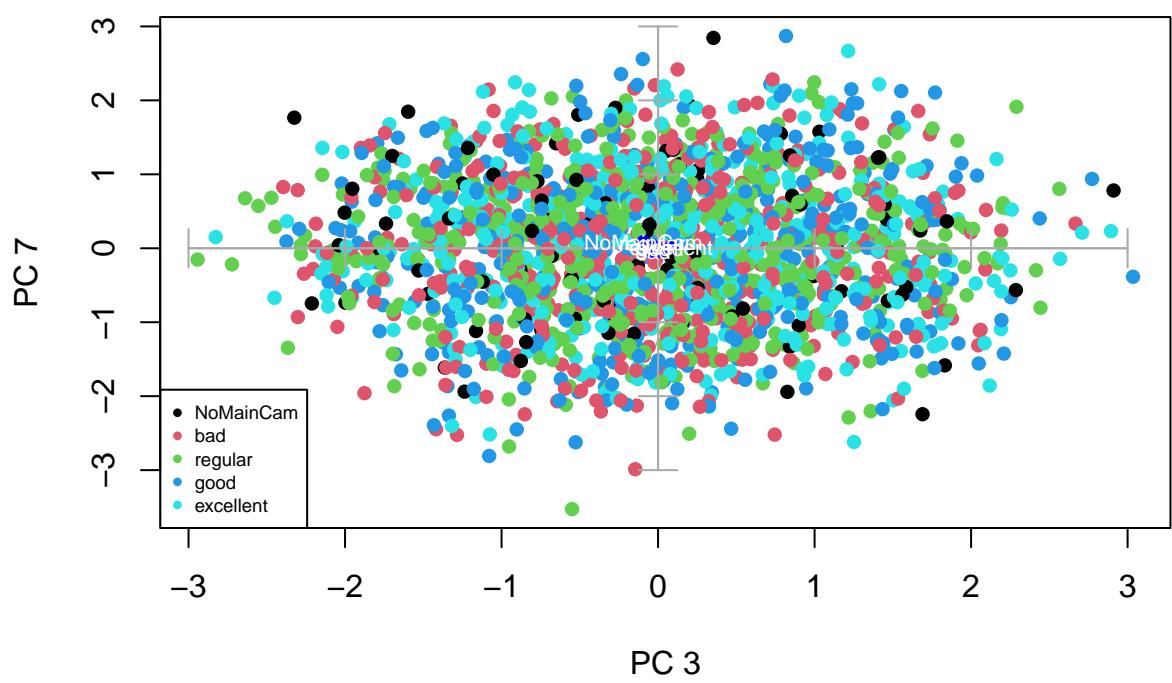
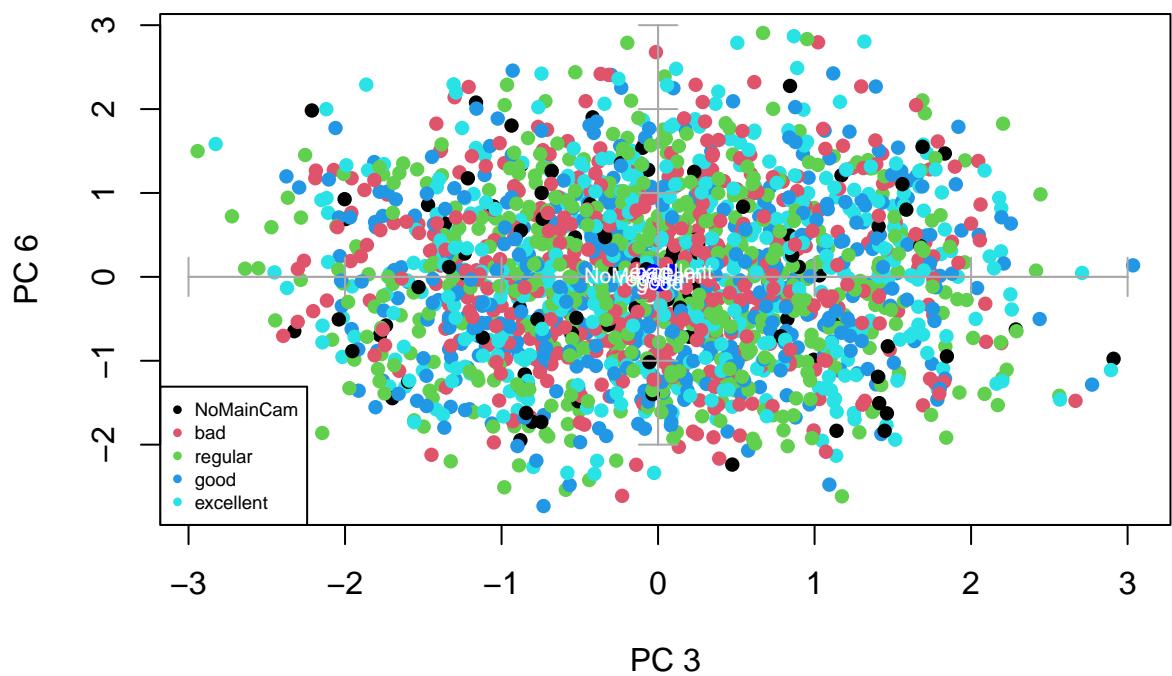


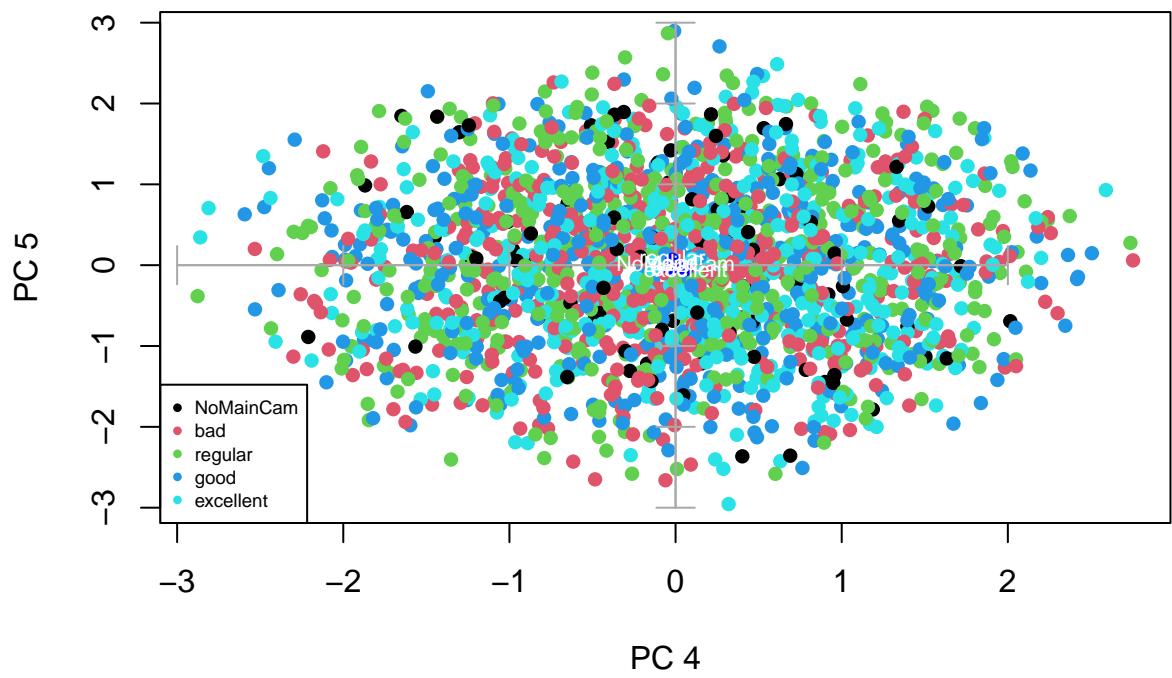
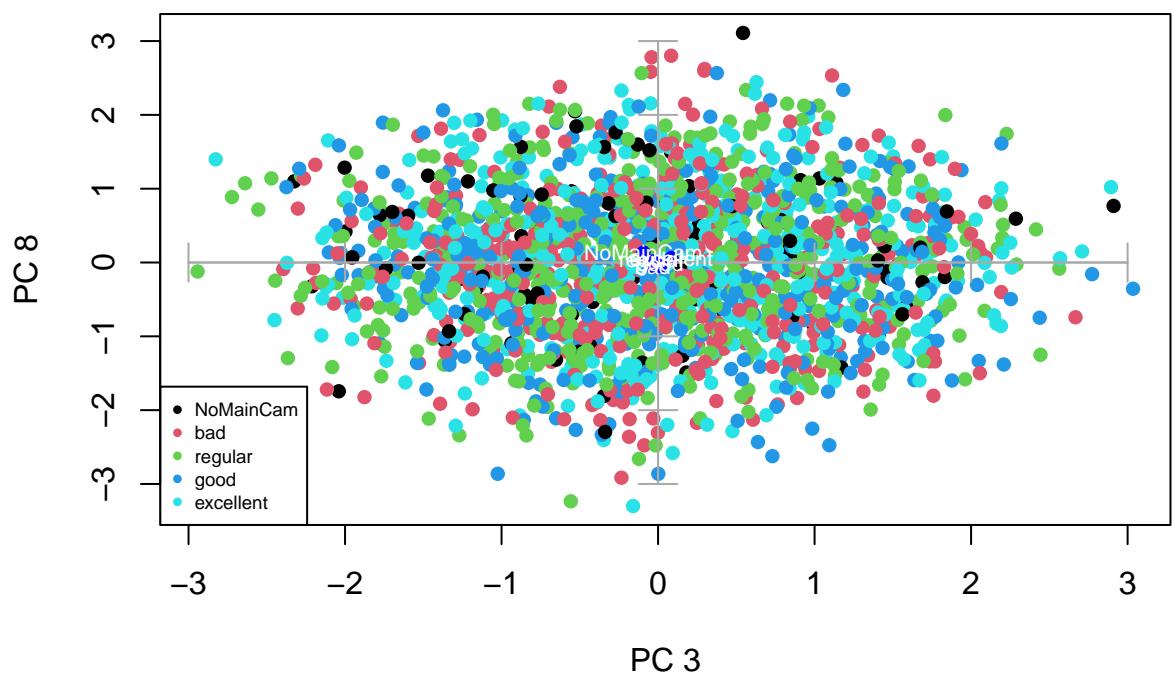


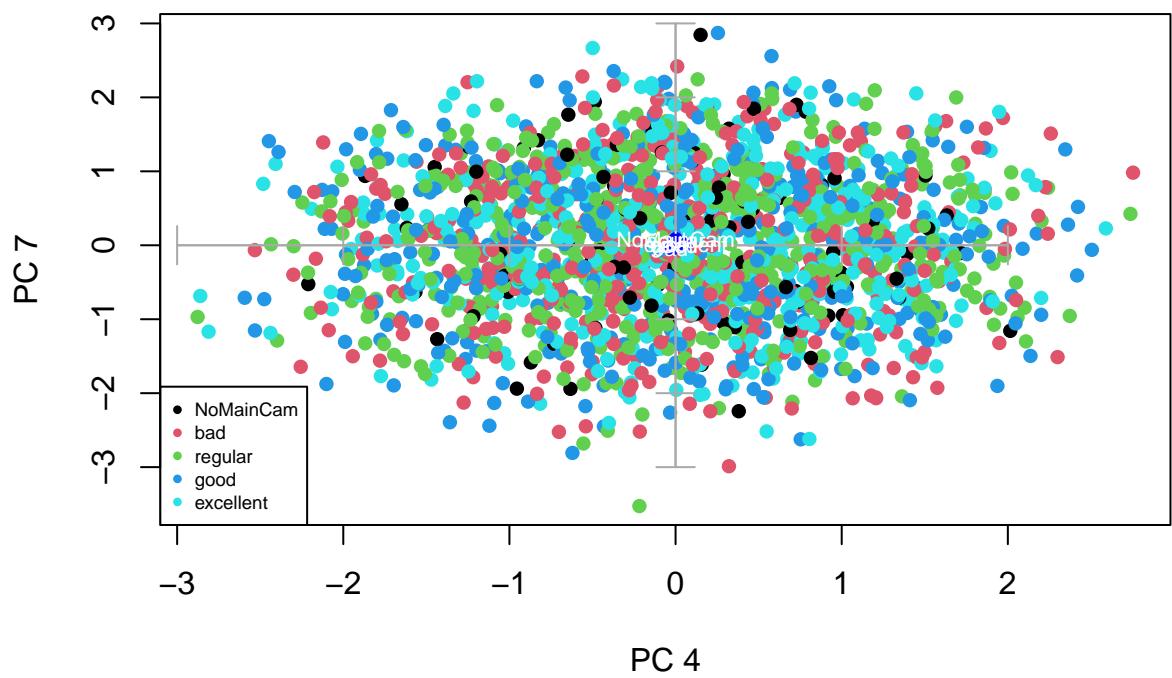
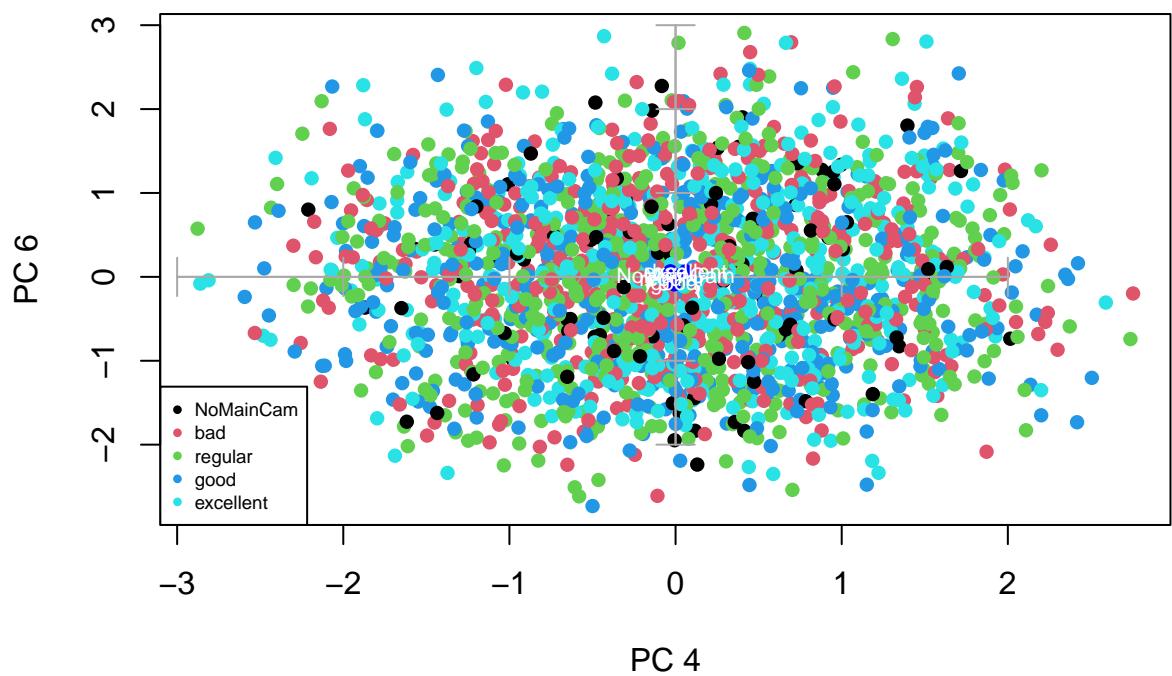


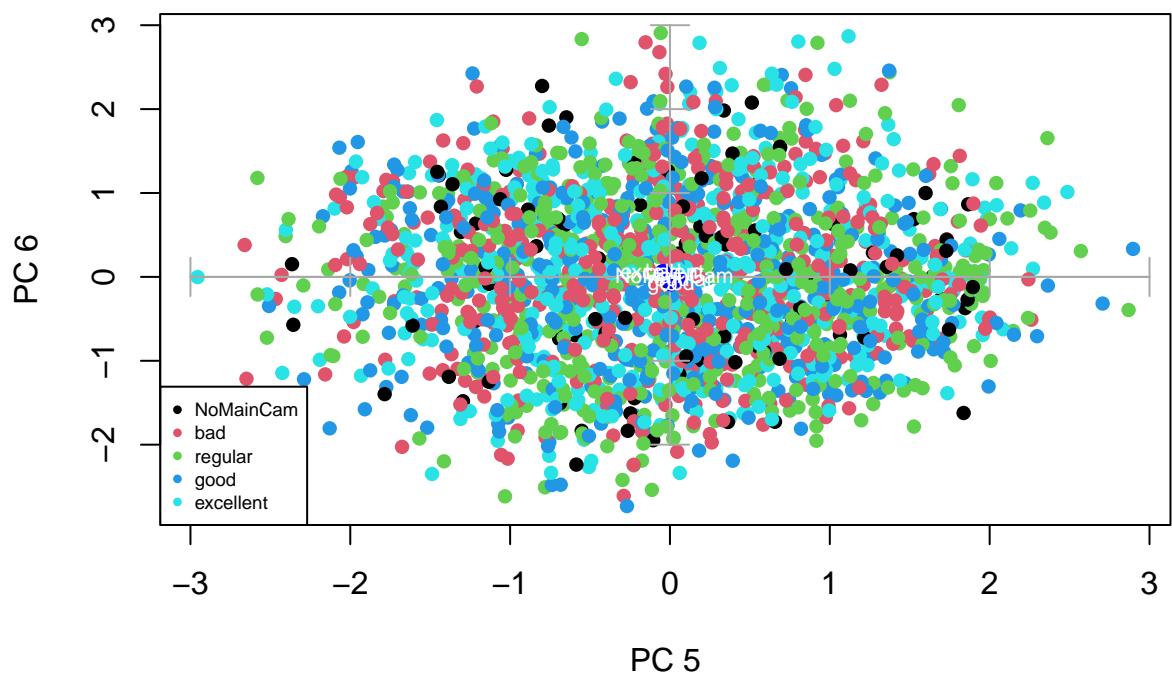
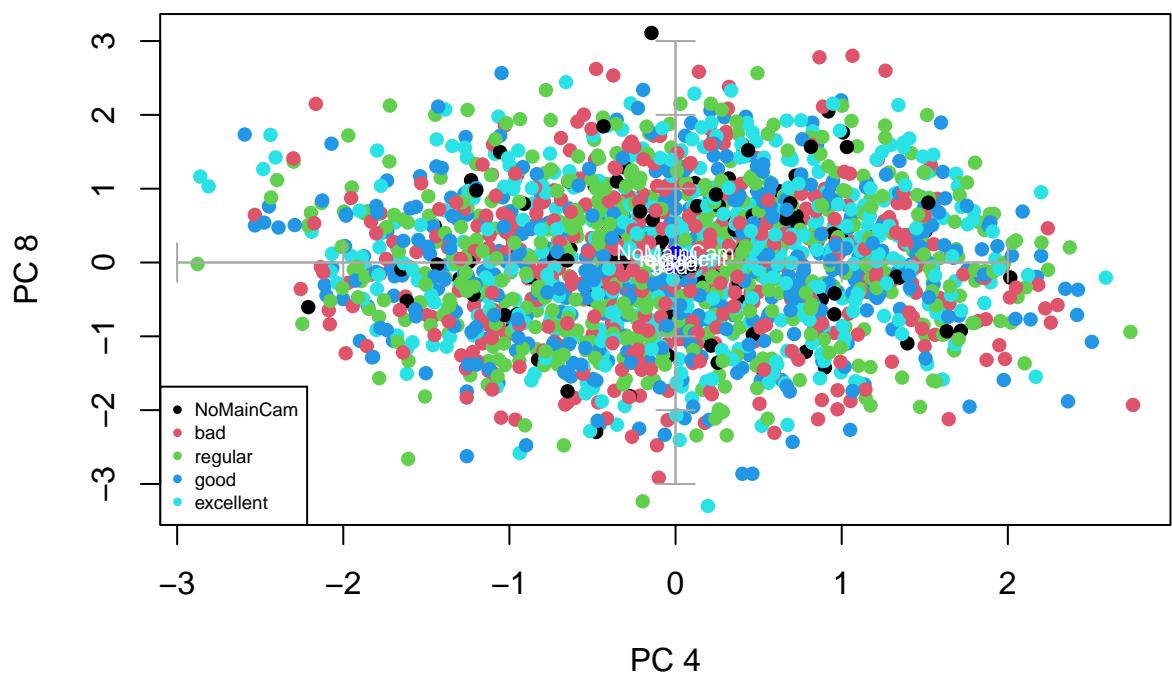


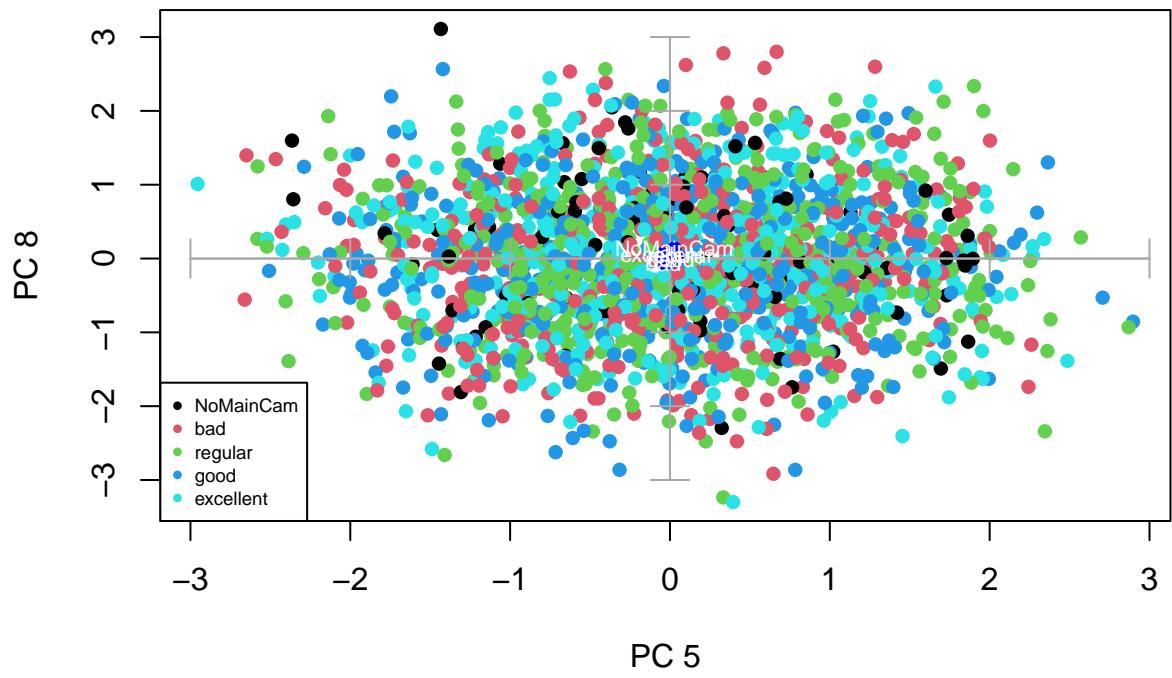
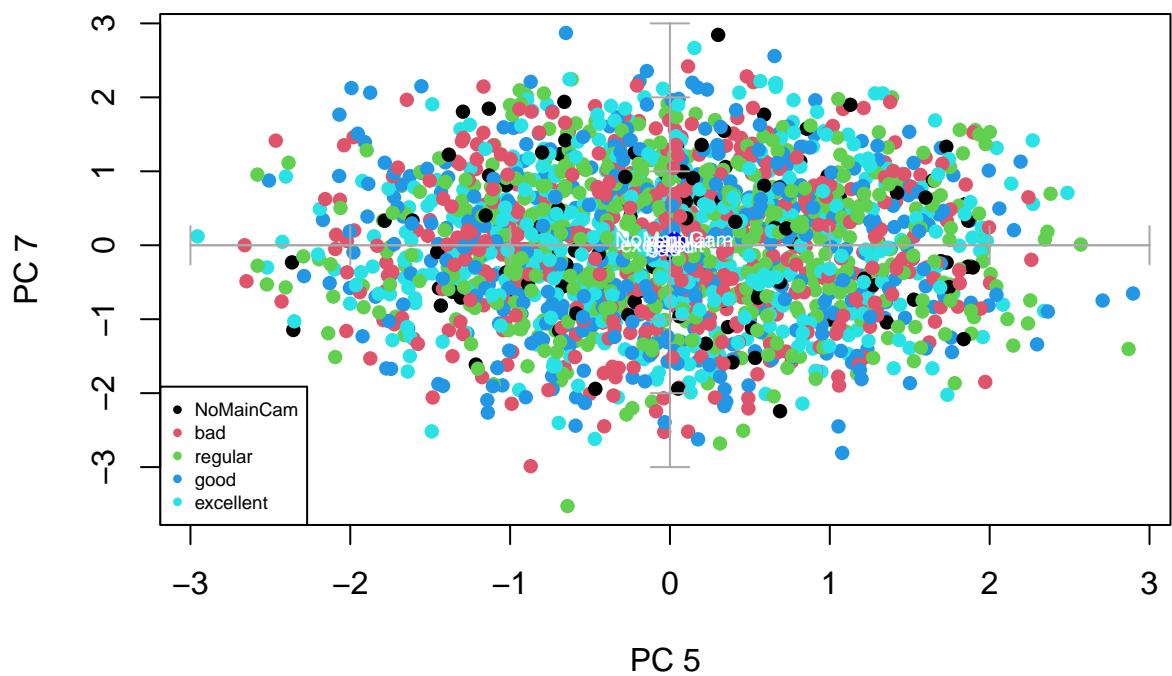


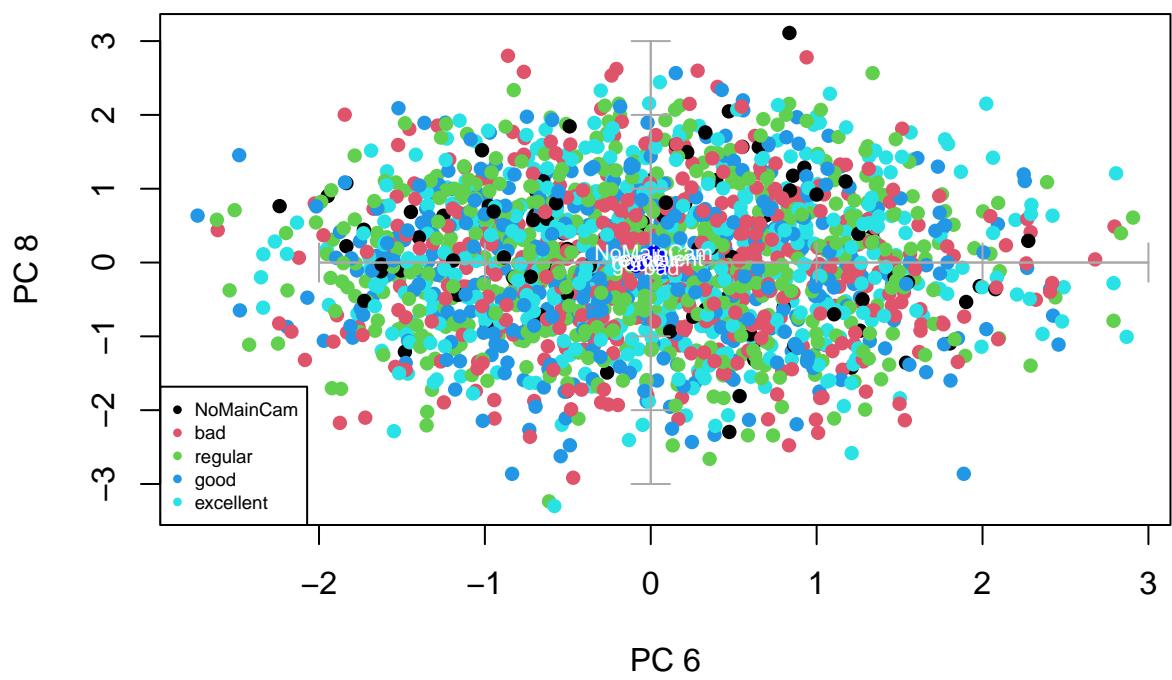
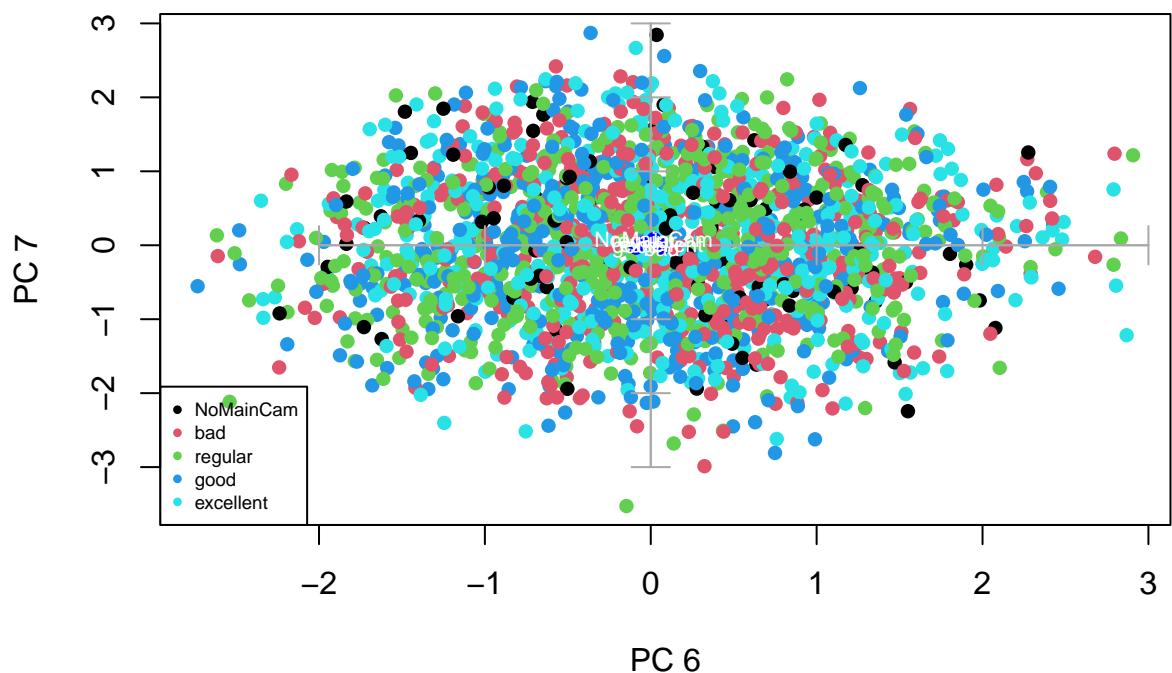


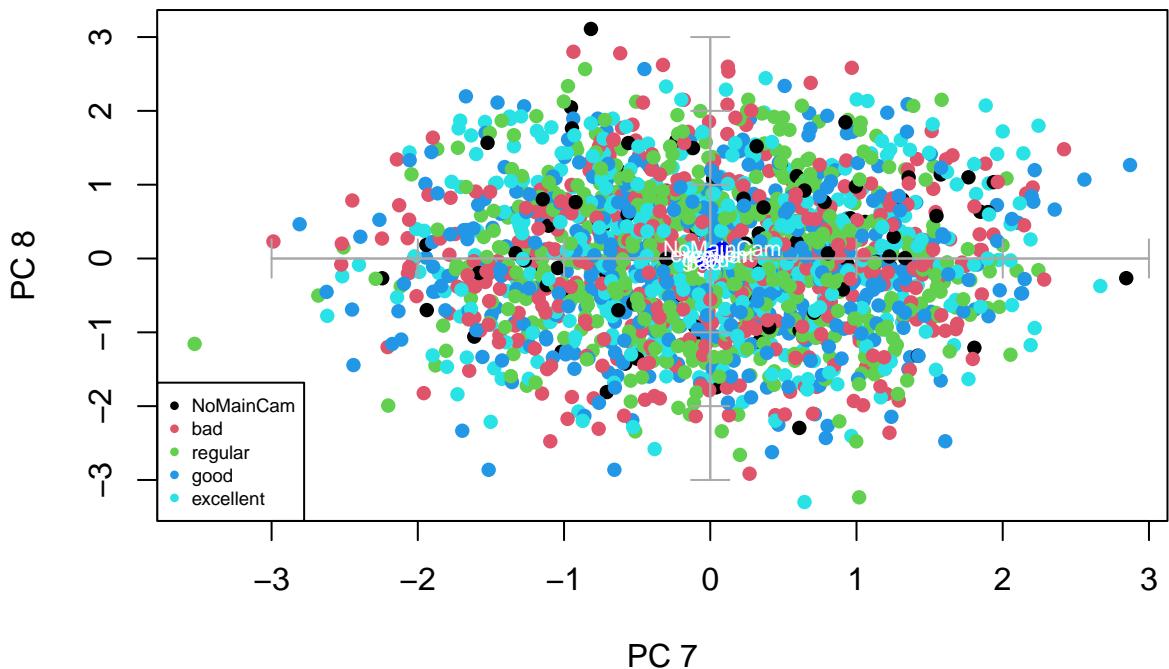










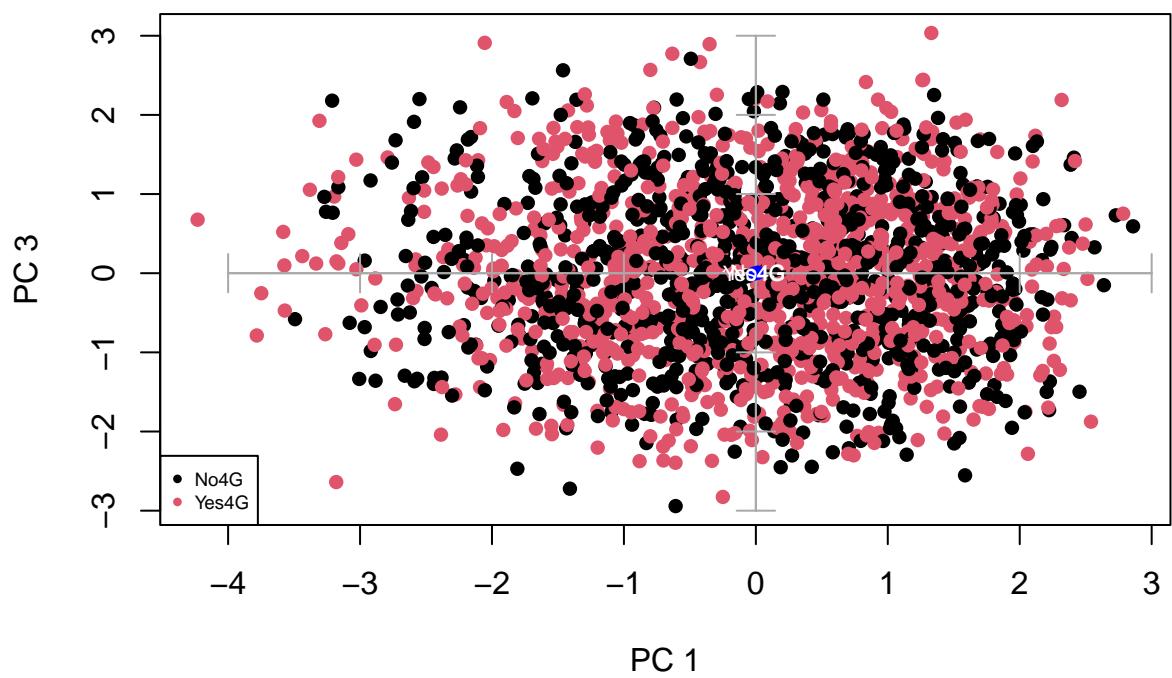
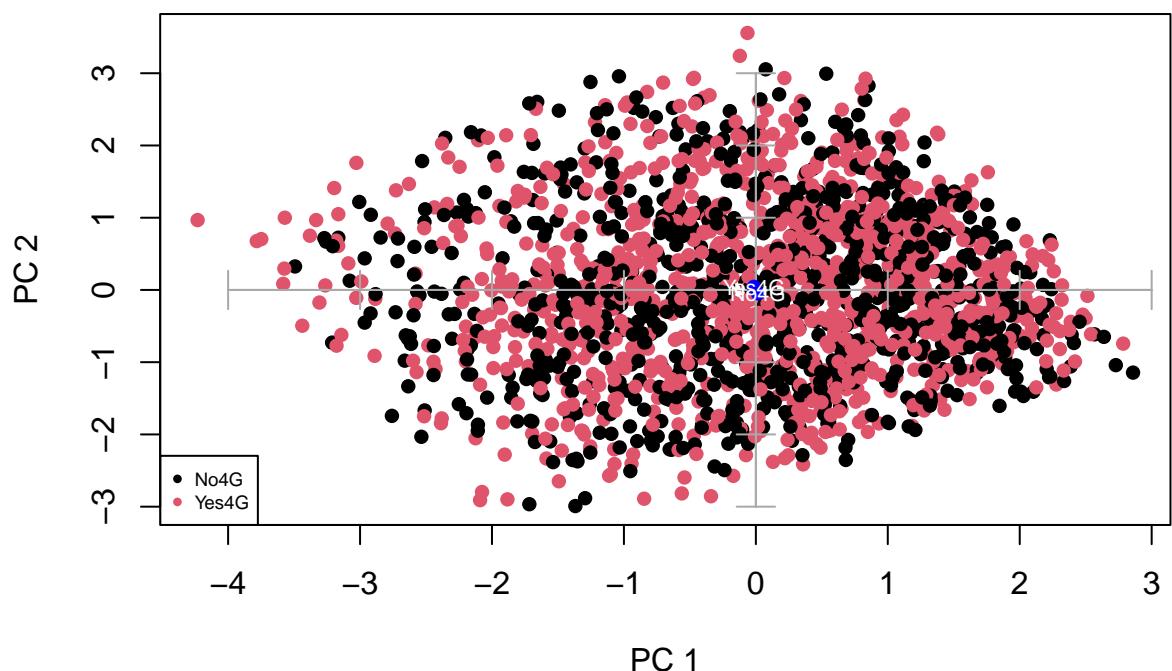


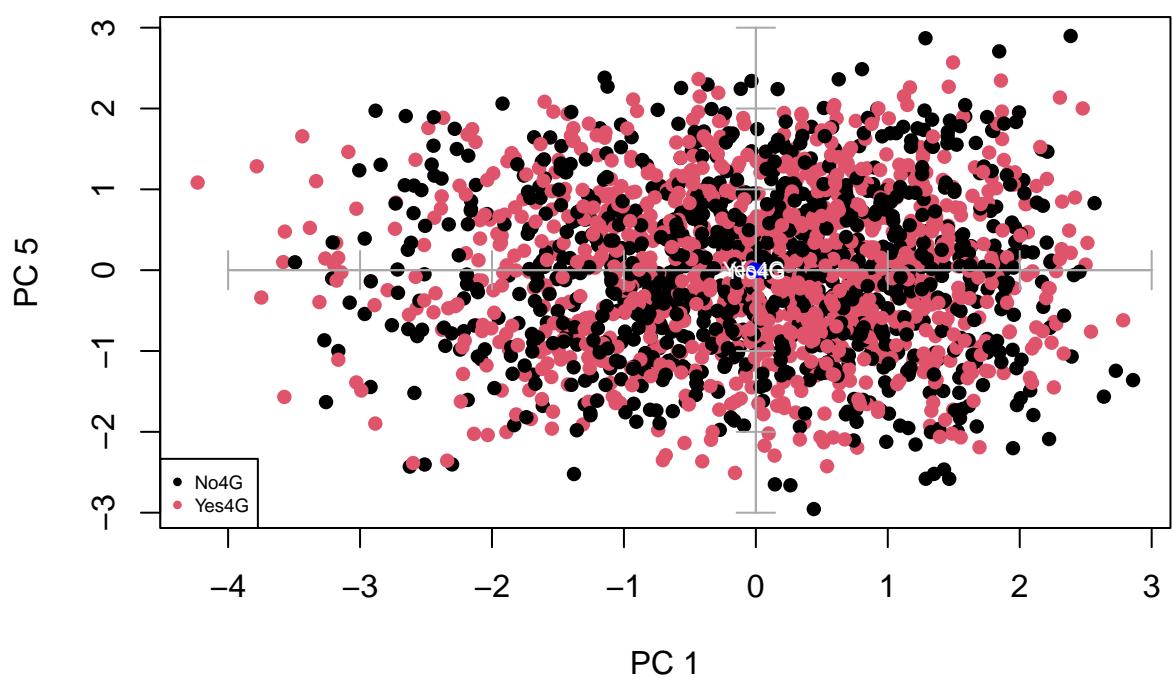
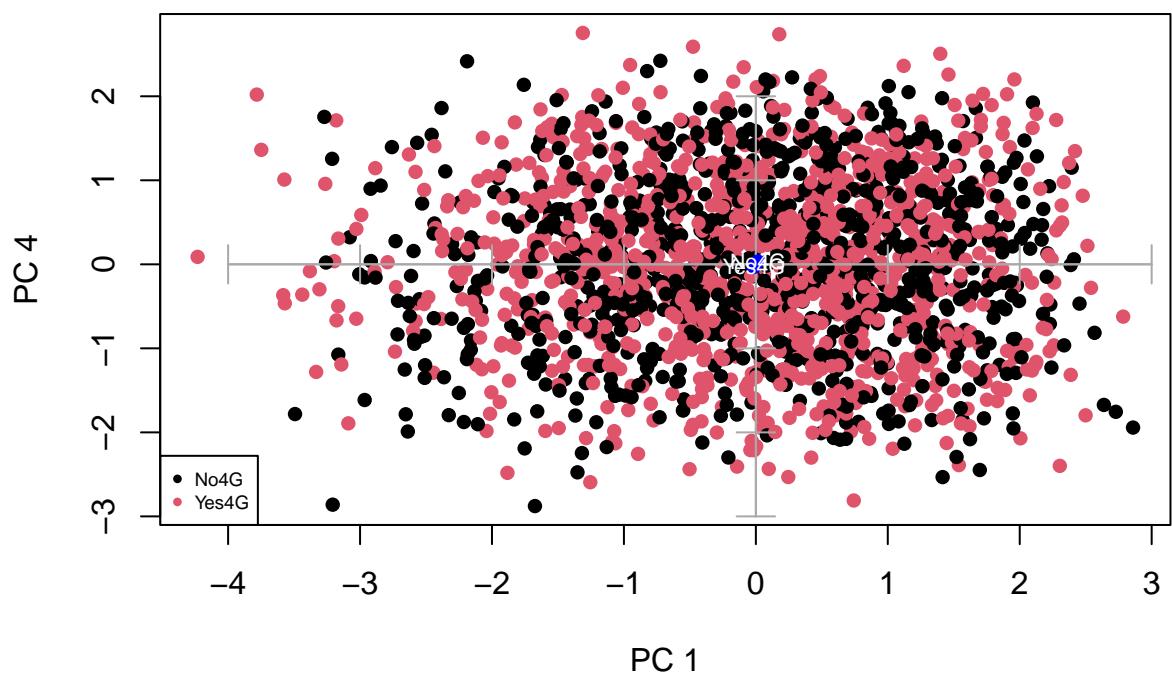
```

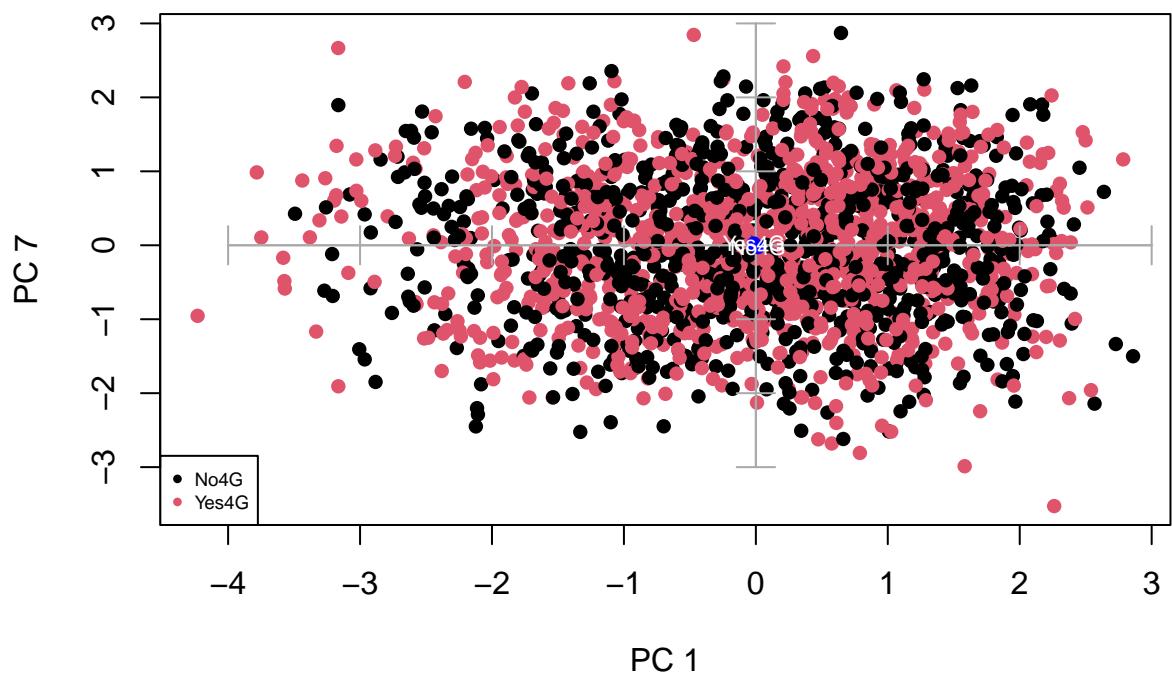
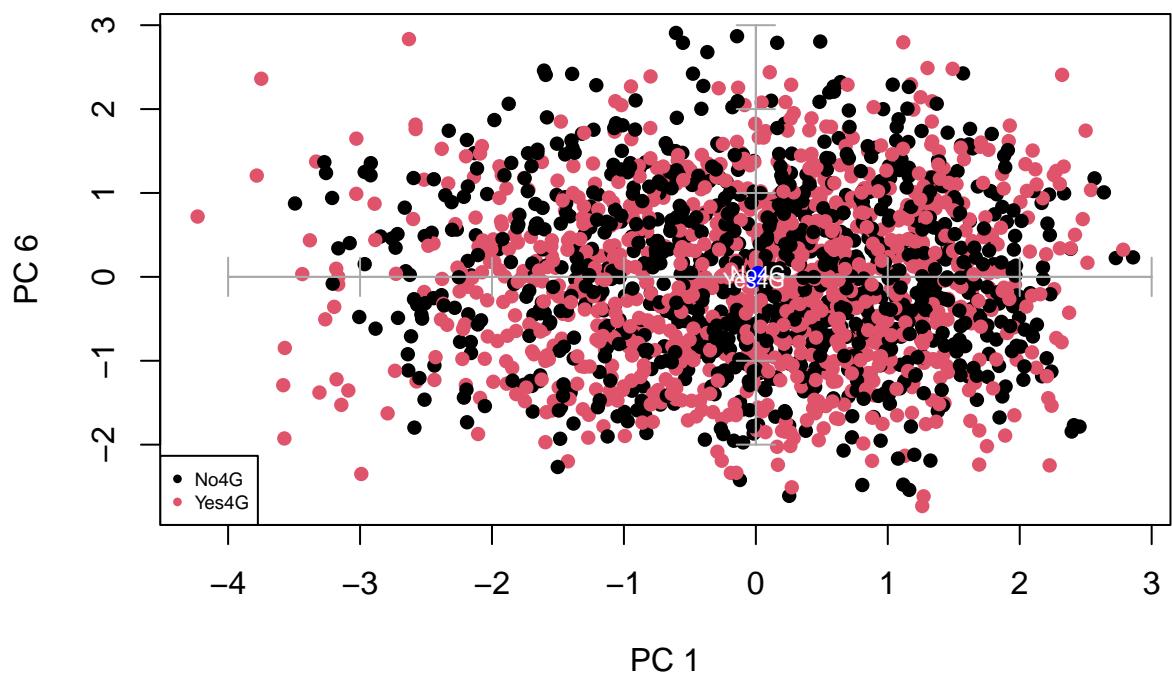
for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$four_g
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab= paste("PC",toString(i)) , ylab= paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
    legend("bottomleft",levels(varcat),pch=16,col=c(1:2), cex=0.6)

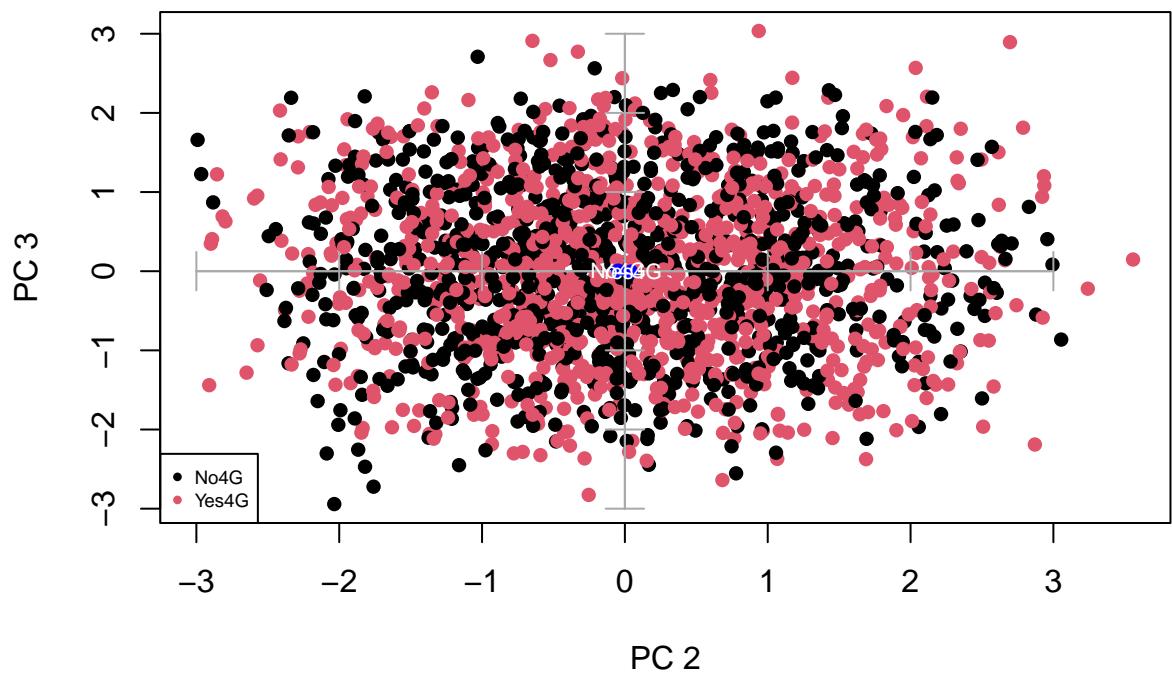
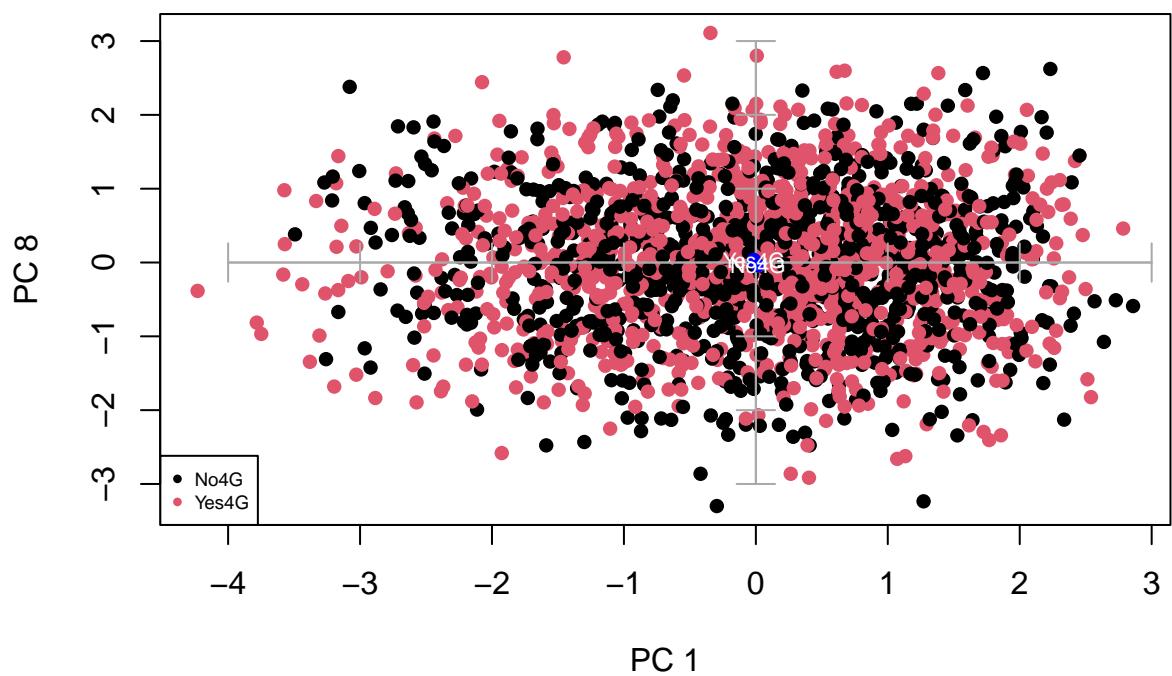
    #select your qualitative variable
    fdic1 = tapply(Psi[,i],varcat,mean)
    fdic2 = tapply(Psi[,j],varcat,mean)
    points(fdic1,fdic2,pch=16,col="blue")
    text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
  }
}

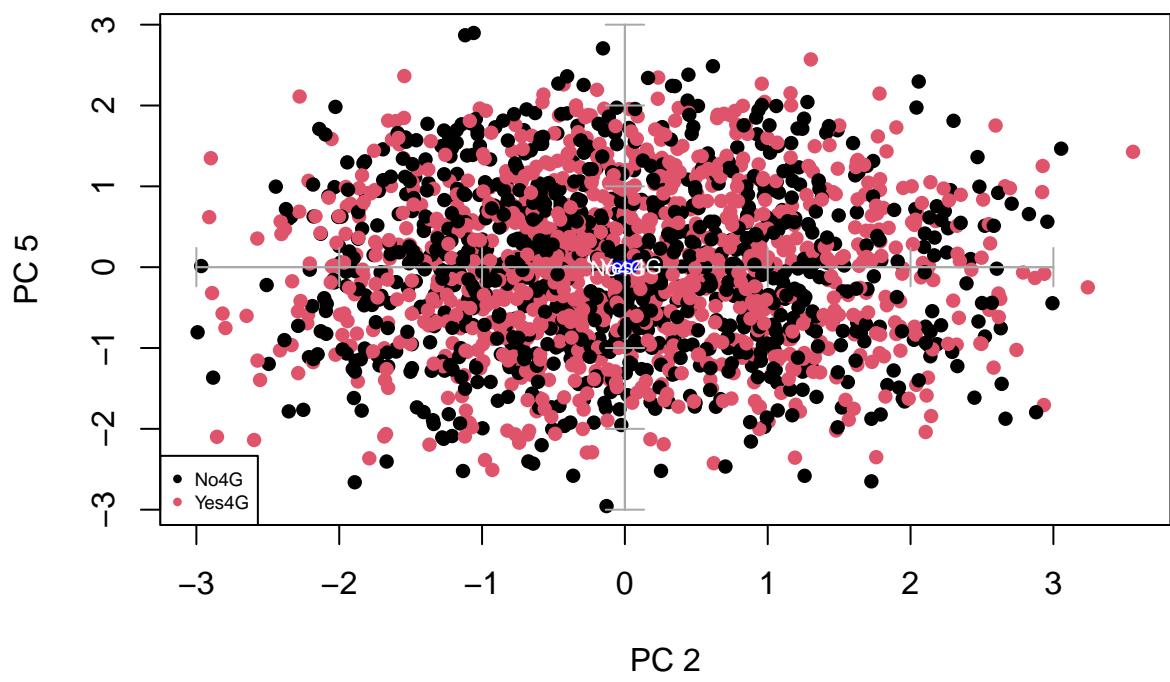
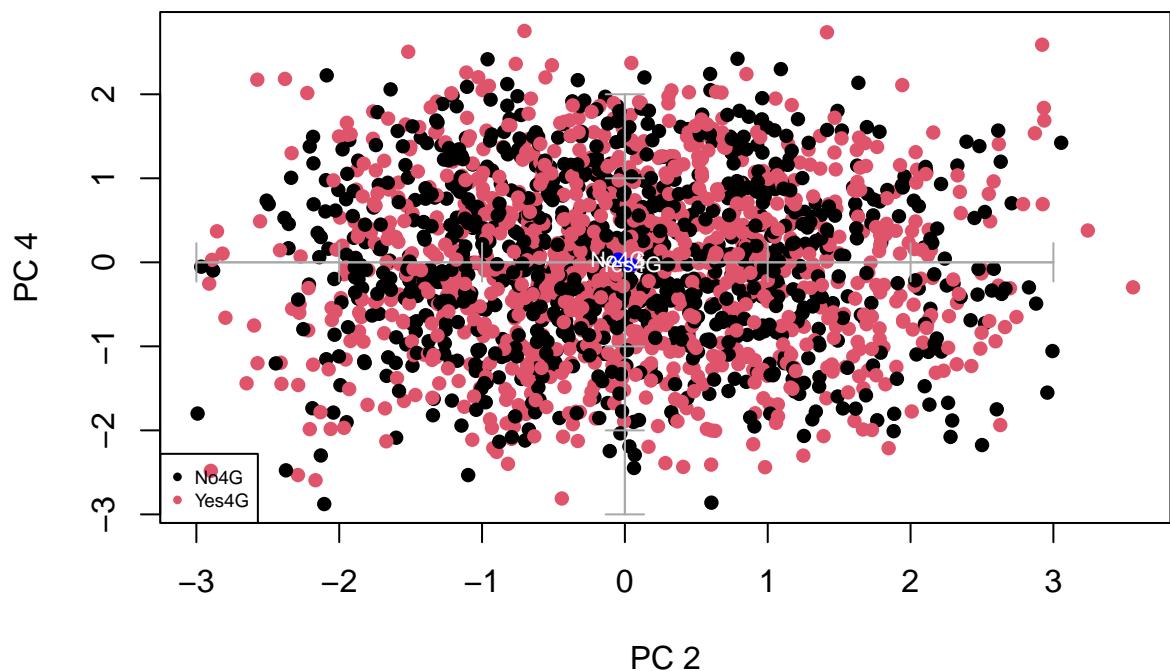
```

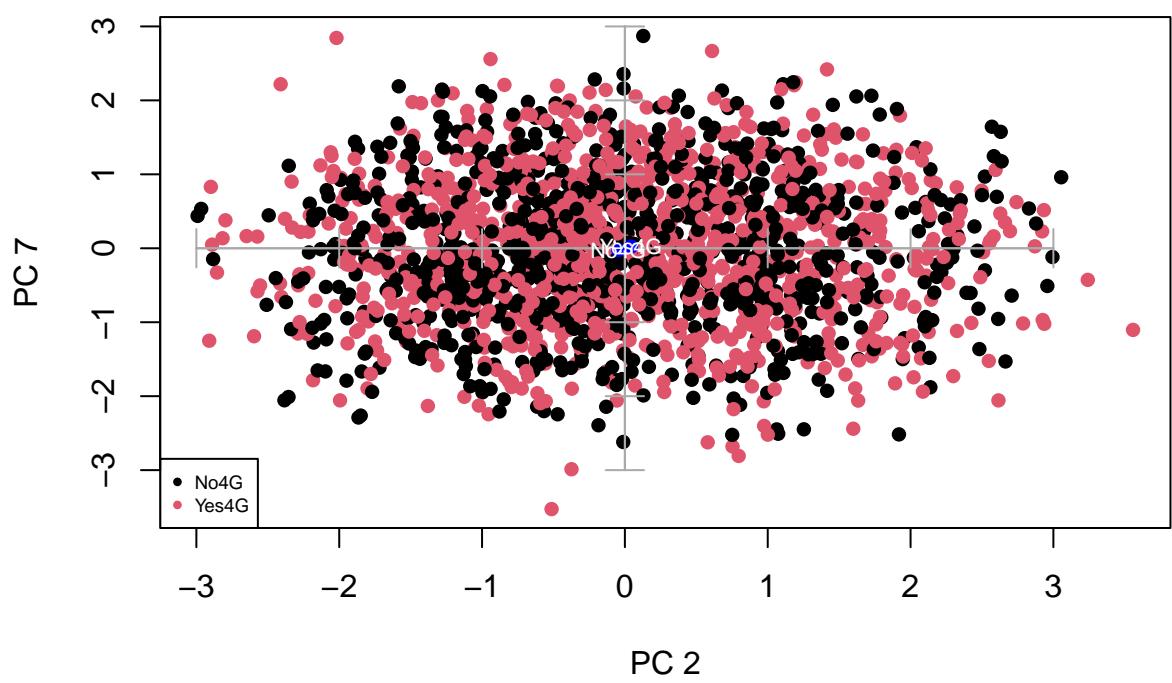
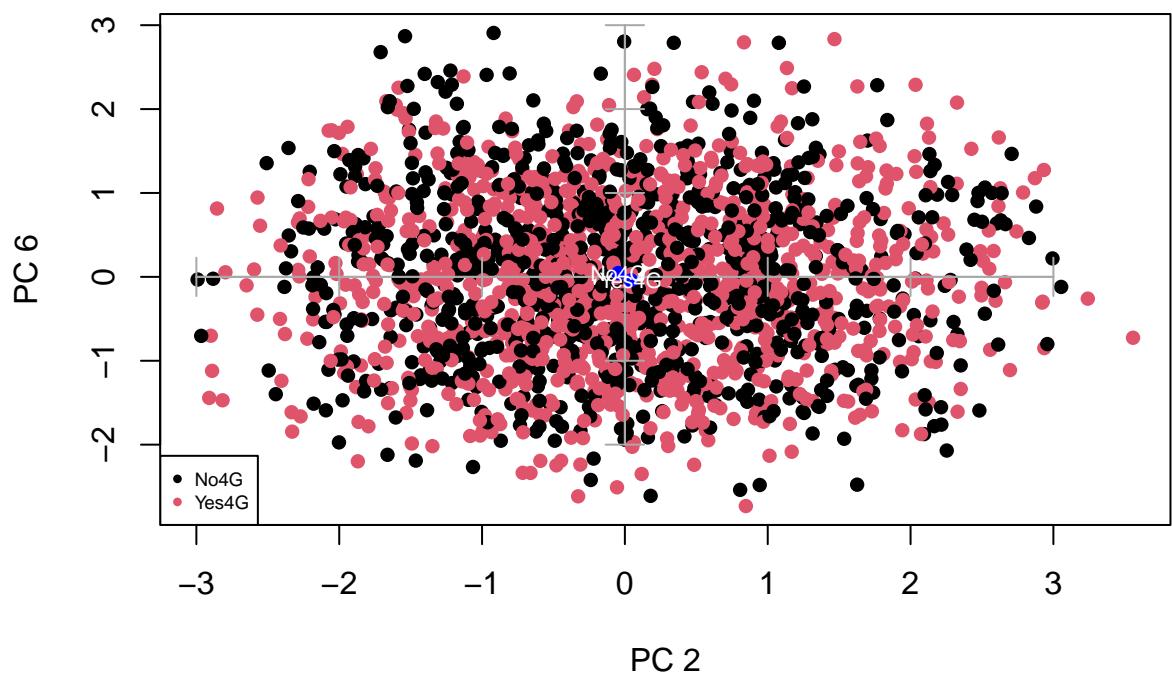


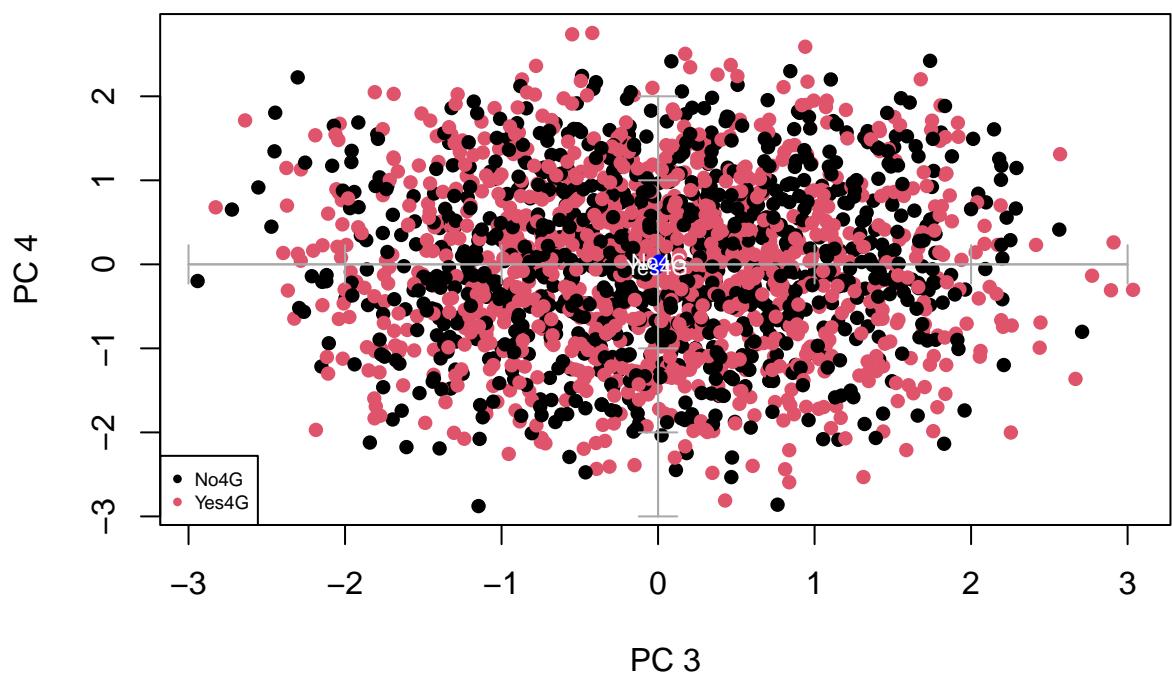
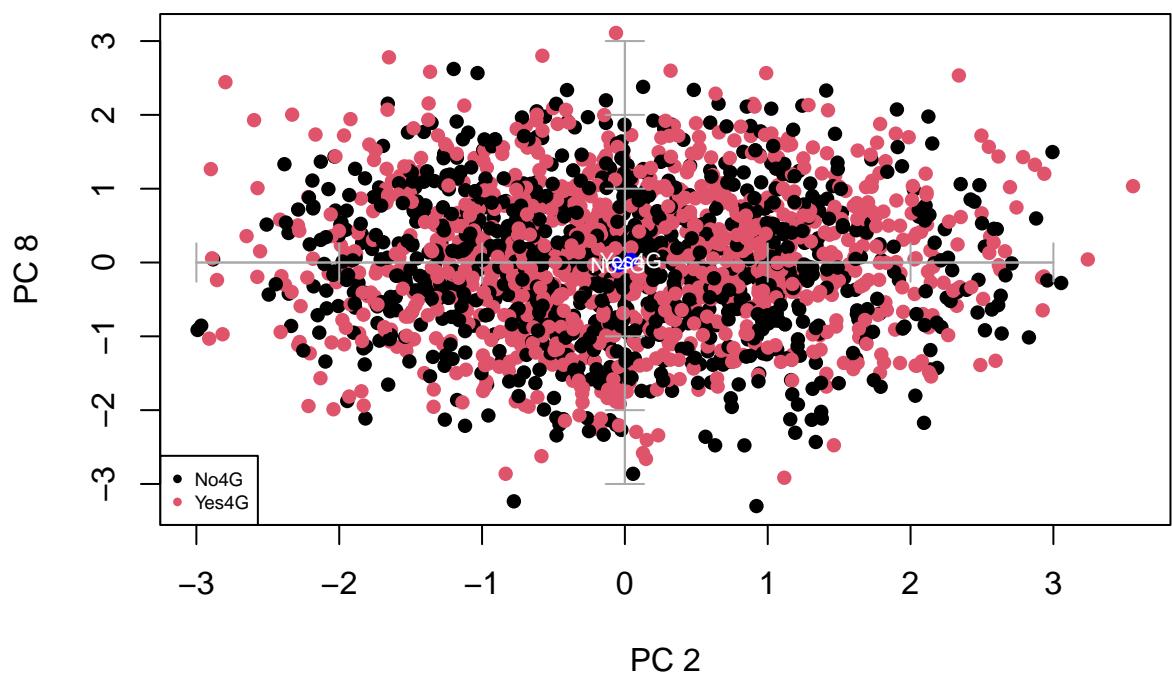


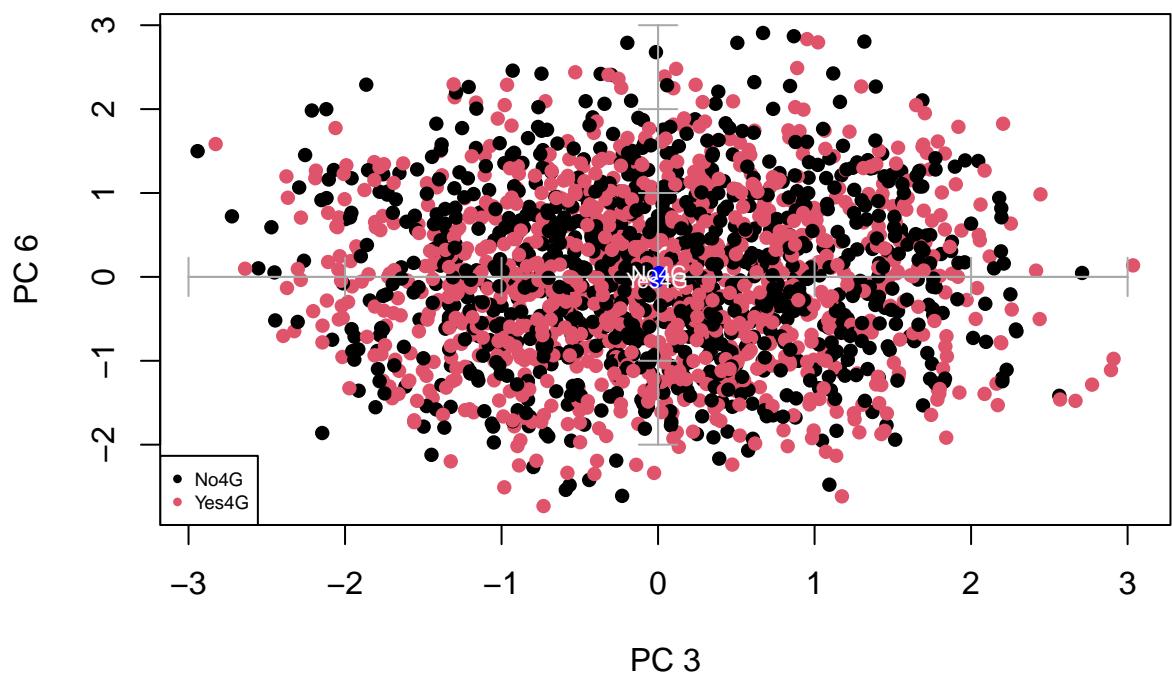
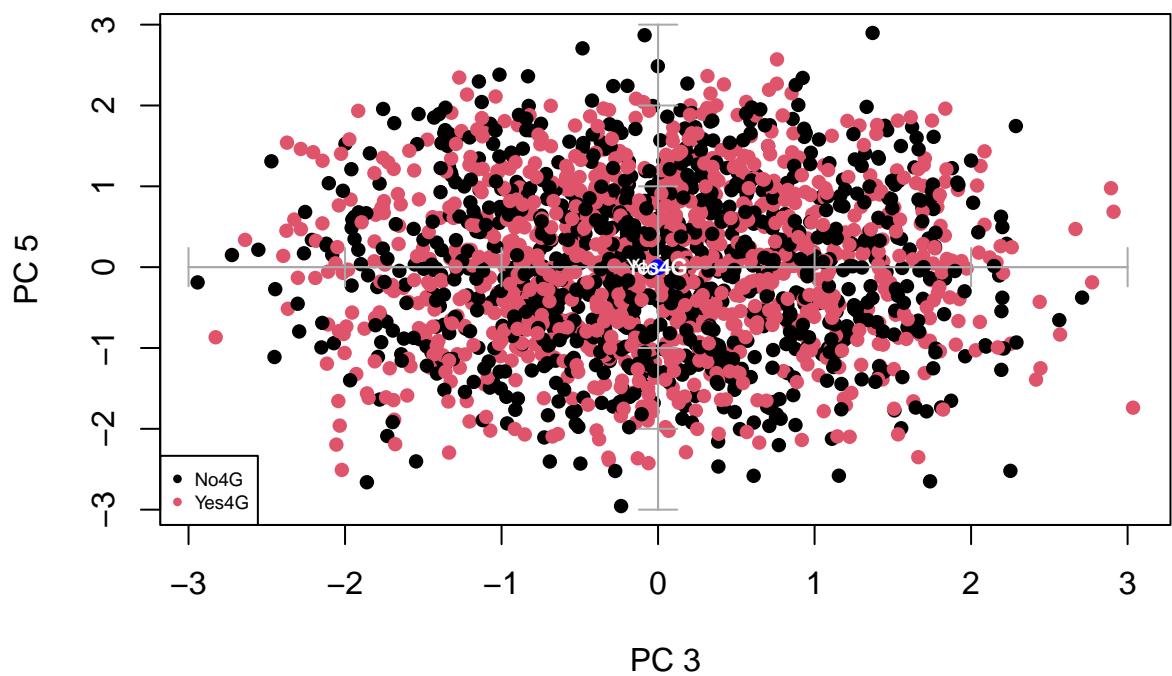


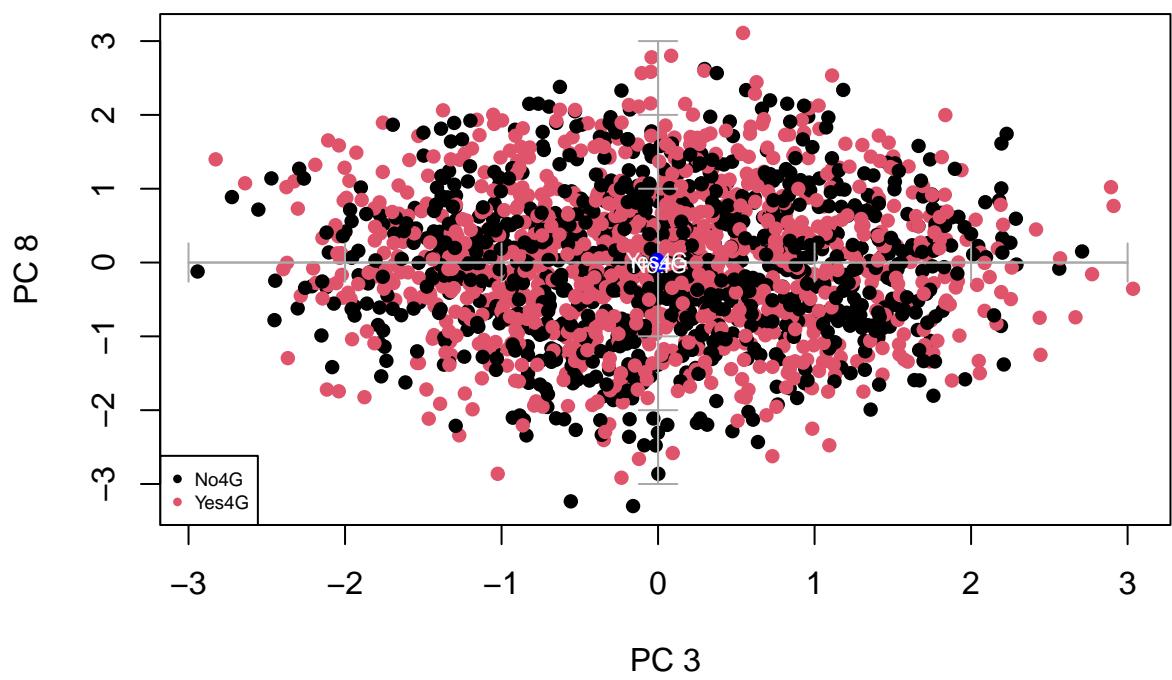
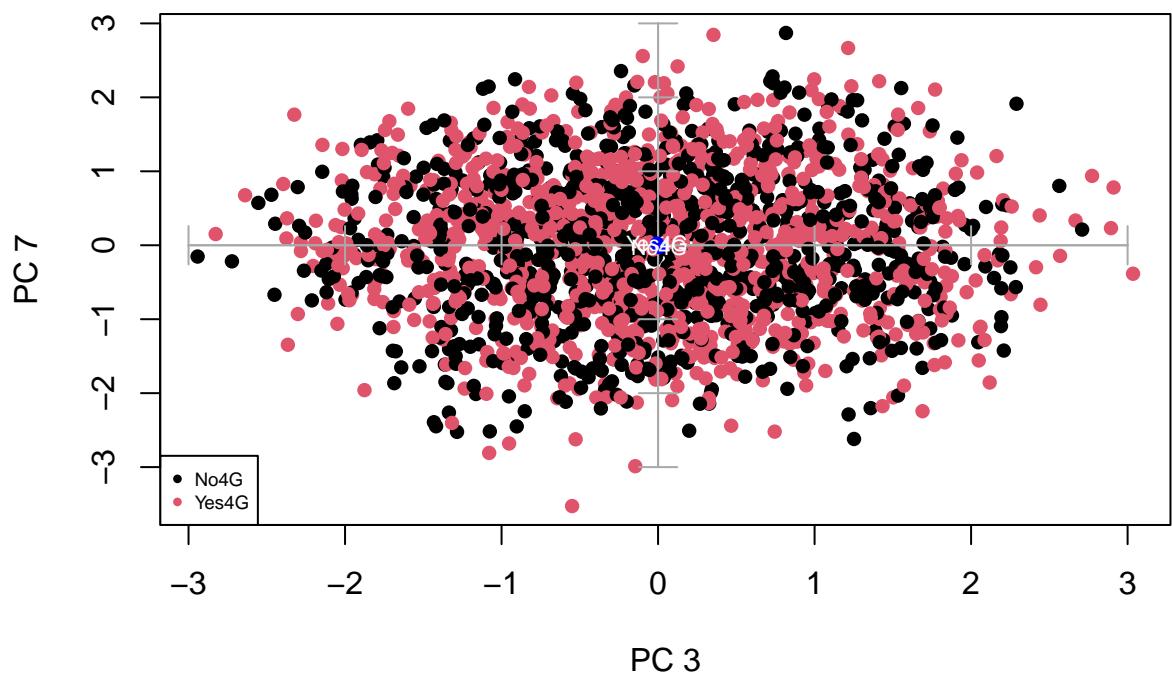


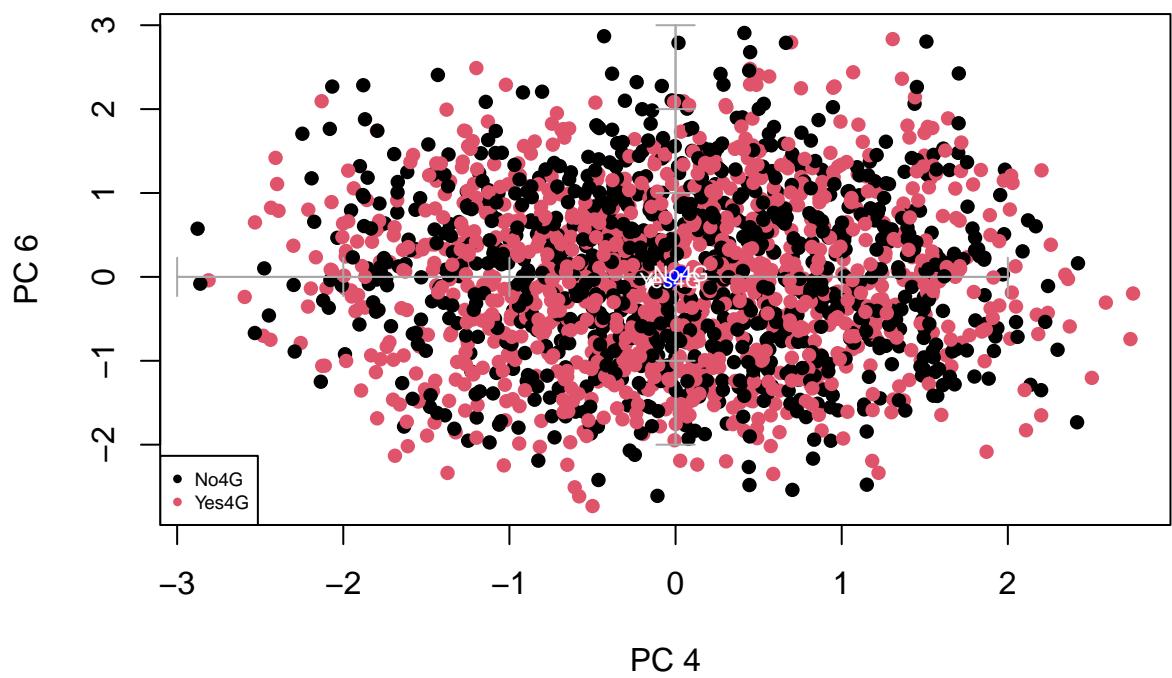
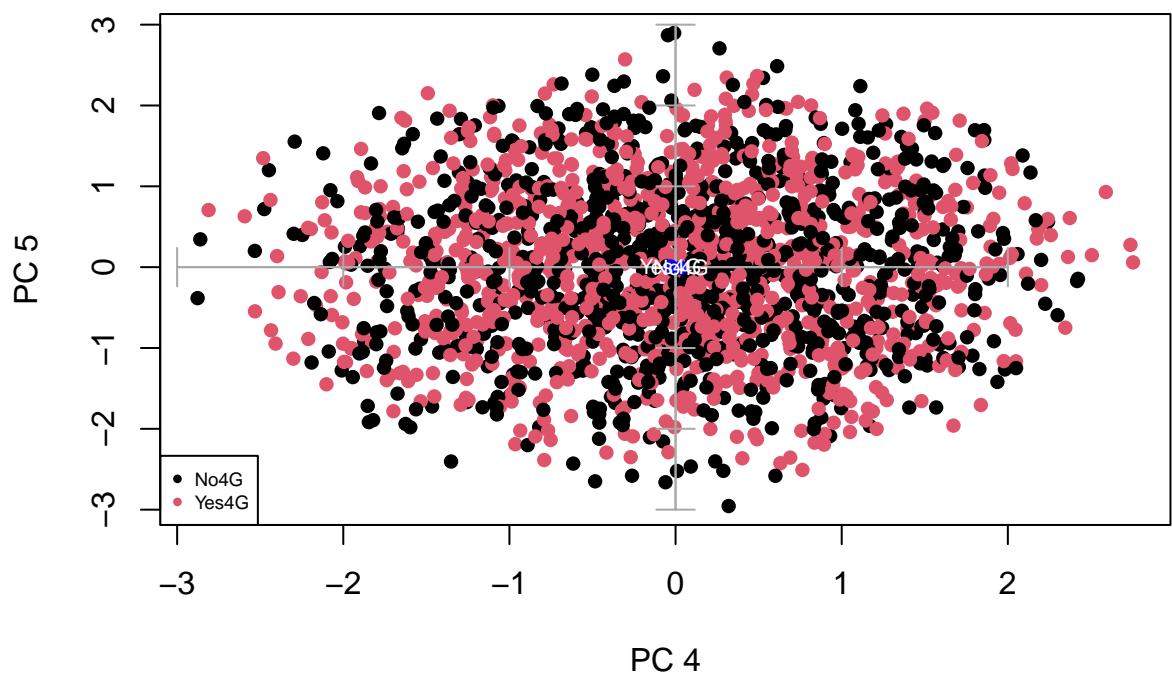


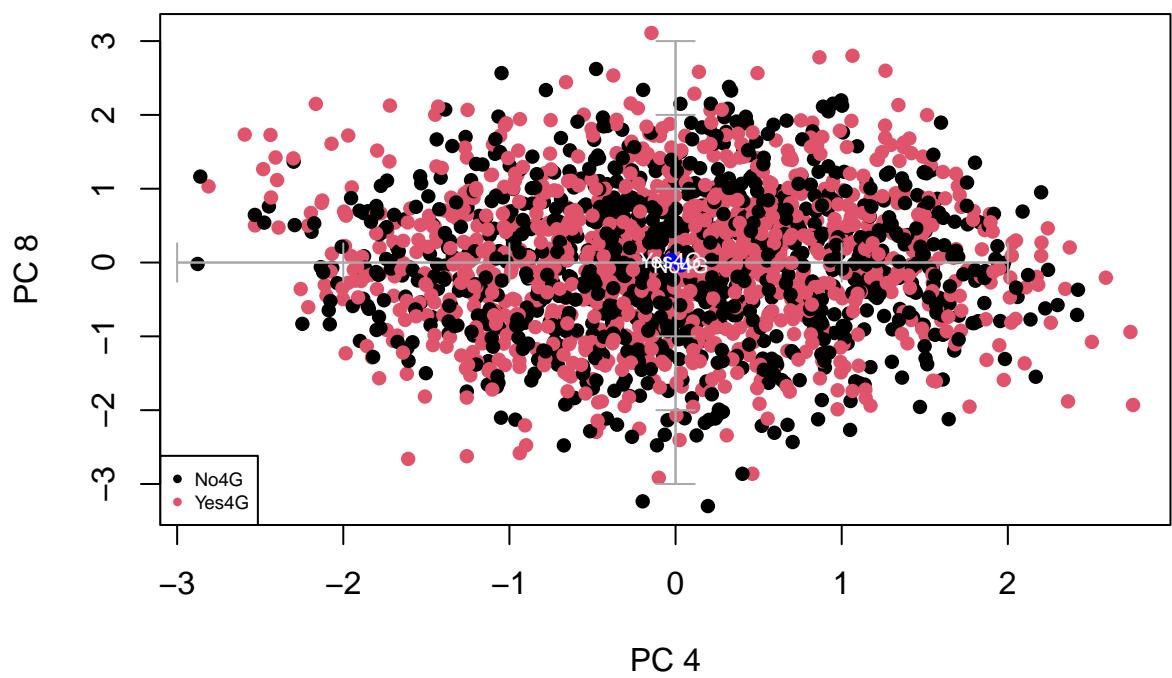
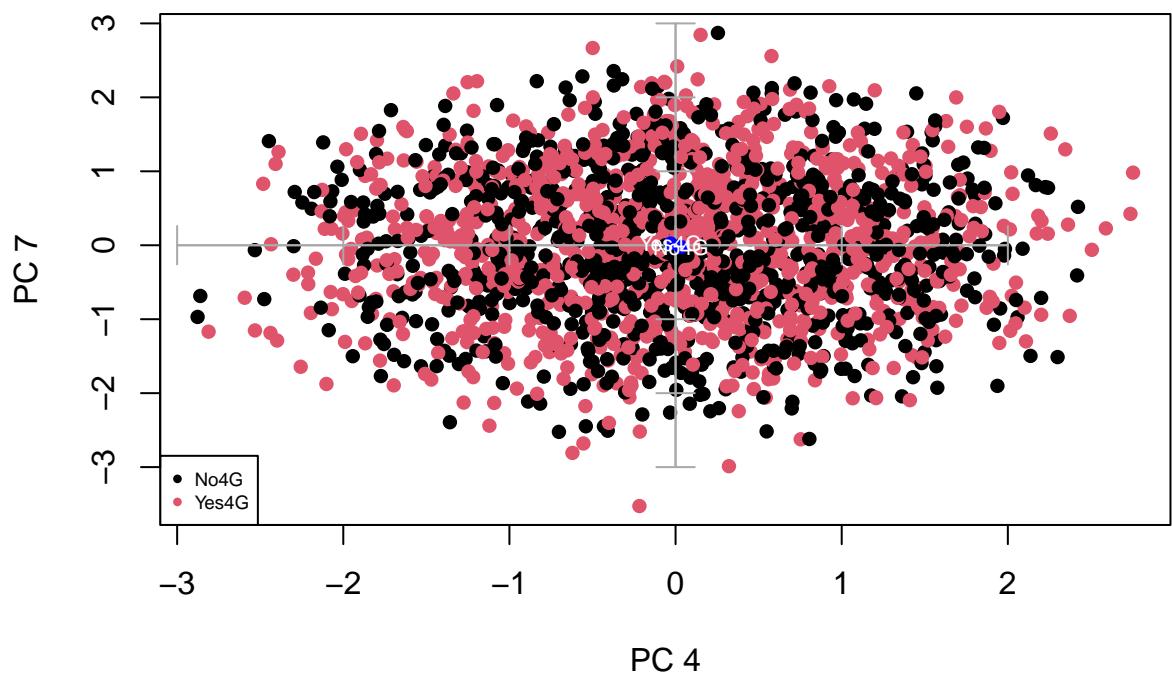


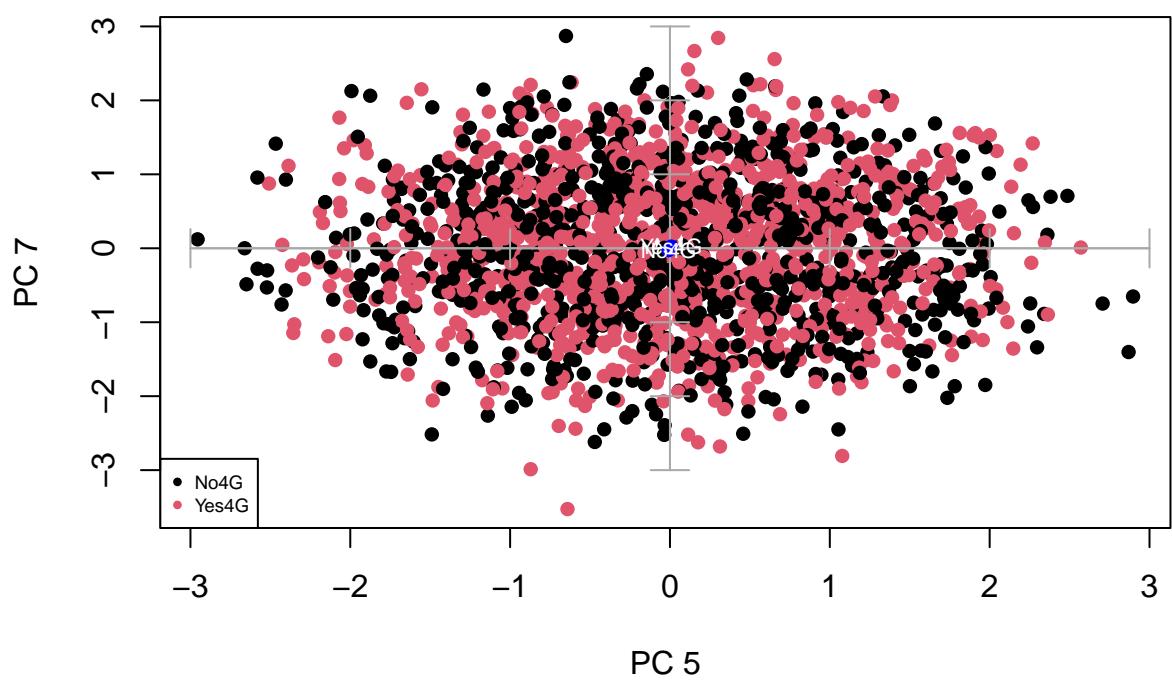
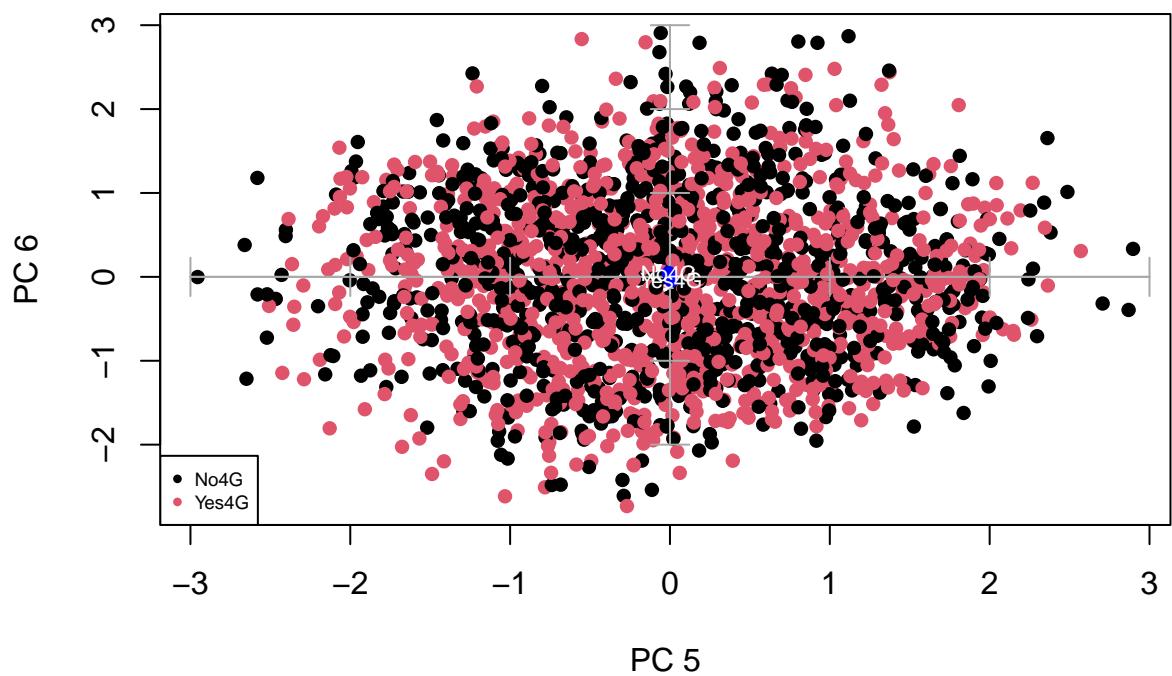


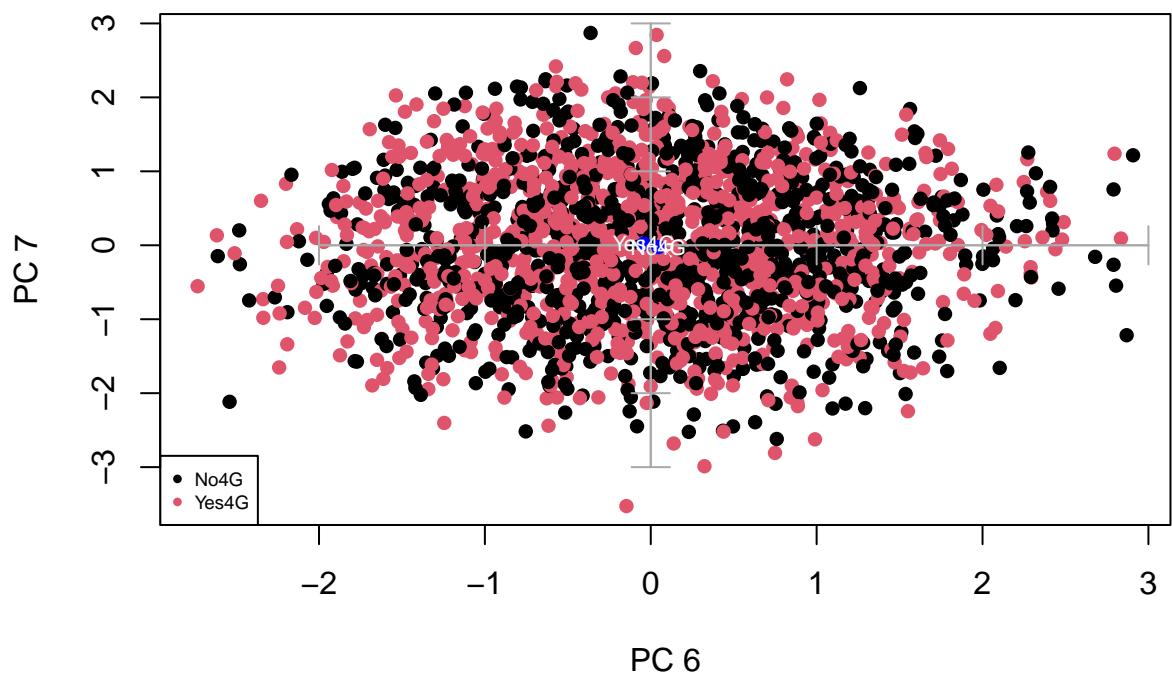
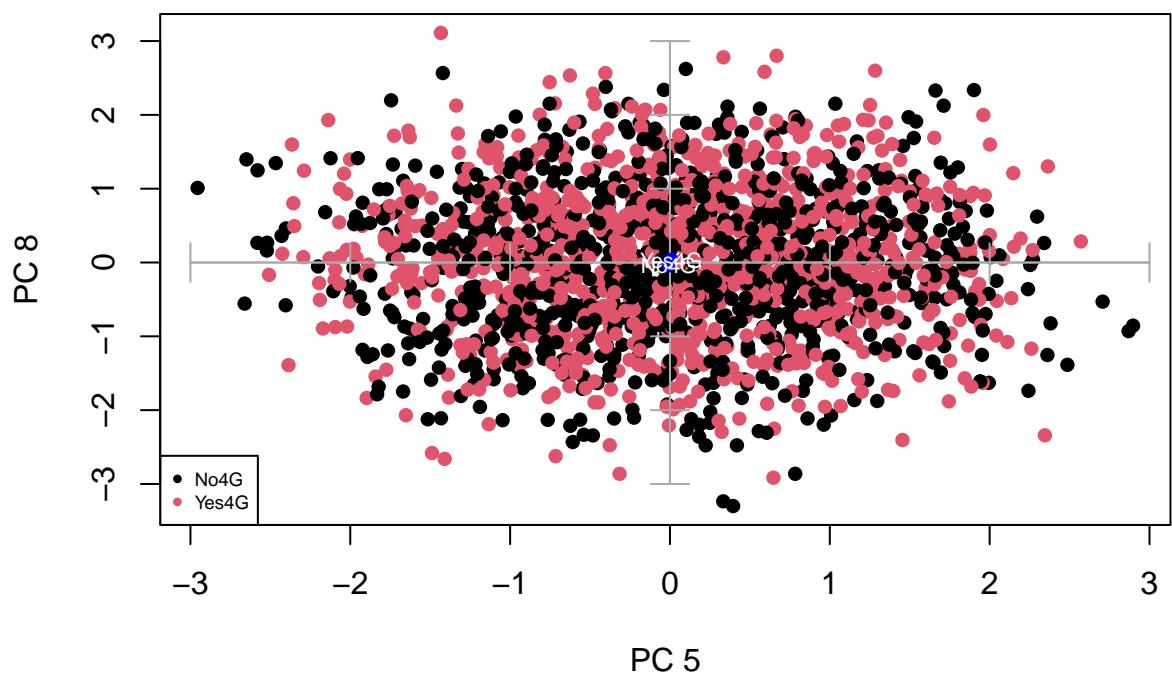


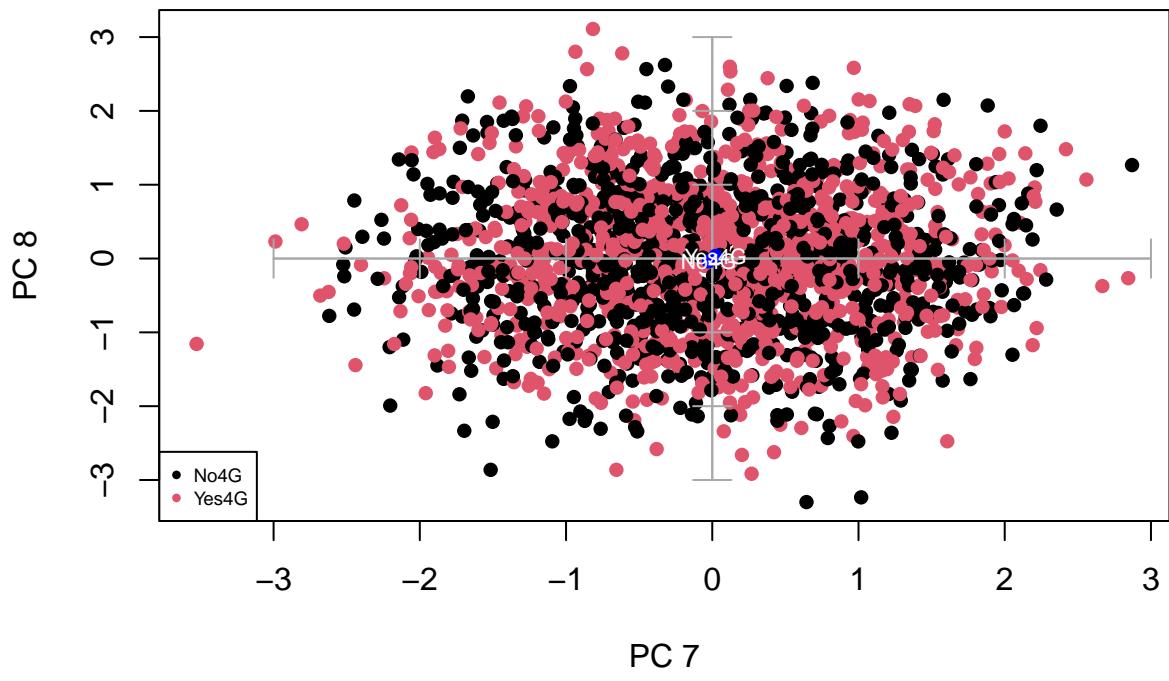
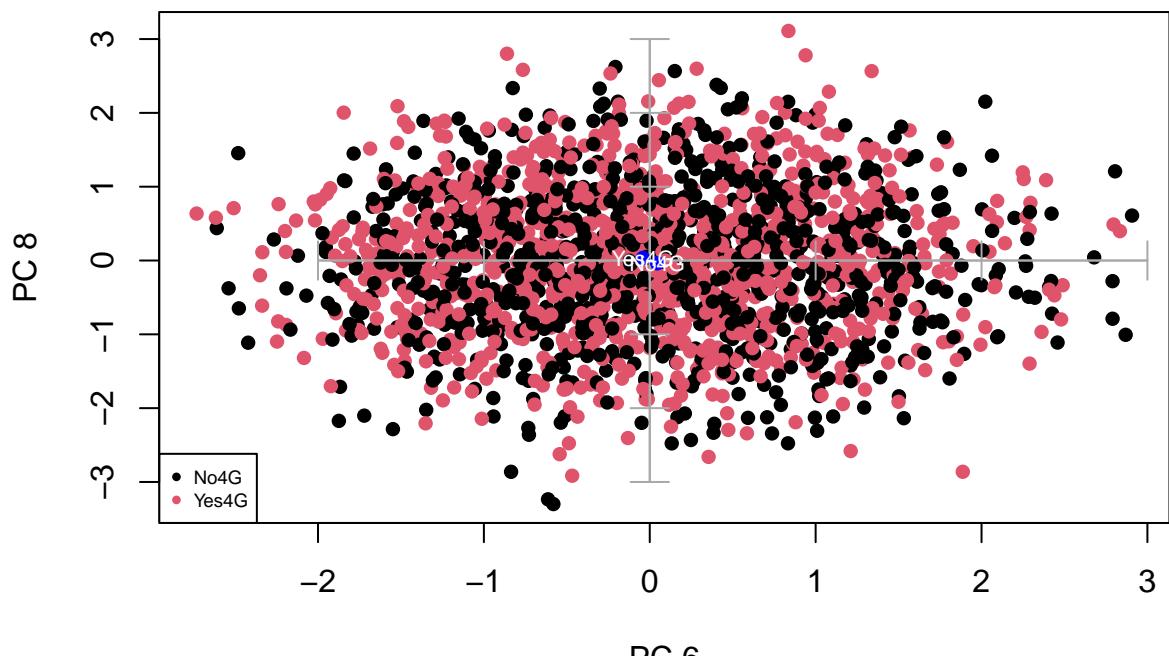












```

for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$fc
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
  }
}
  
```

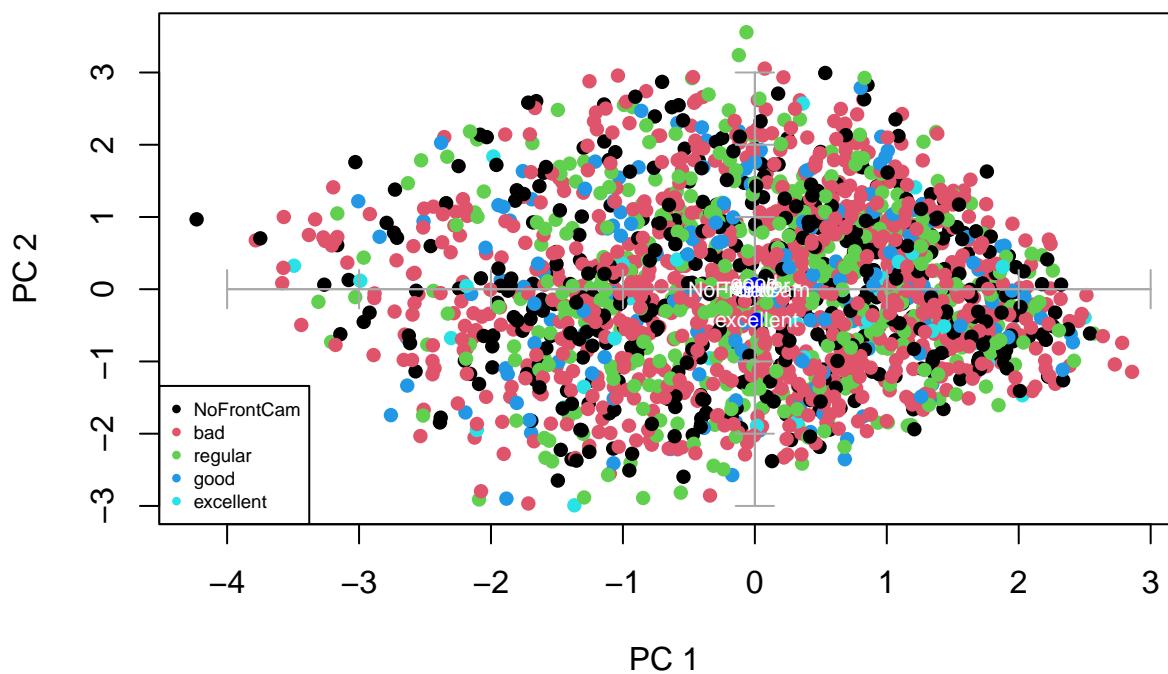
```

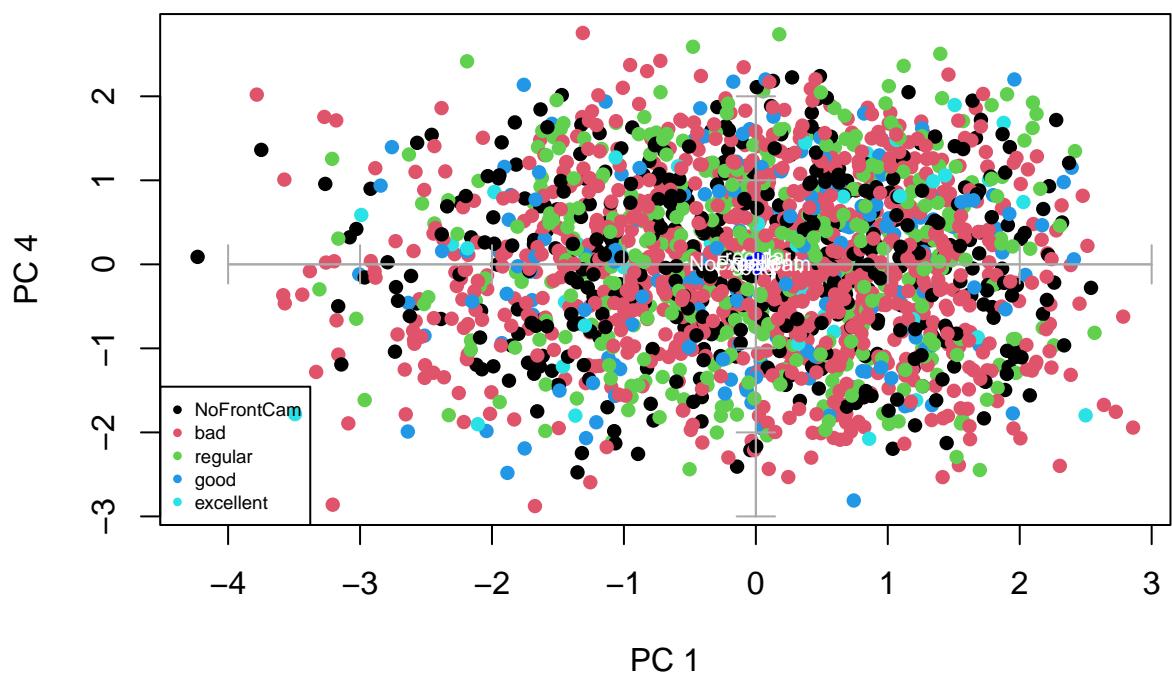
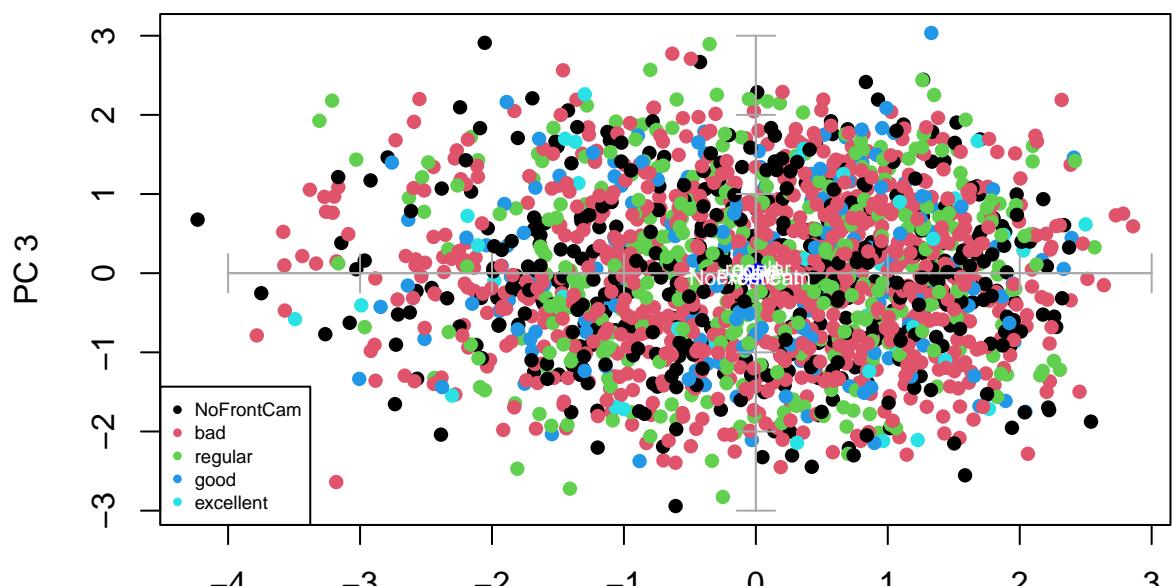
legend("bottomleft",levels(varcat),pch=16,col=c(1:5), cex=0.6)

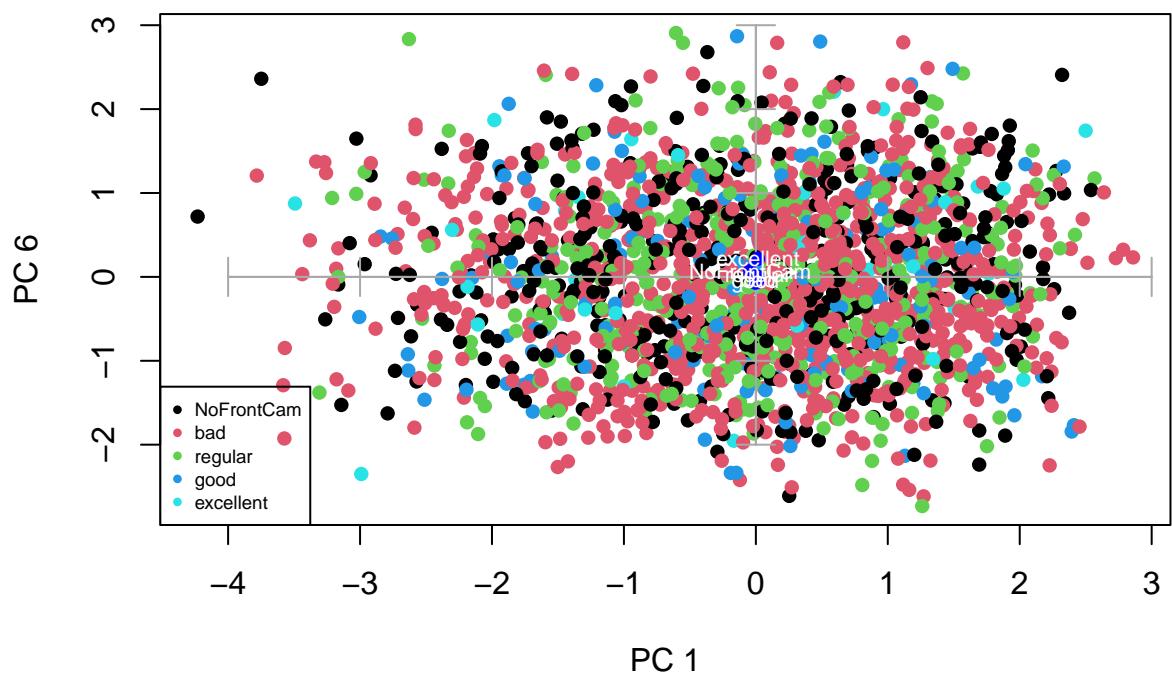
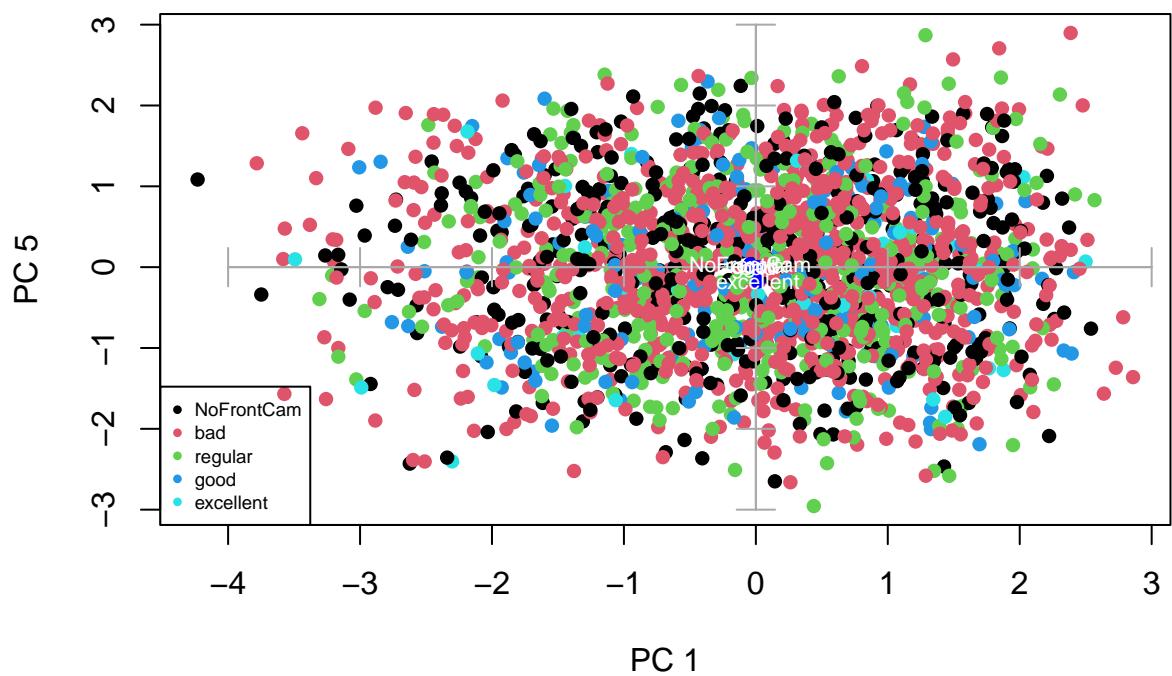
#select your qualitative variable
fdic1 = tapply(Psi[,i],varcat,mean)
fdic2 = tapply(Psi[,j],varcat,mean)
points(fdic1,fdic2,pch=16,col="blue")
text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
}

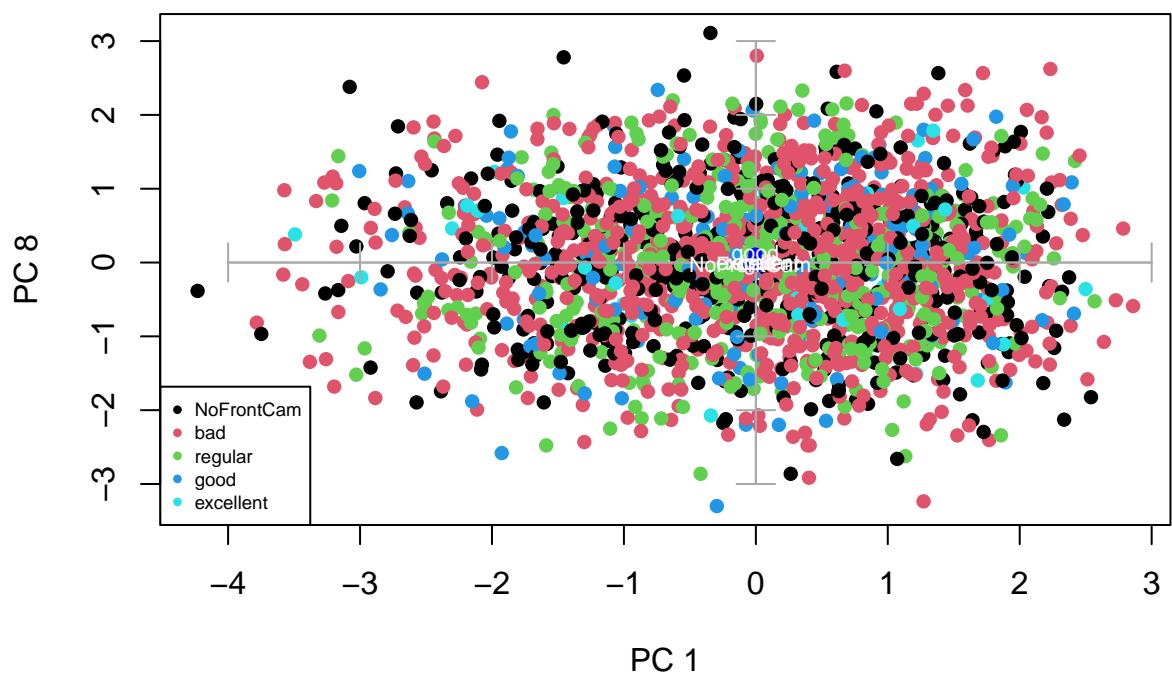
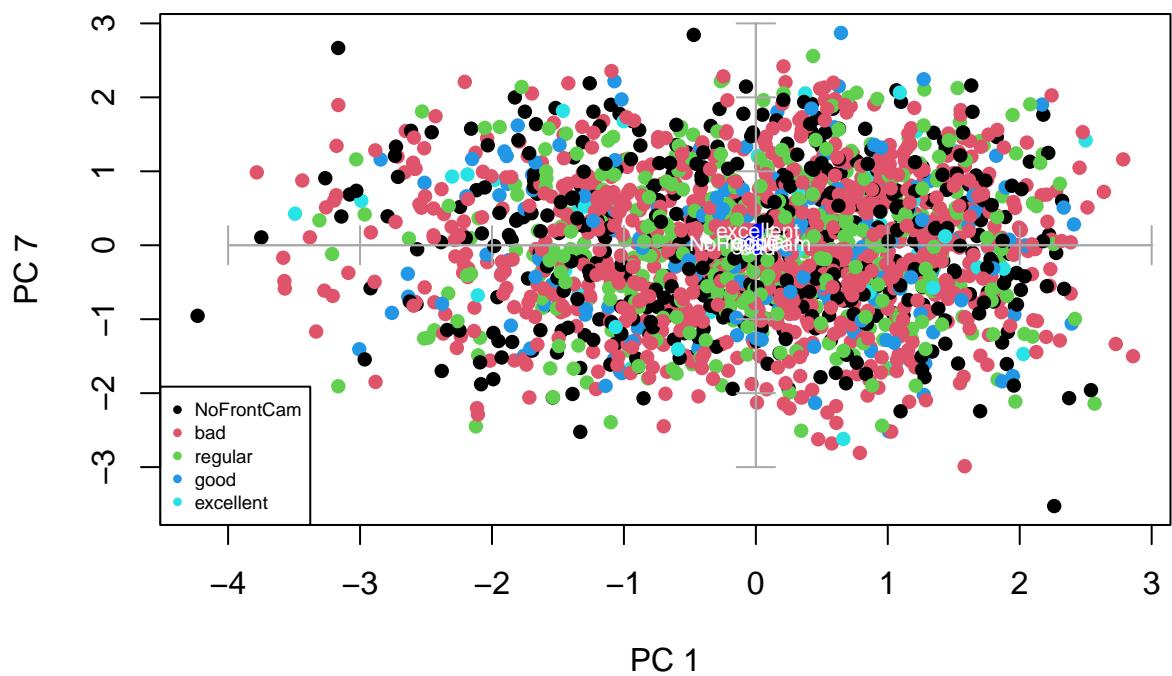
}

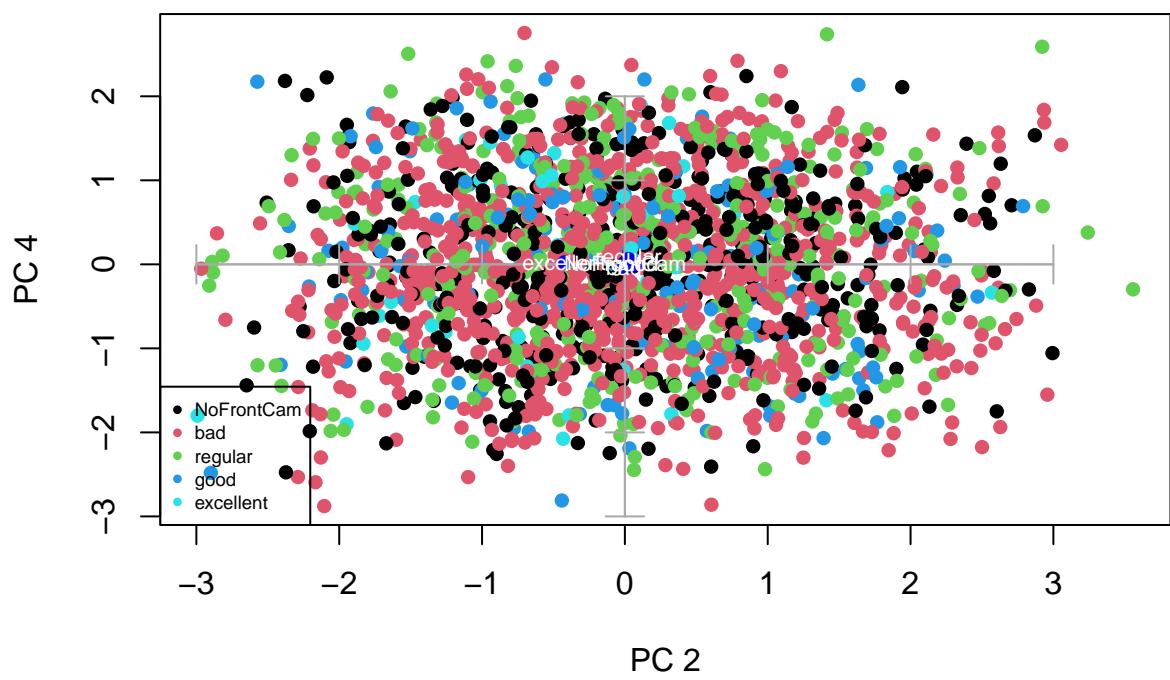
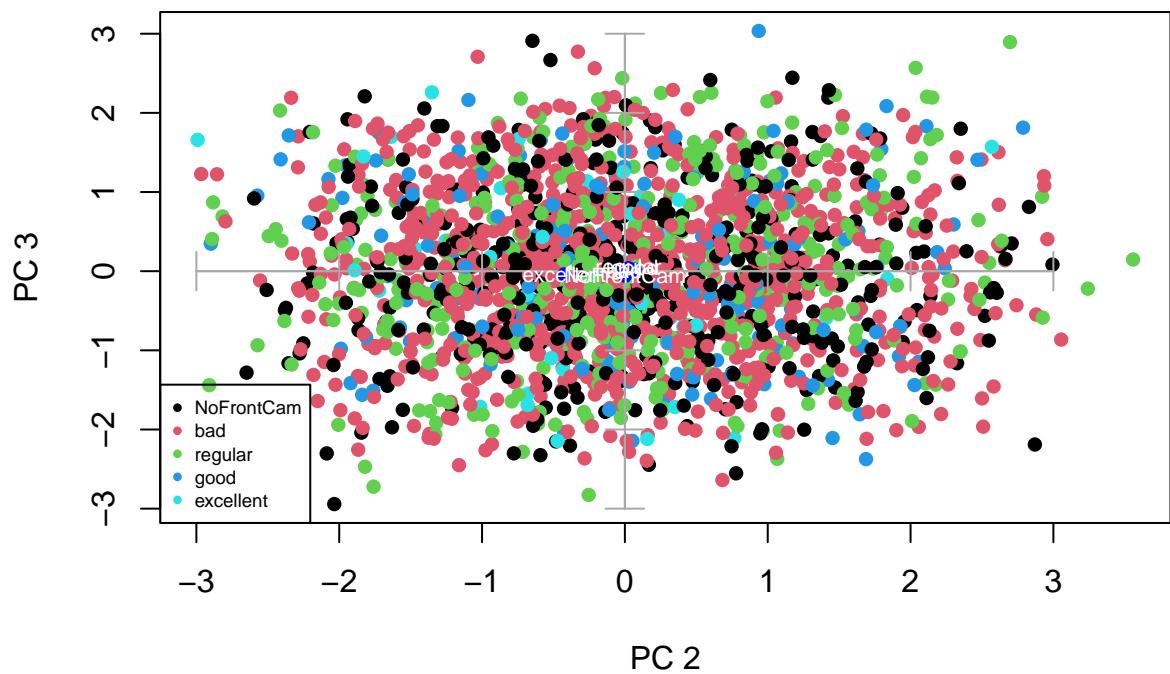
```

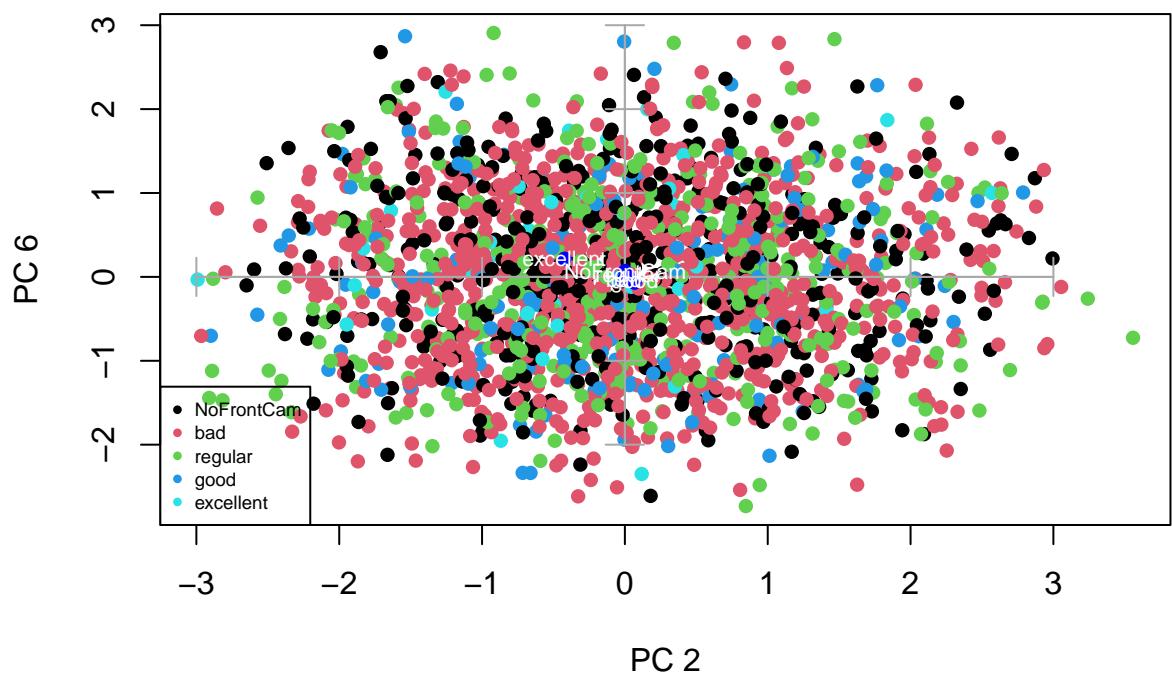
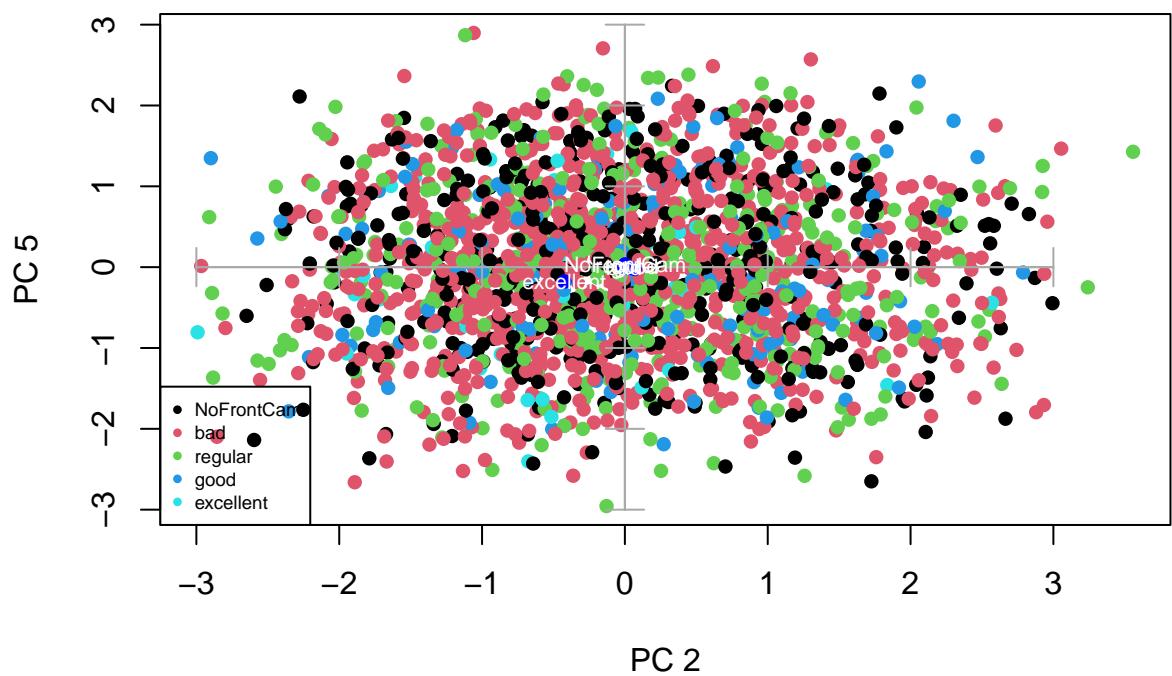


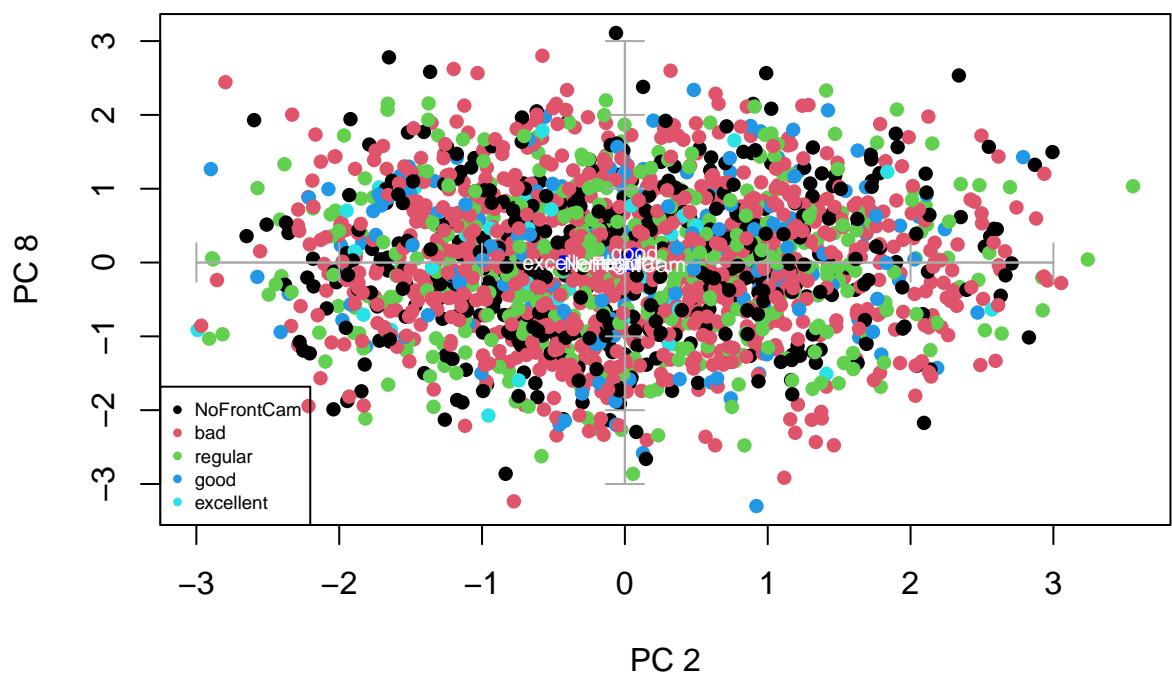
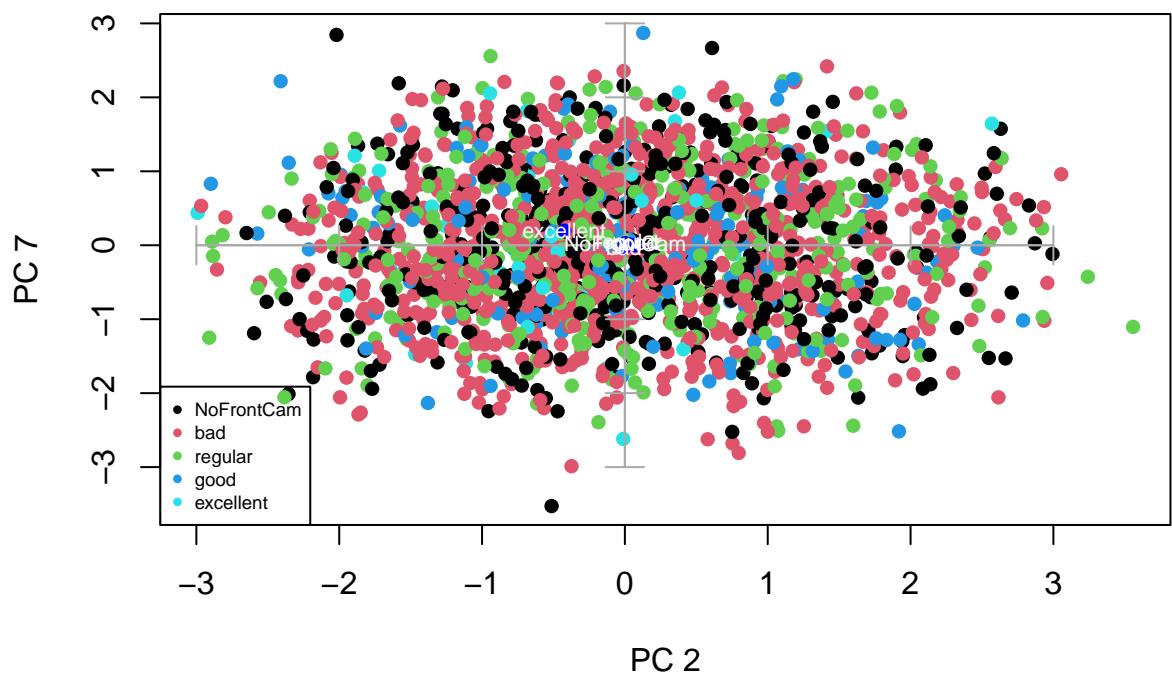


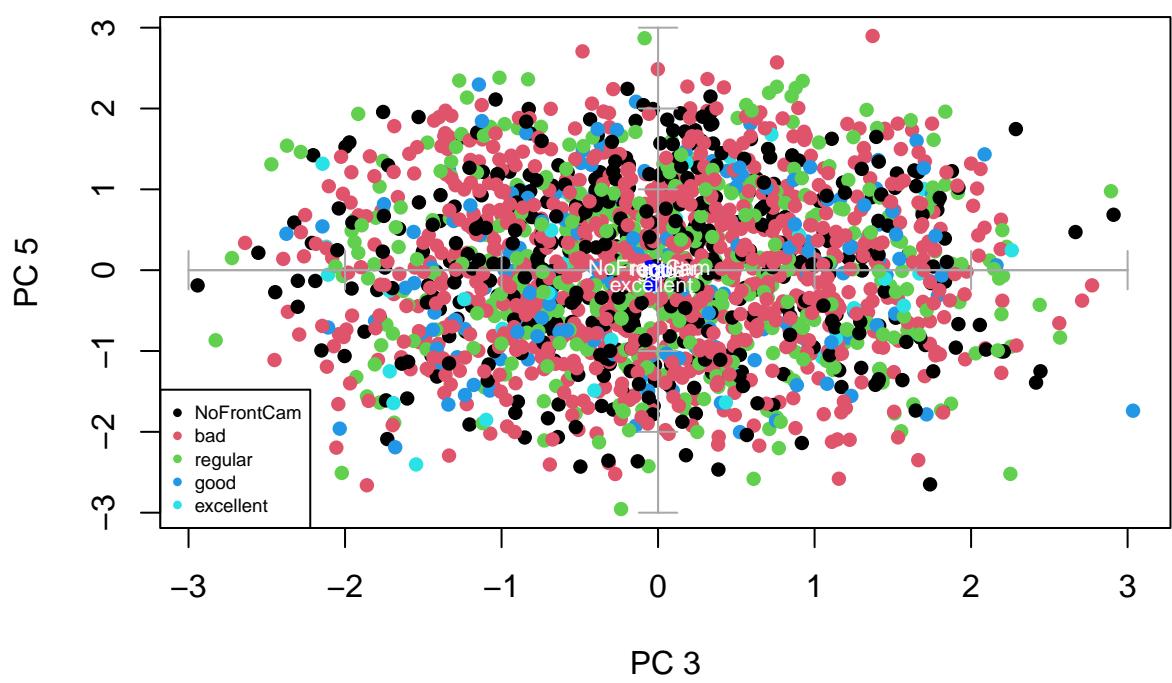
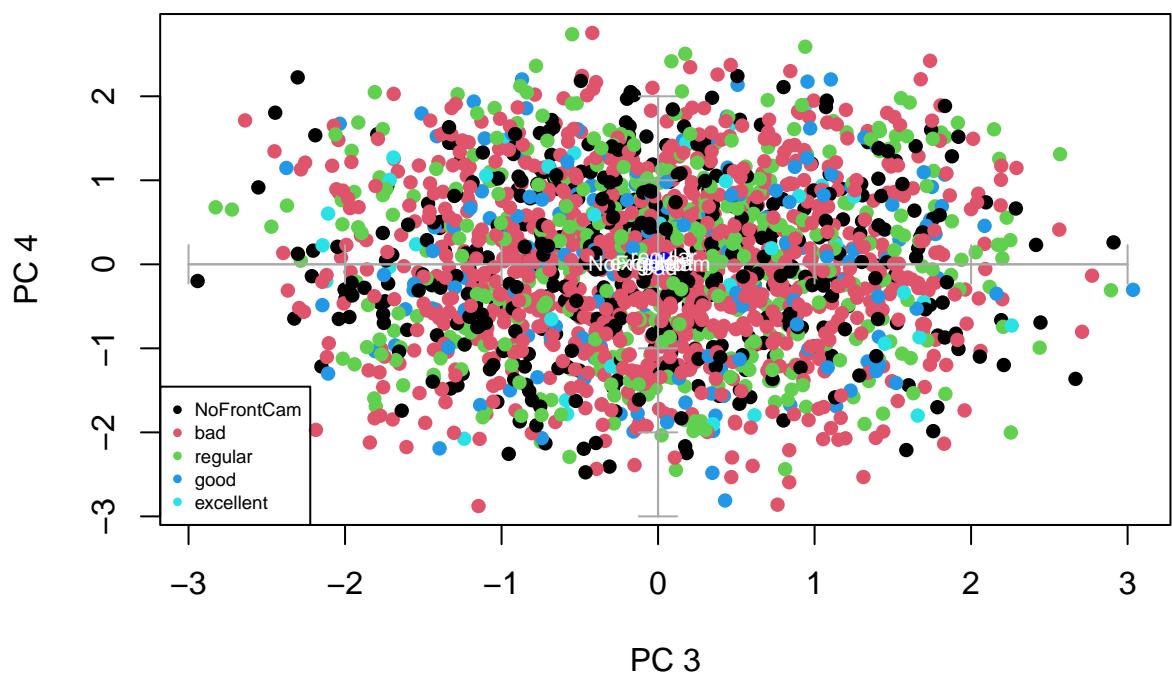


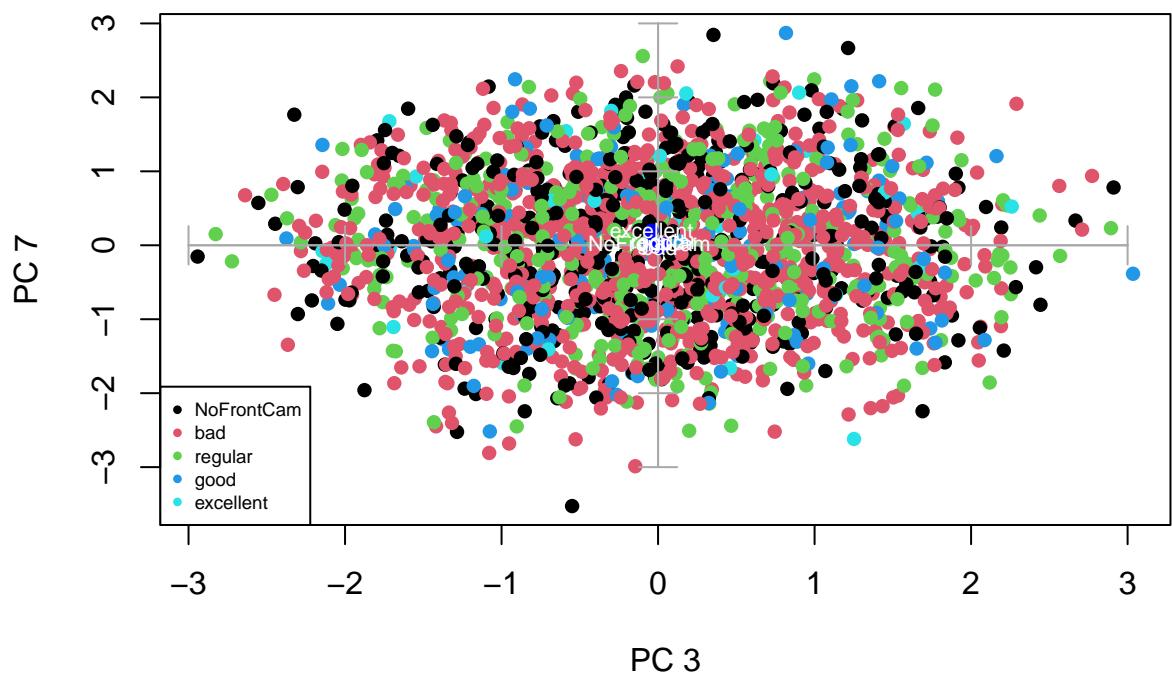
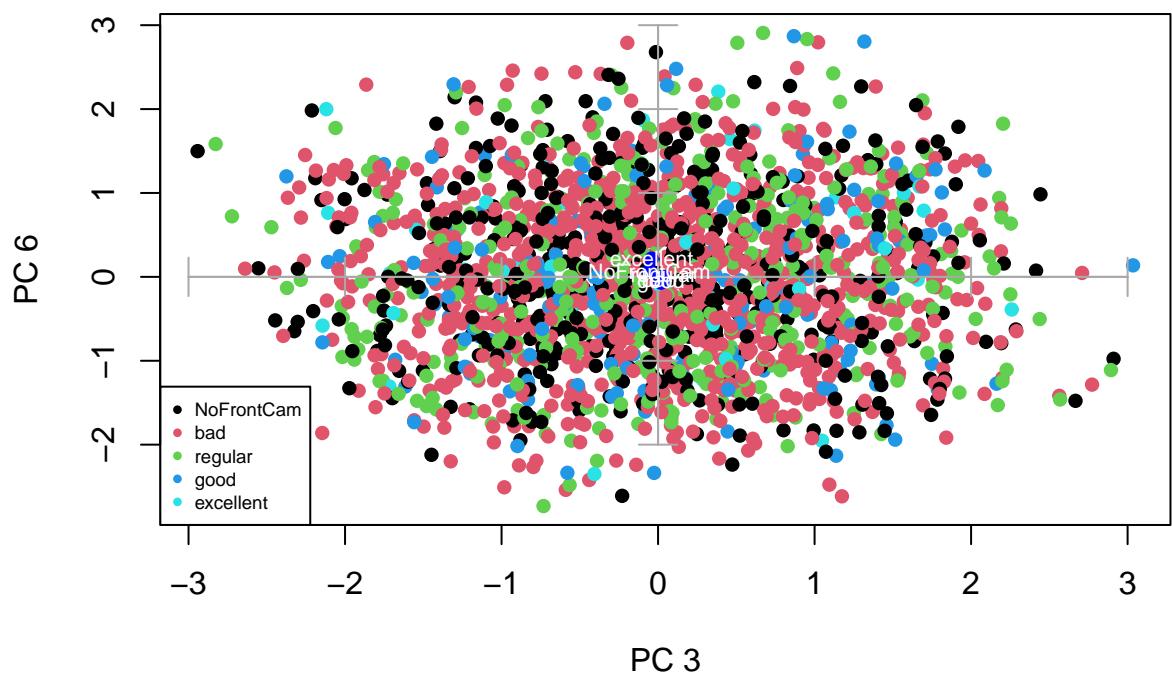


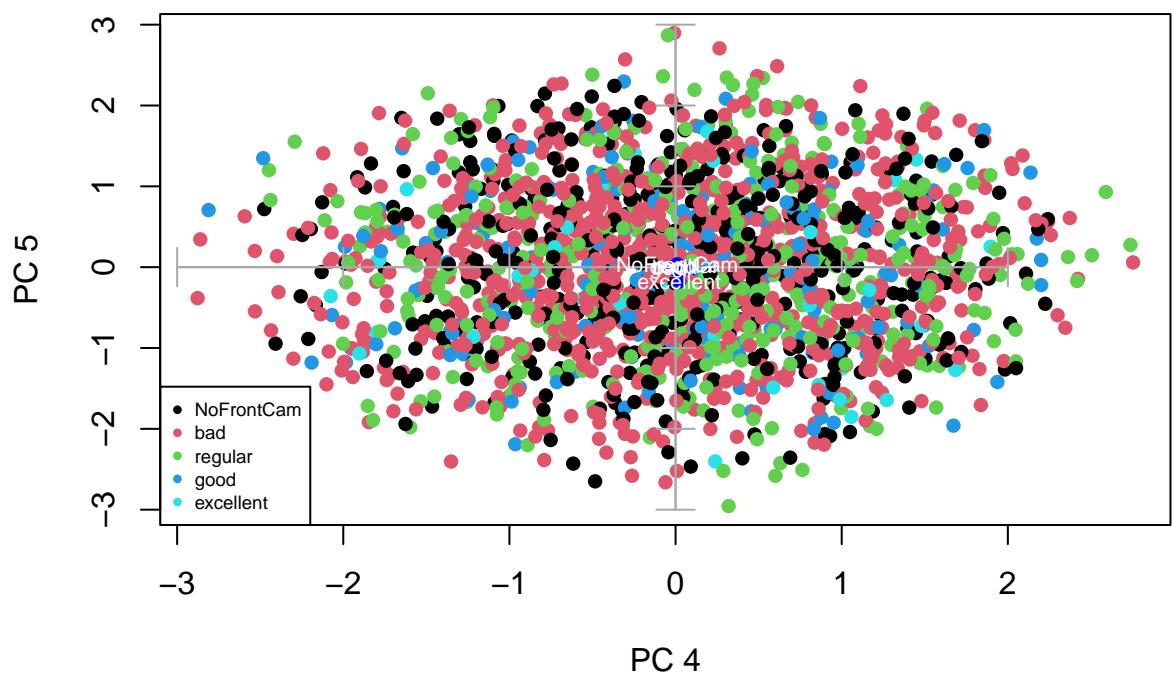
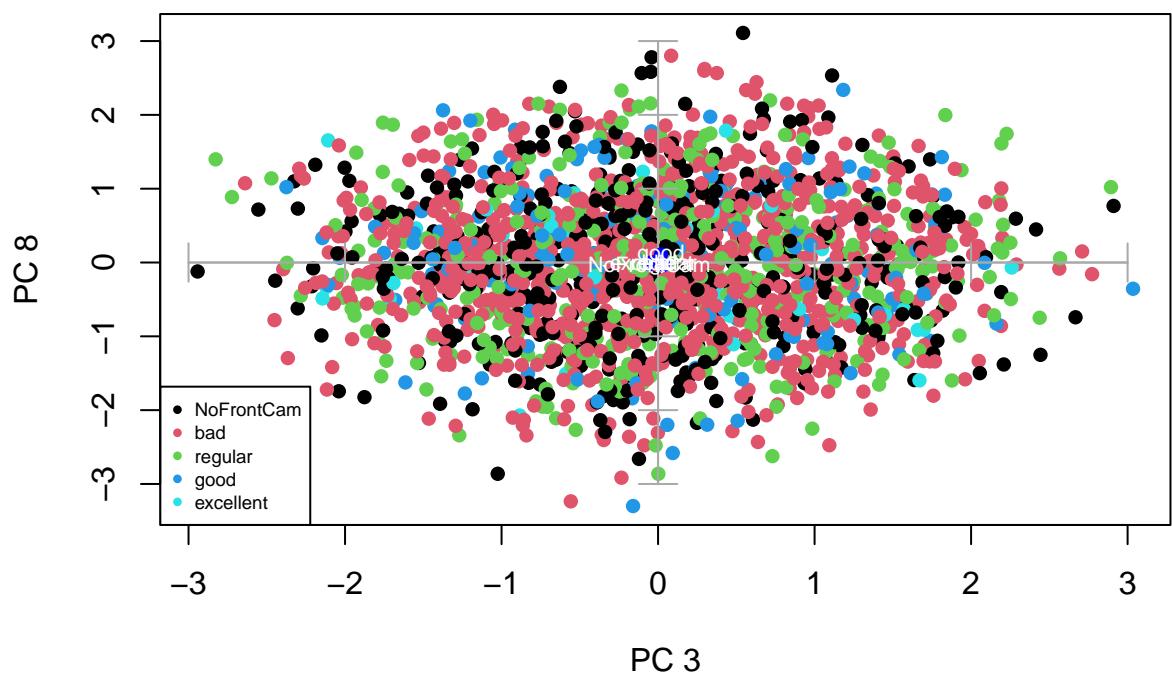


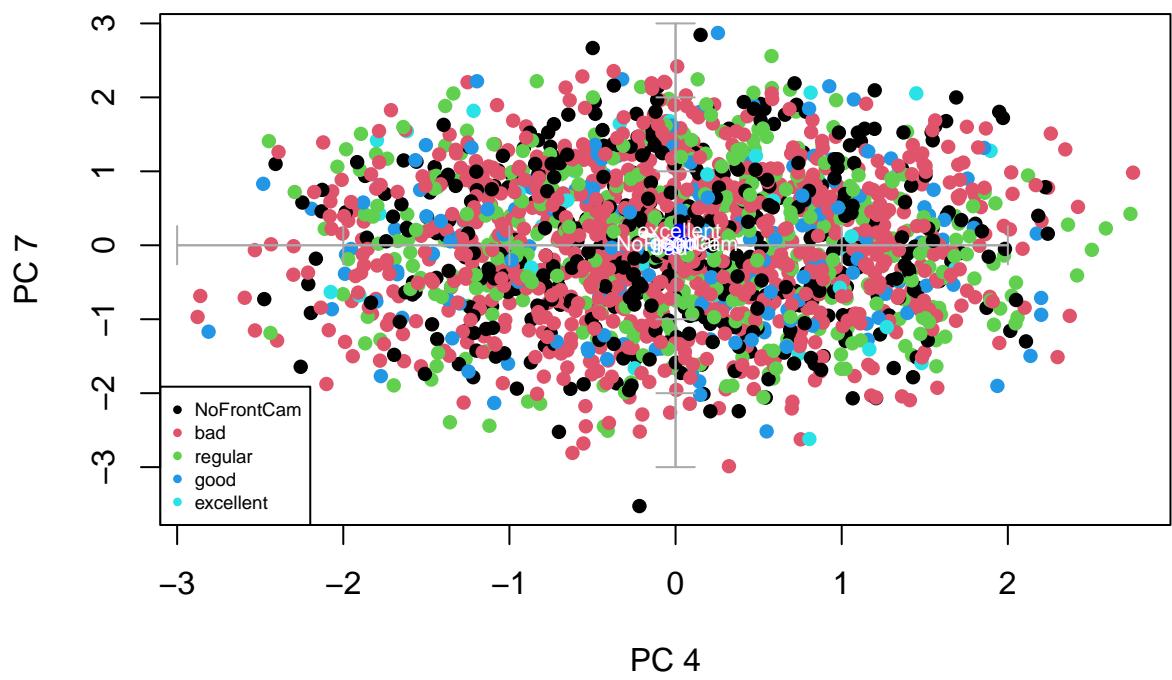
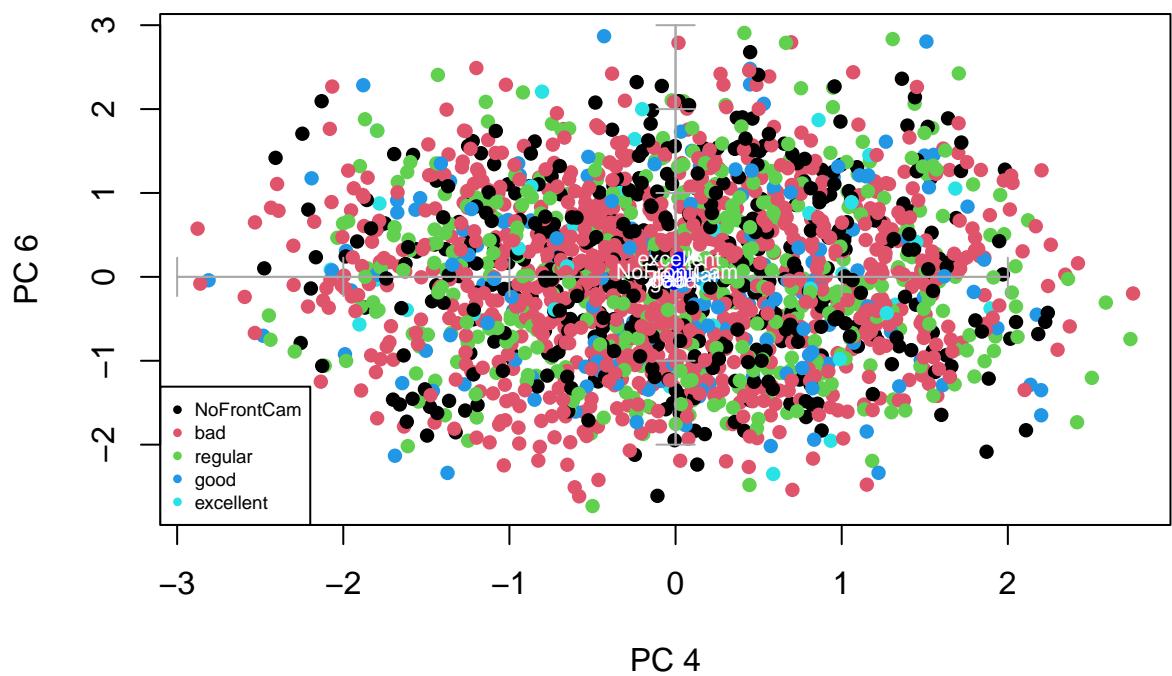


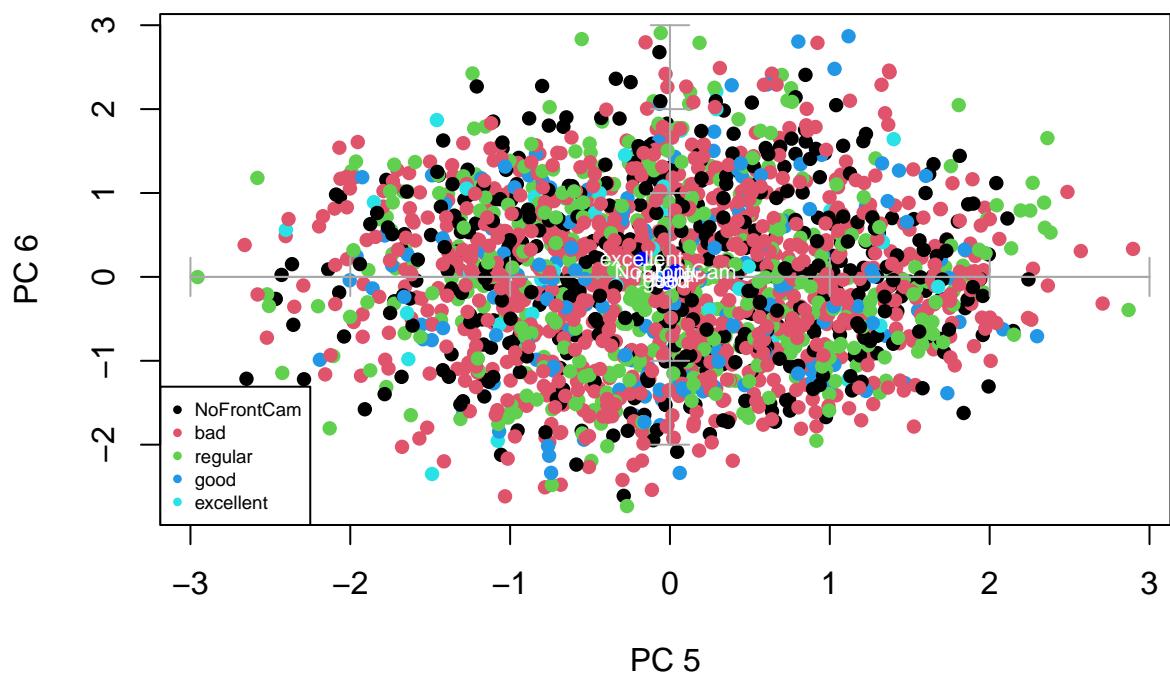
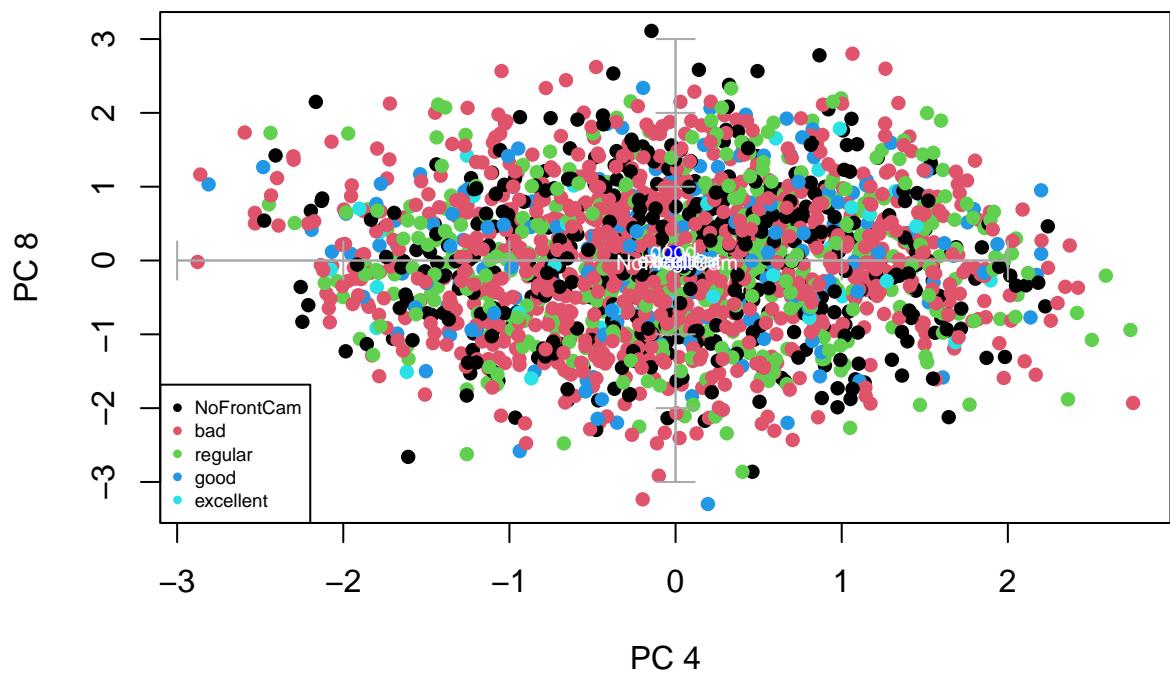


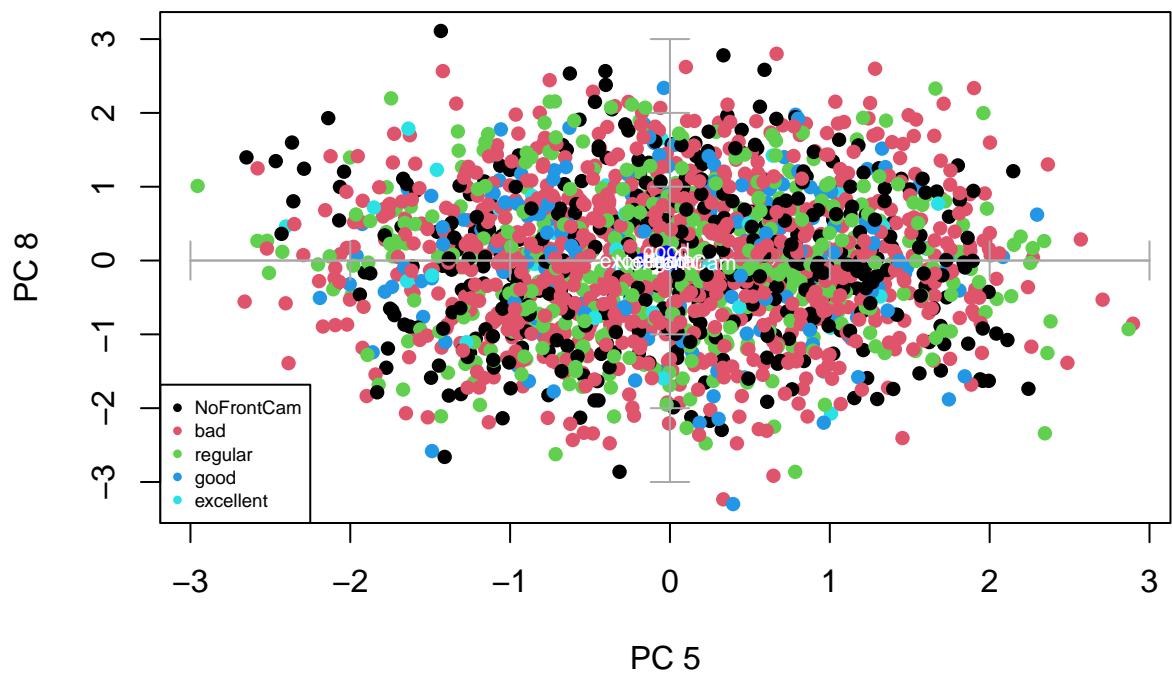
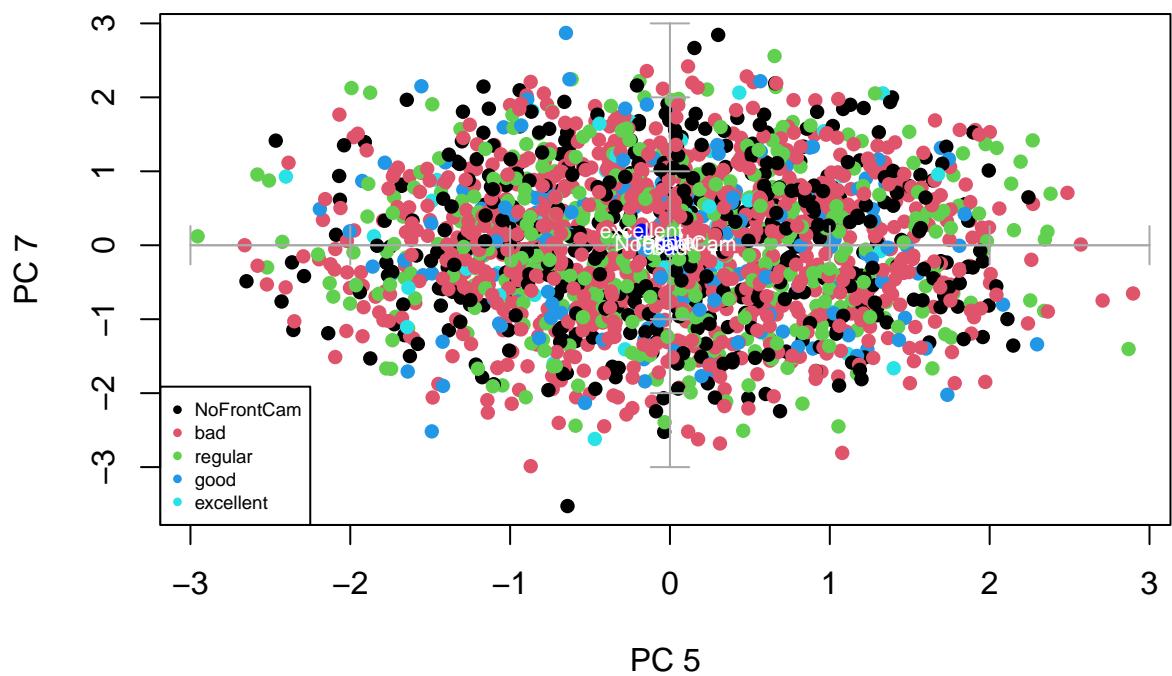


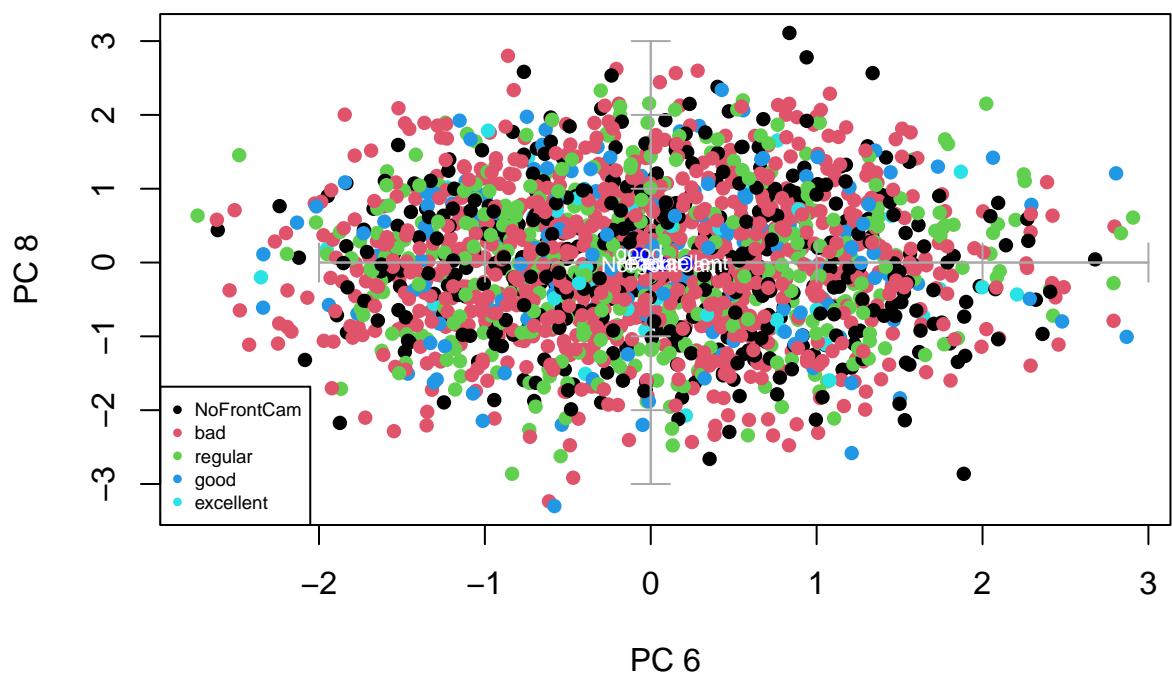
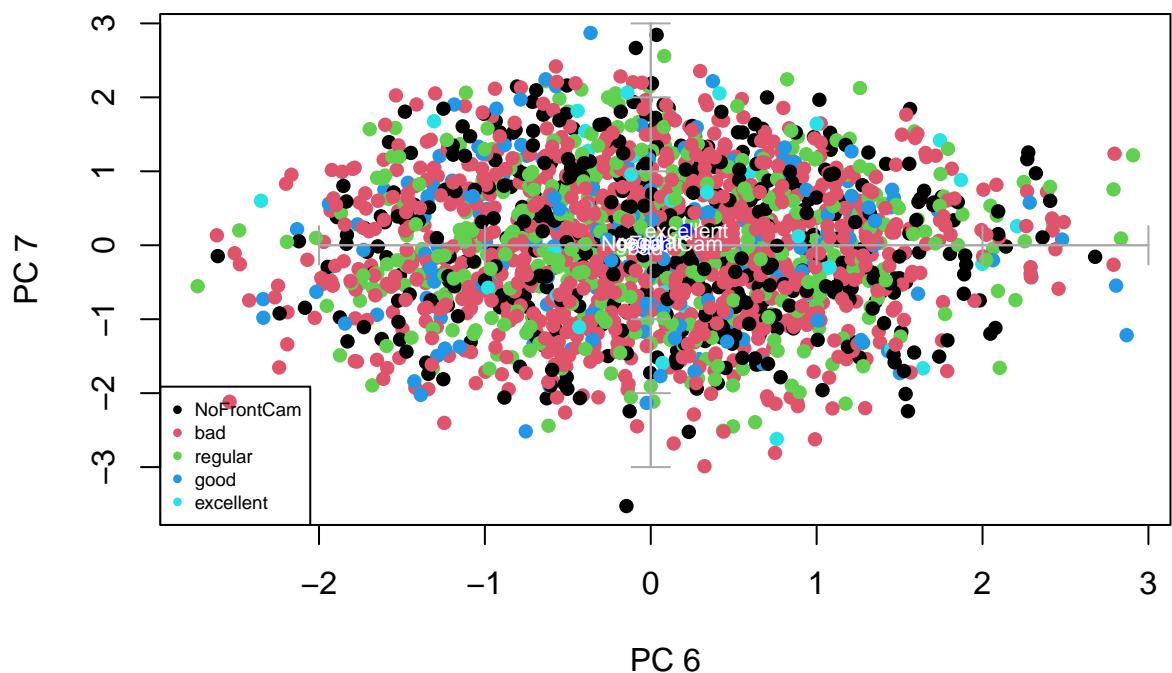


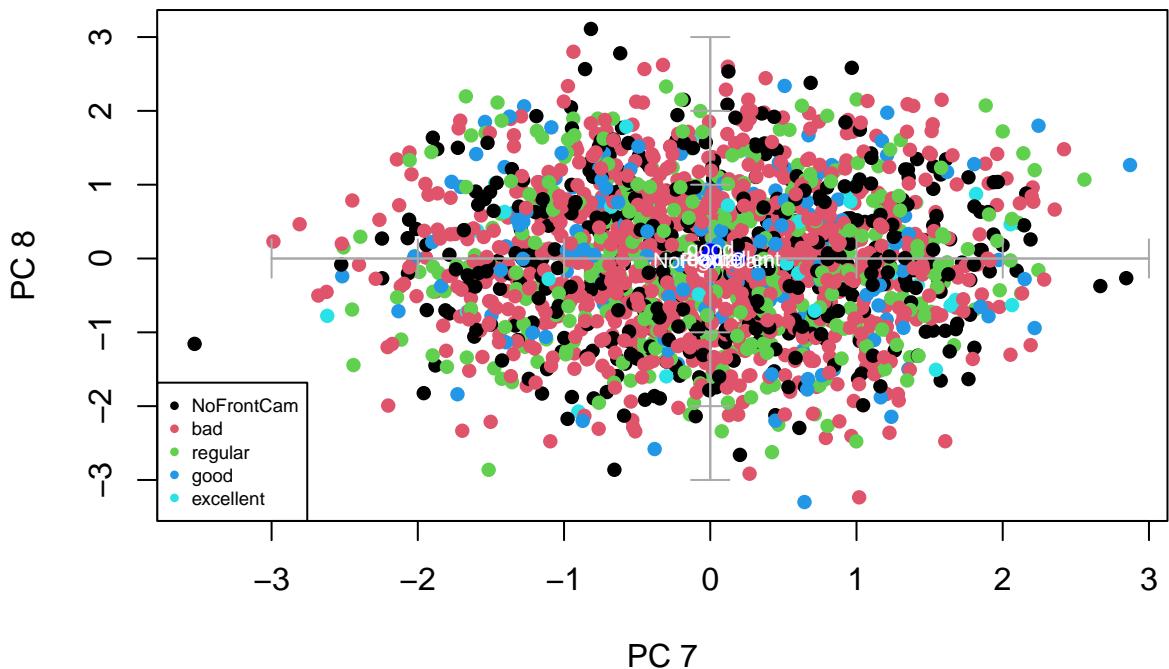










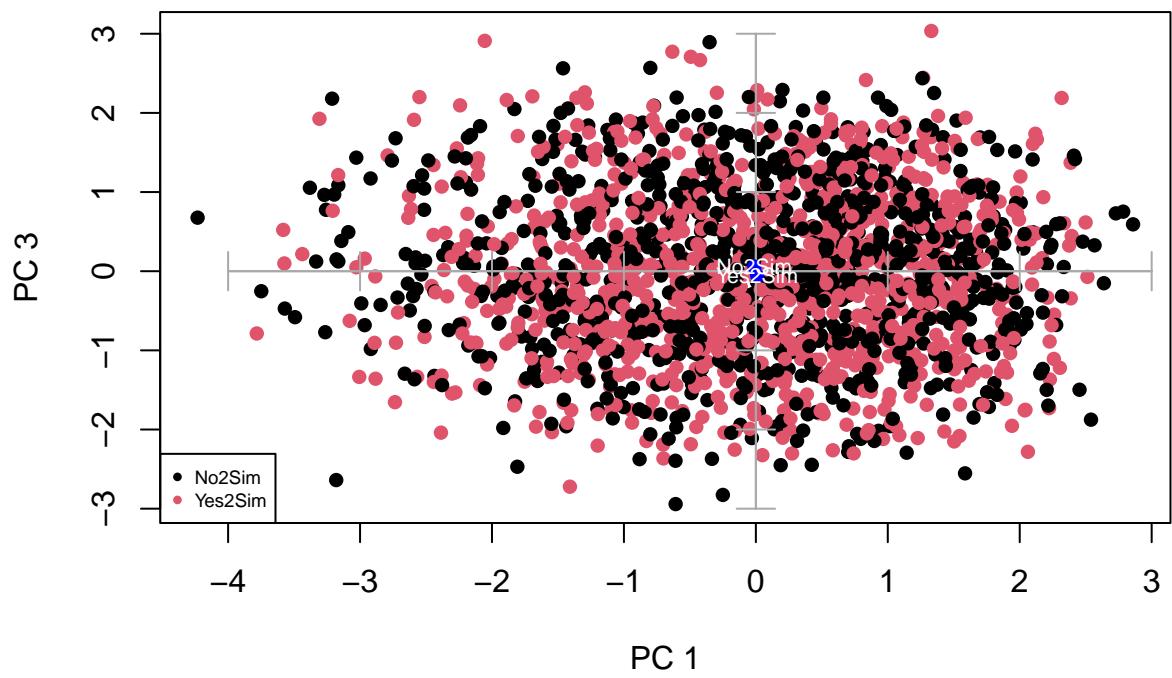
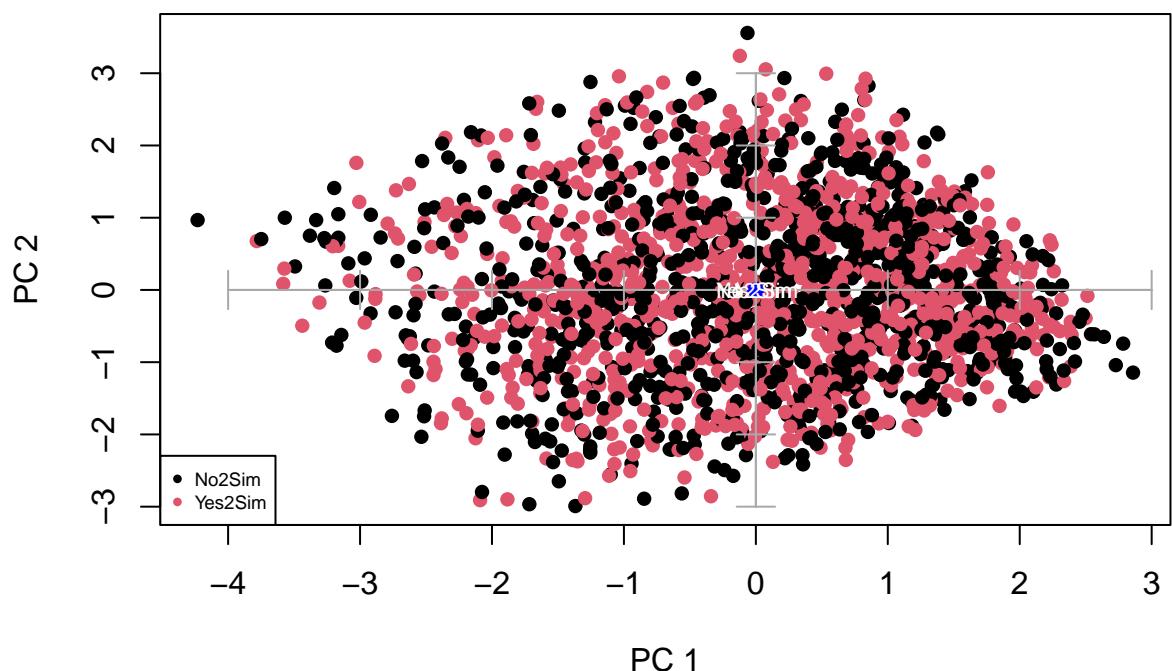


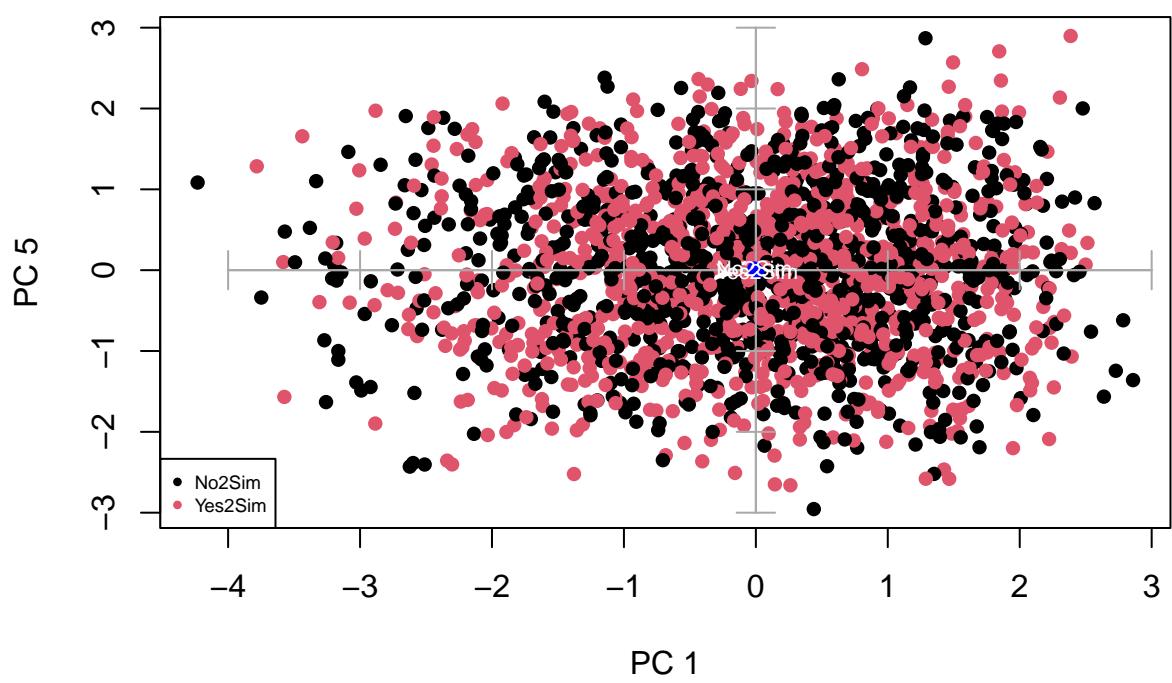
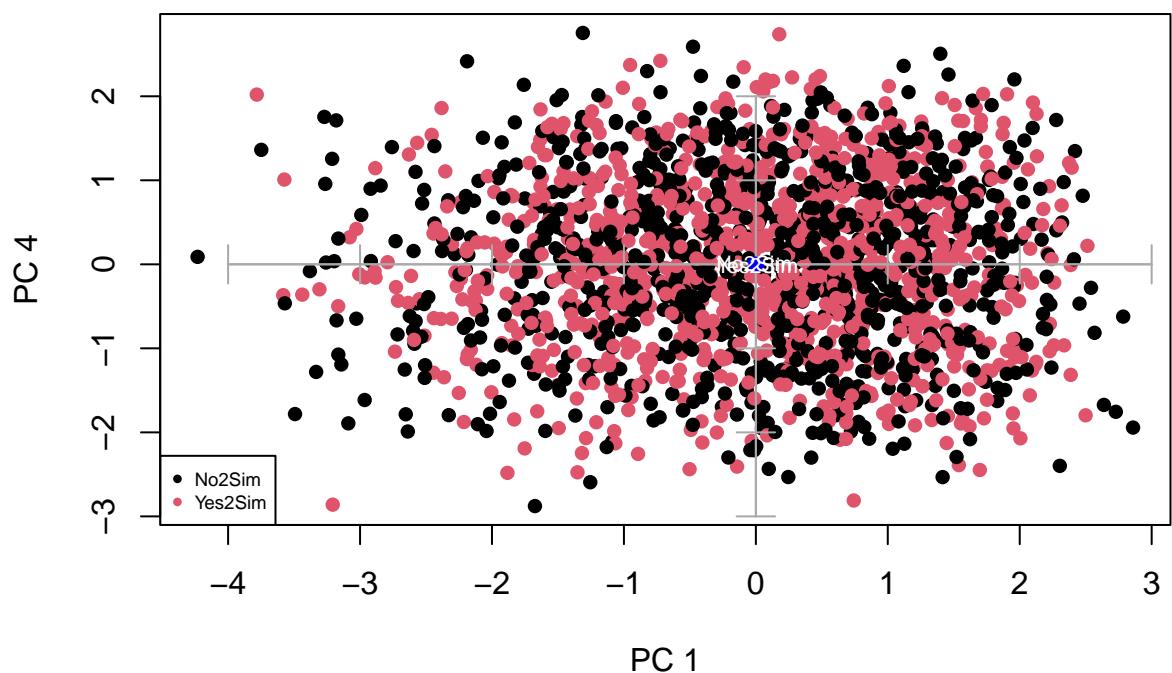
```

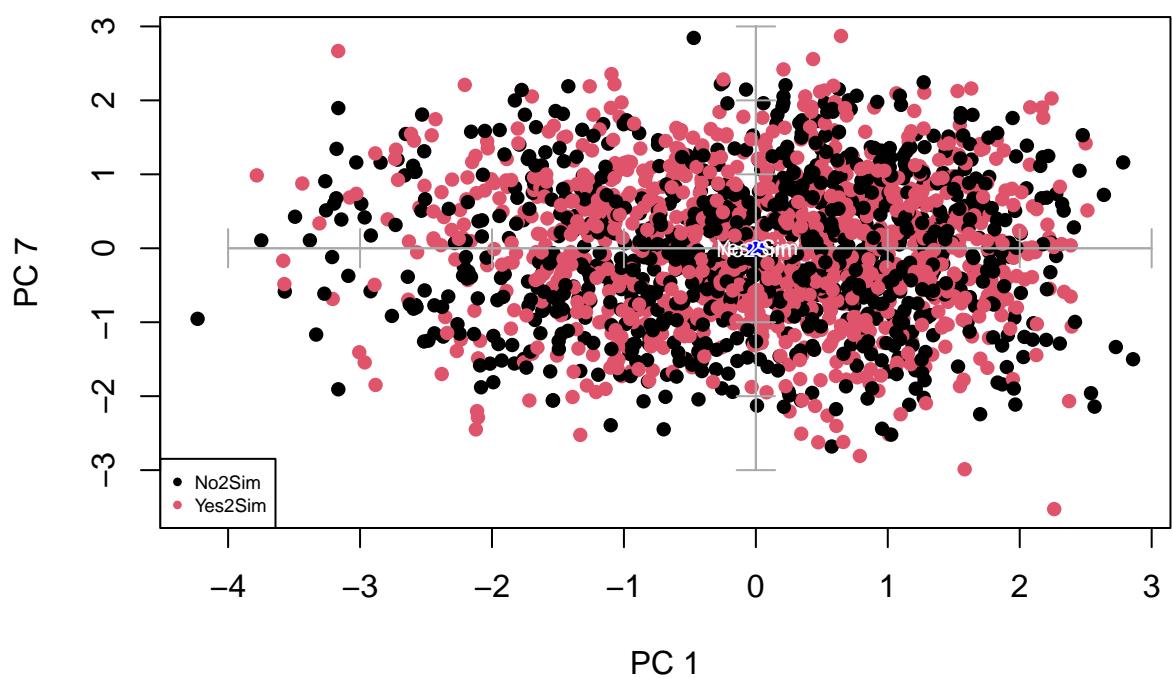
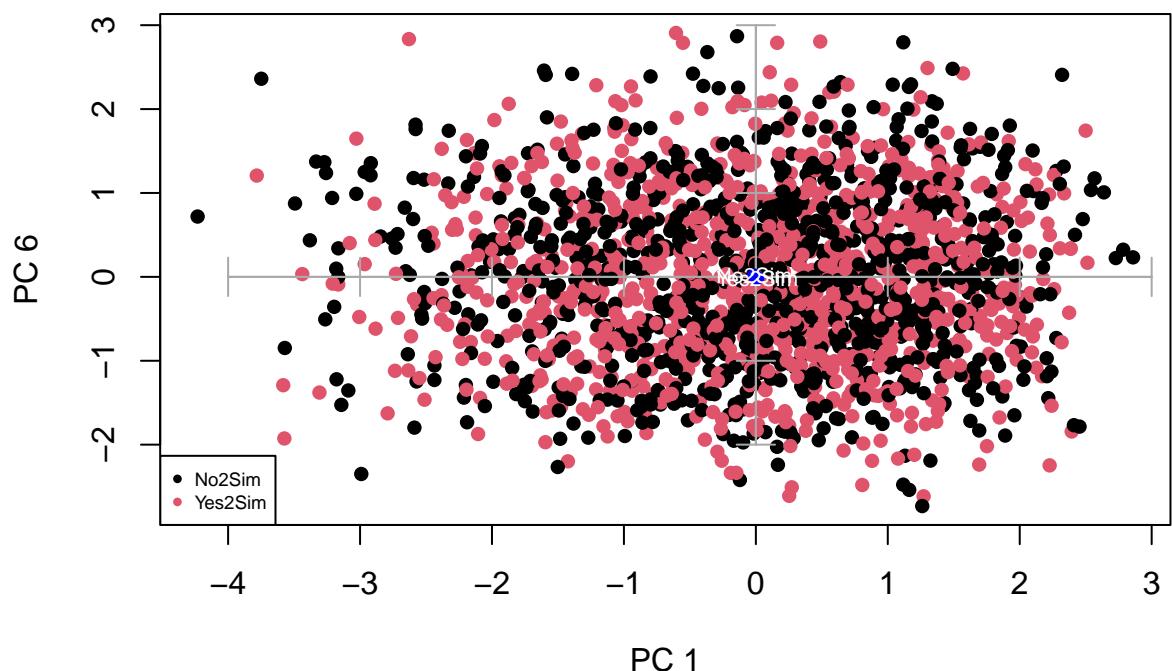
for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$dual_sim
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
    legend("bottomleft",levels(varcat),pch=16,col=c(1:2) , cex=0.6)

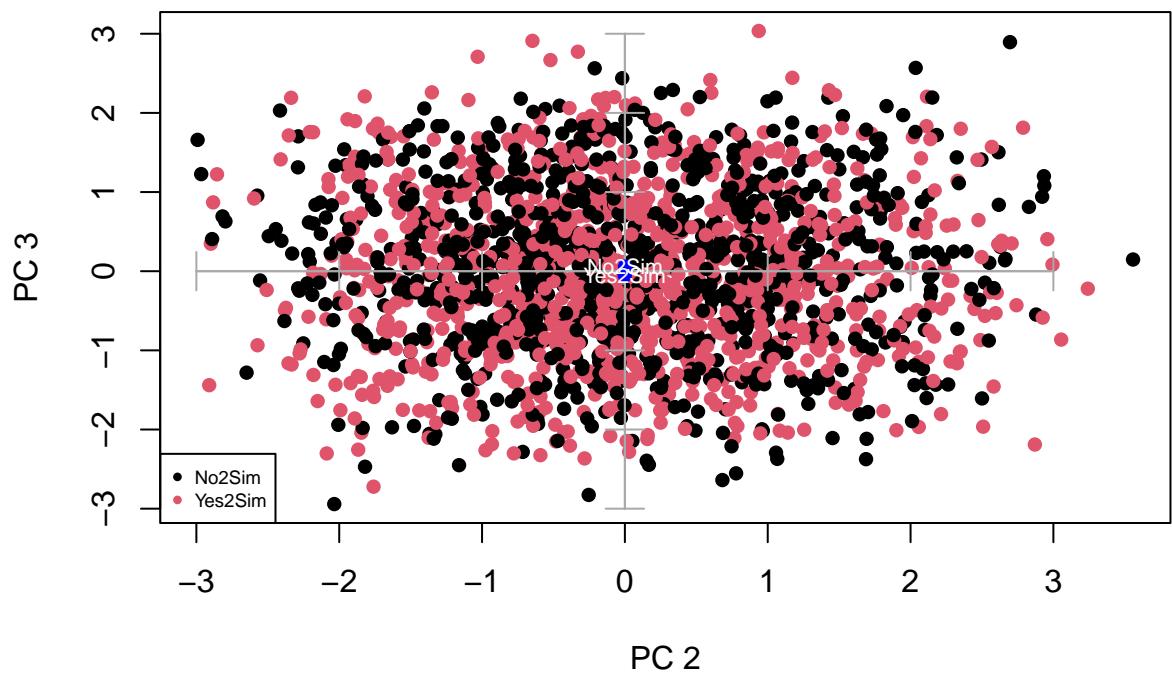
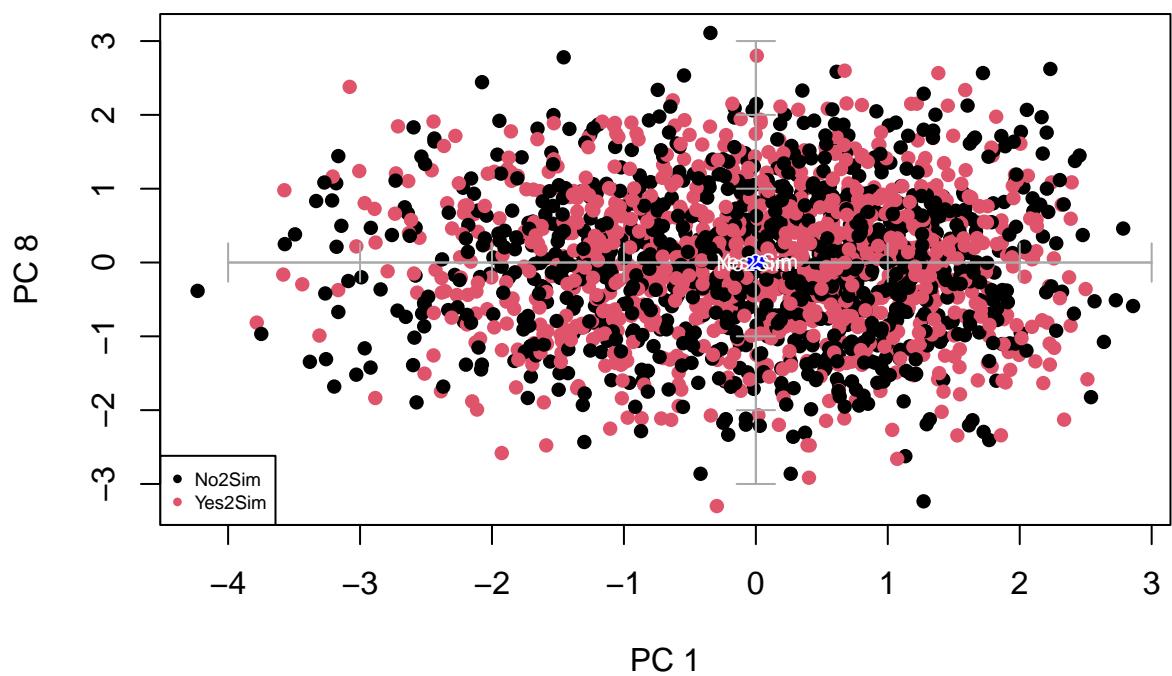
    #select your qualitative variable
    fdic1 = tapply(Psi[,i],varcat,mean)
    fdic2 = tapply(Psi[,j],varcat,mean)
    points(fdic1,fdic2,pch=16,col="blue")
    text(fdic1,fdic2,labels=levels(varcat),col="white" , cex=0.7)
  }
}

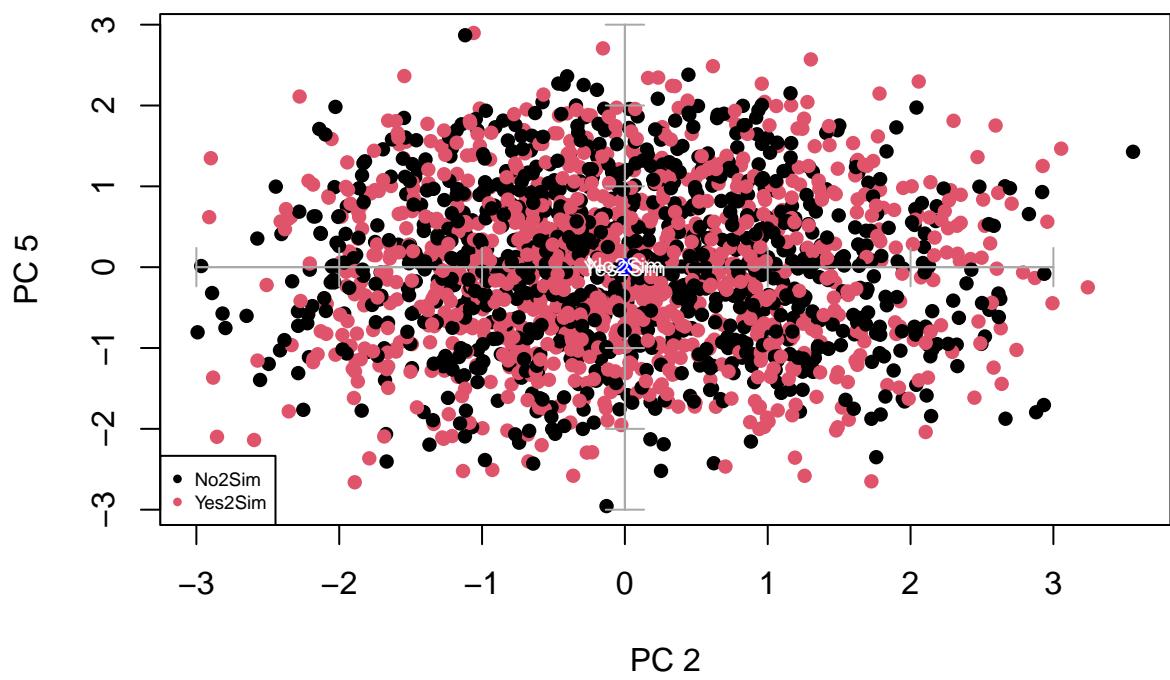
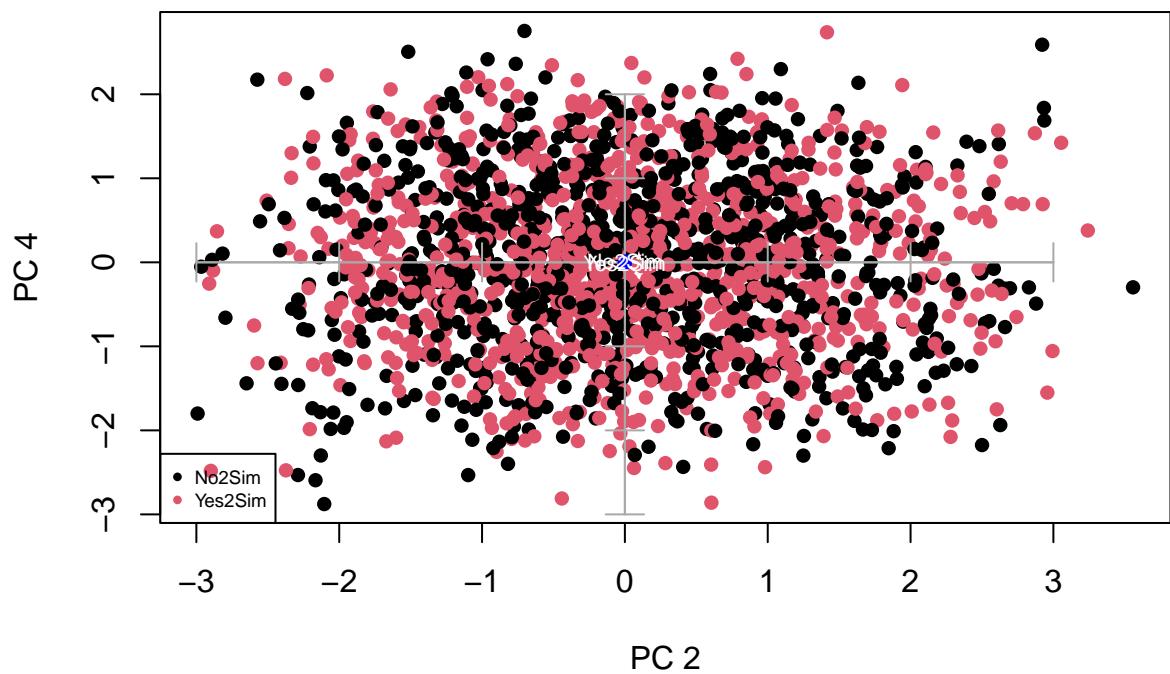
```

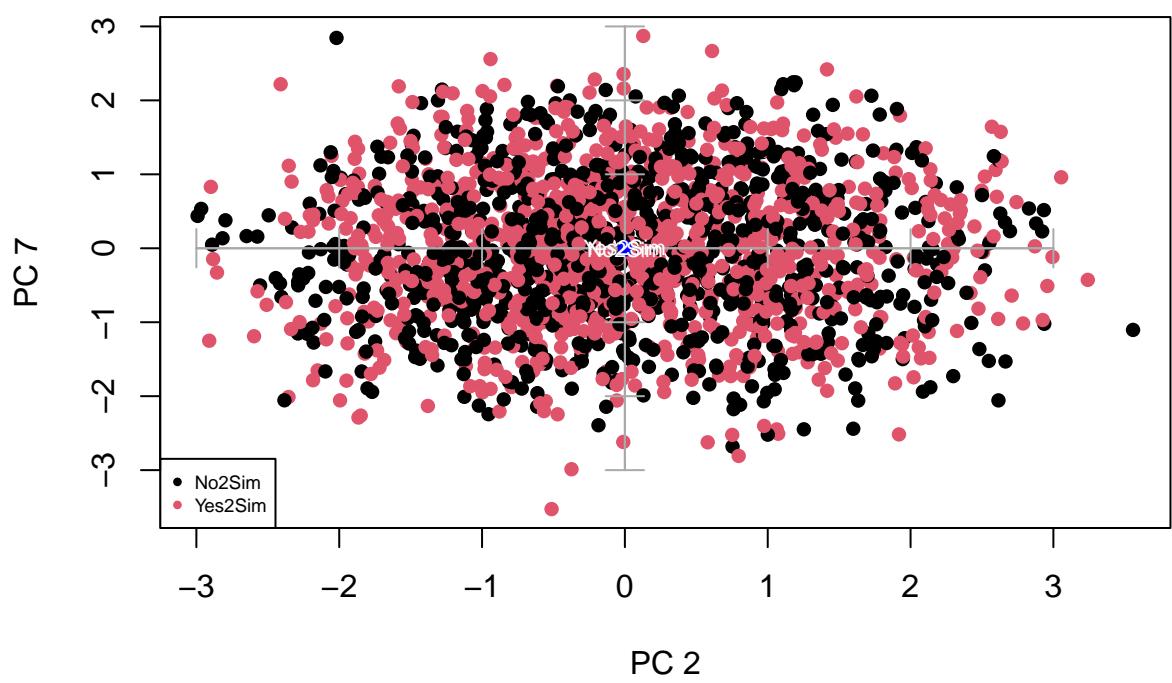
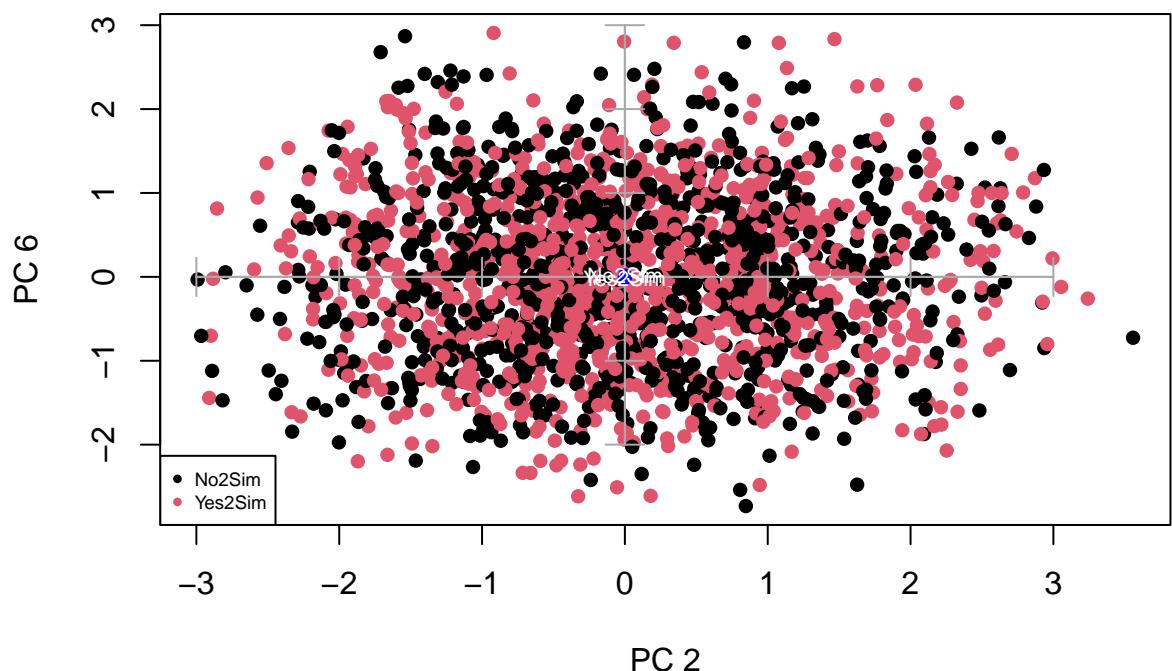


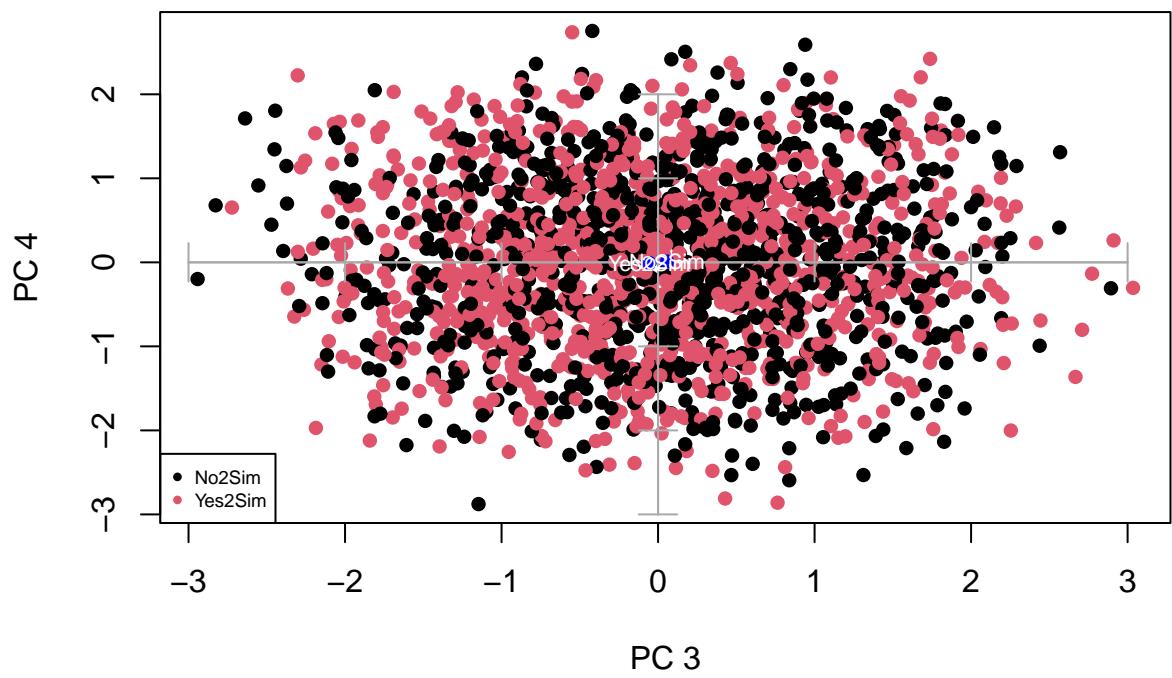
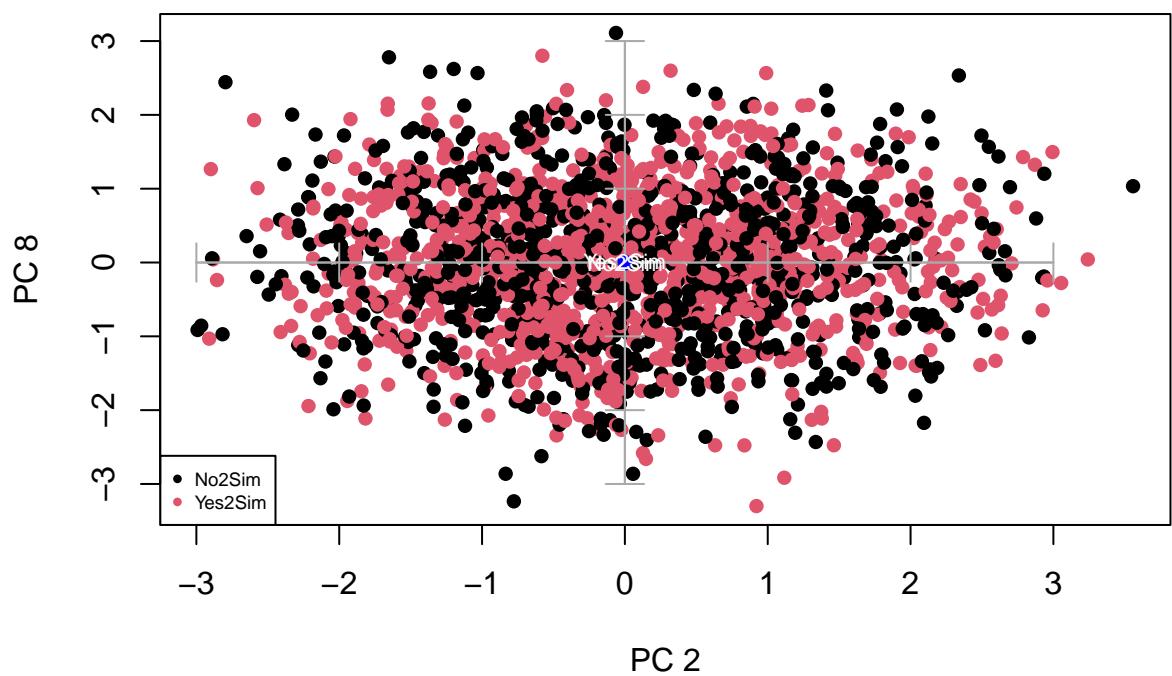


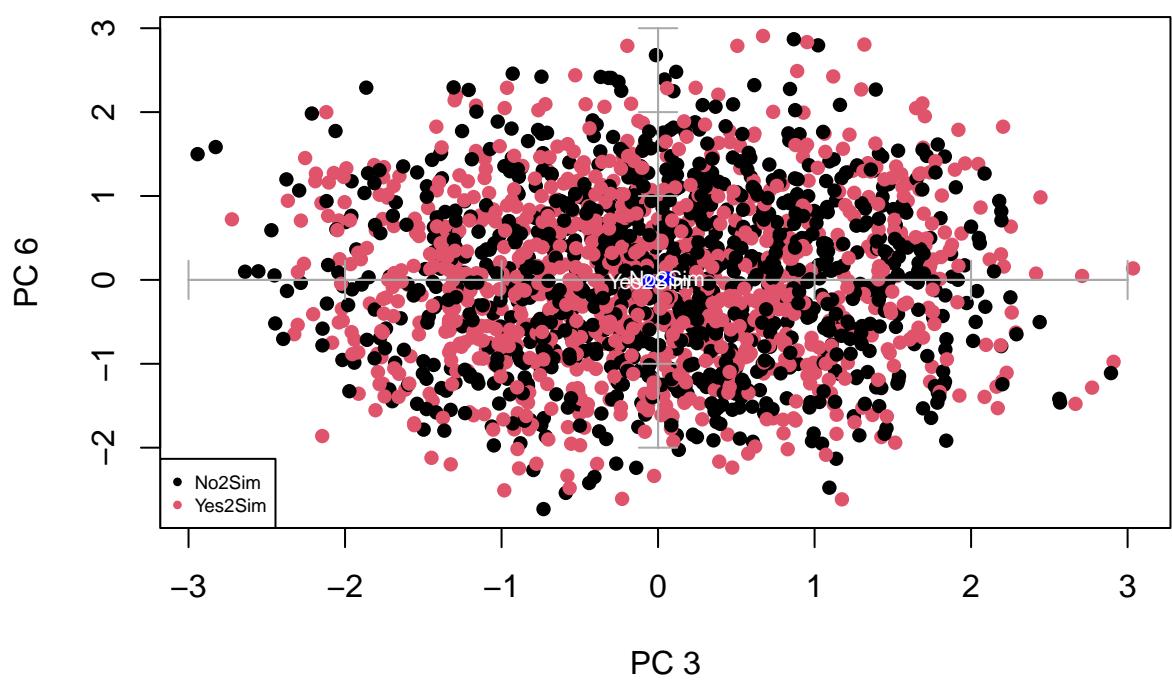
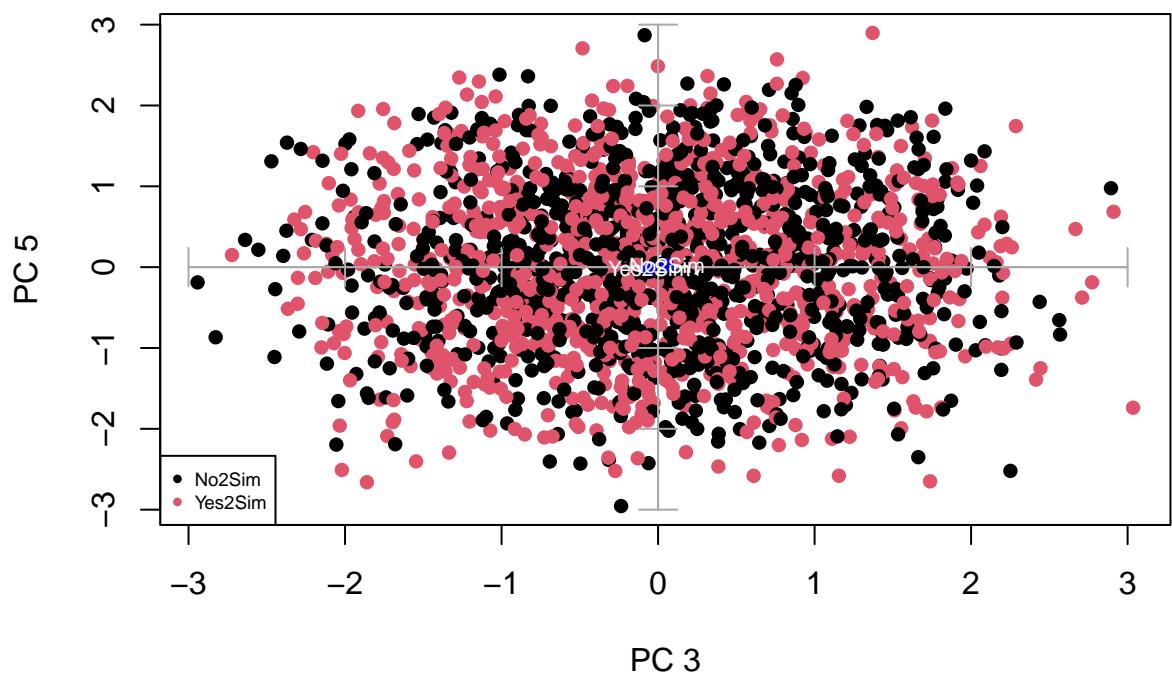


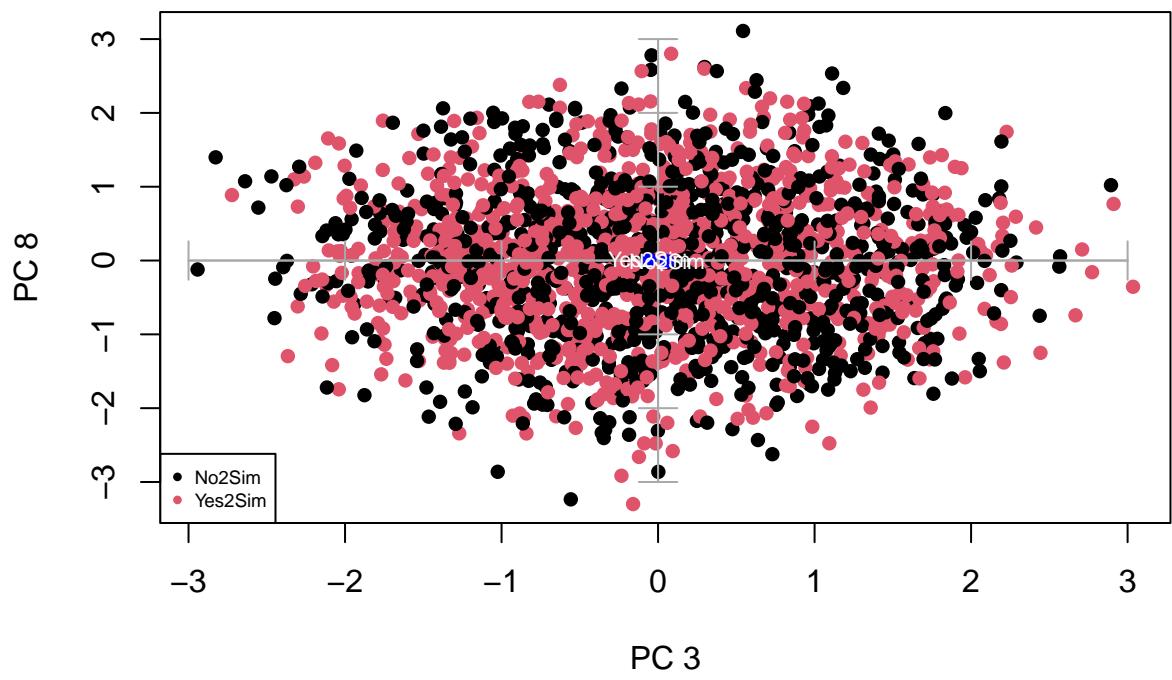
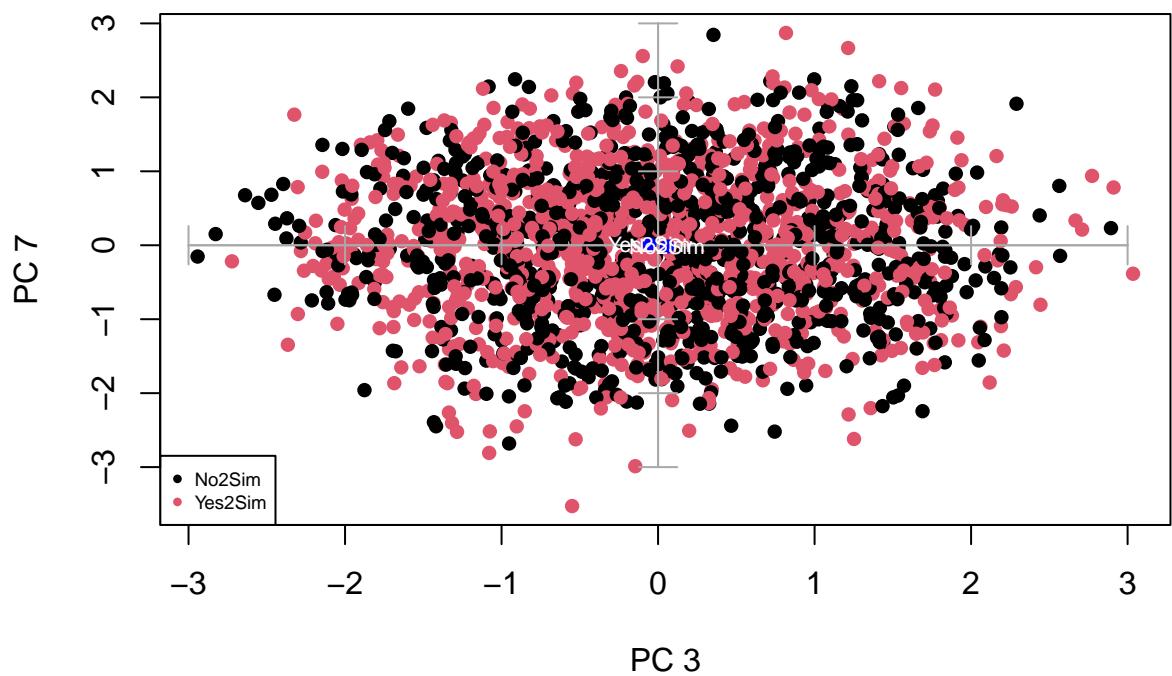


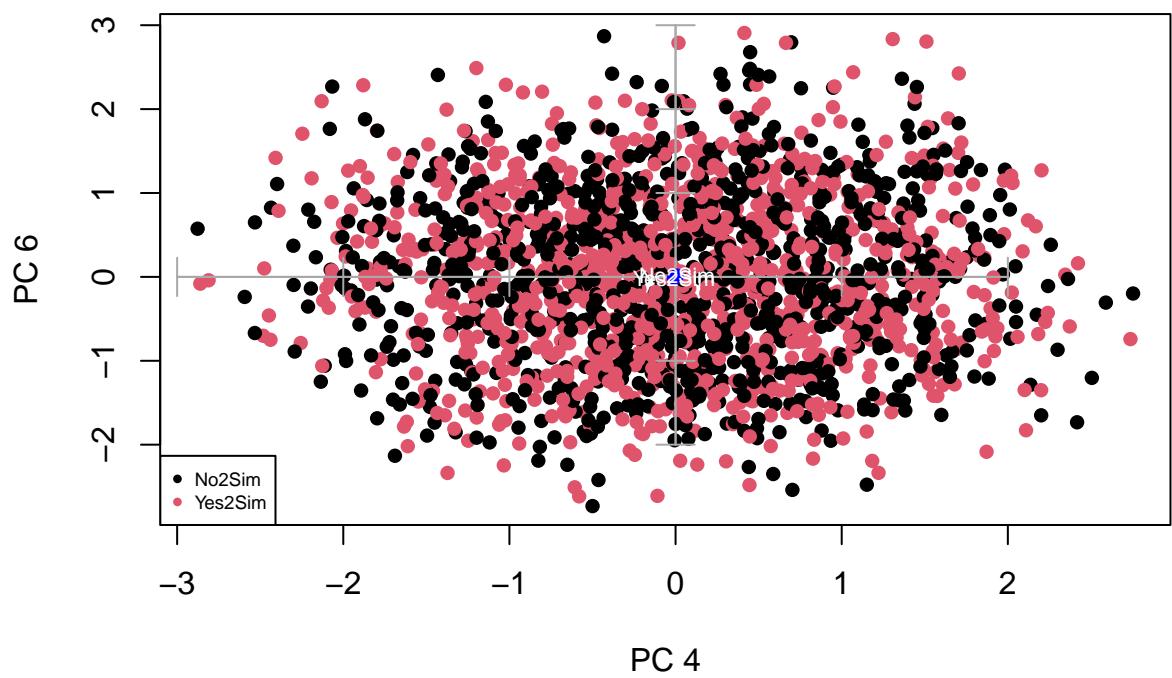
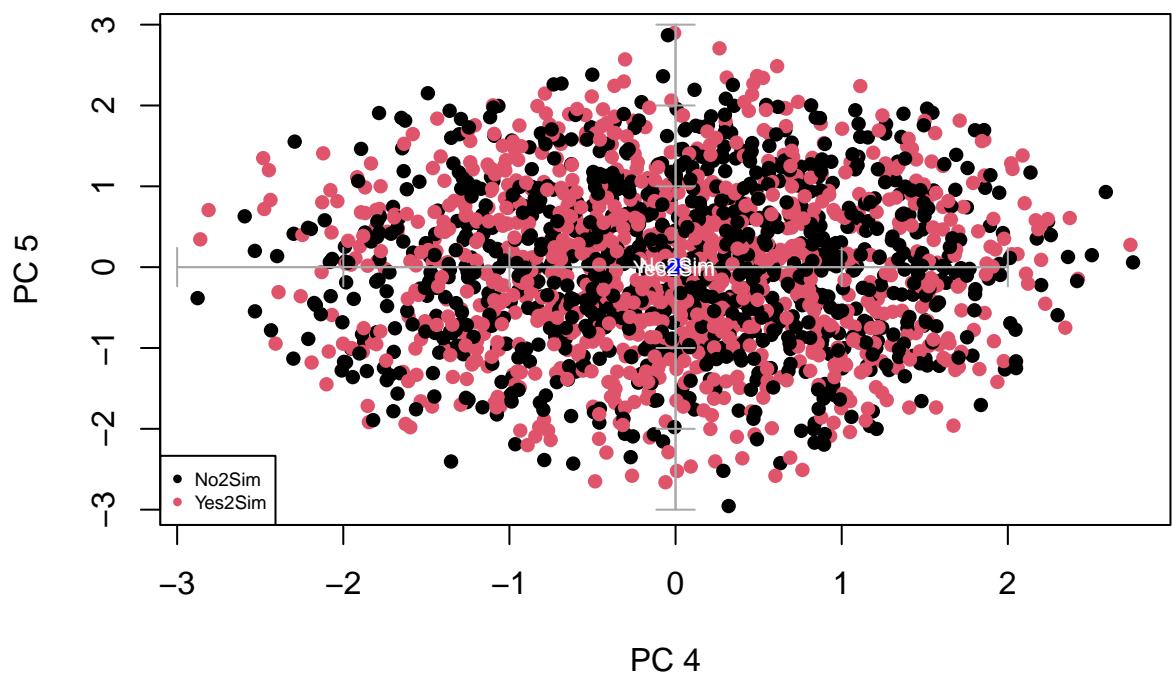


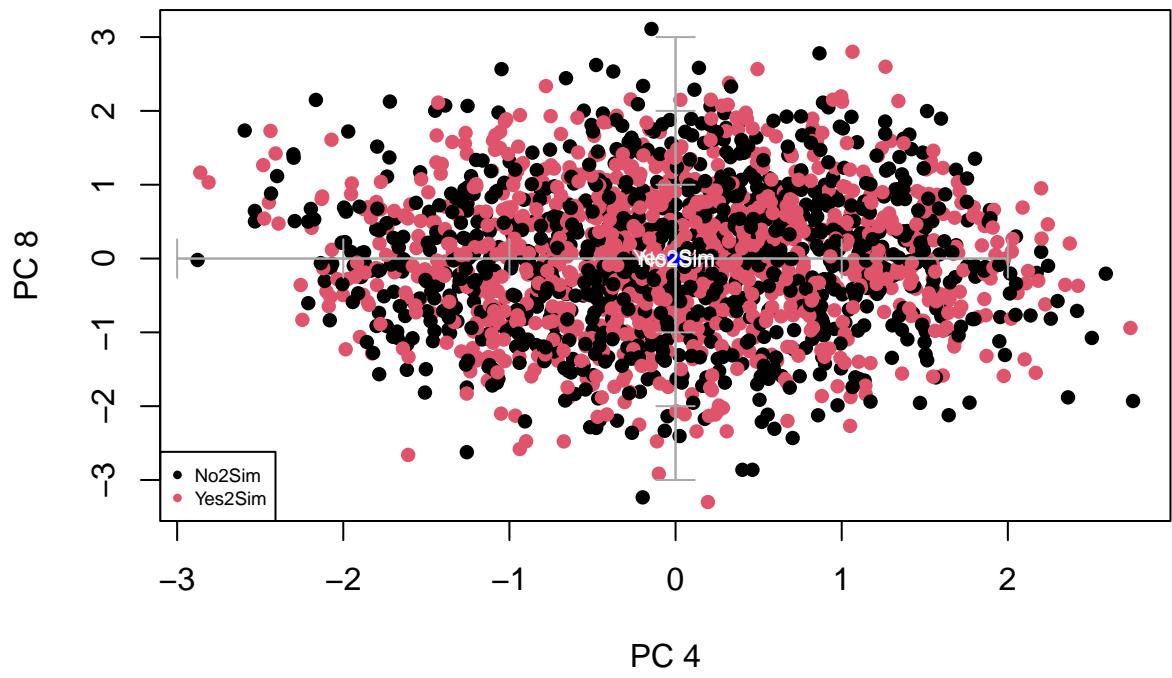
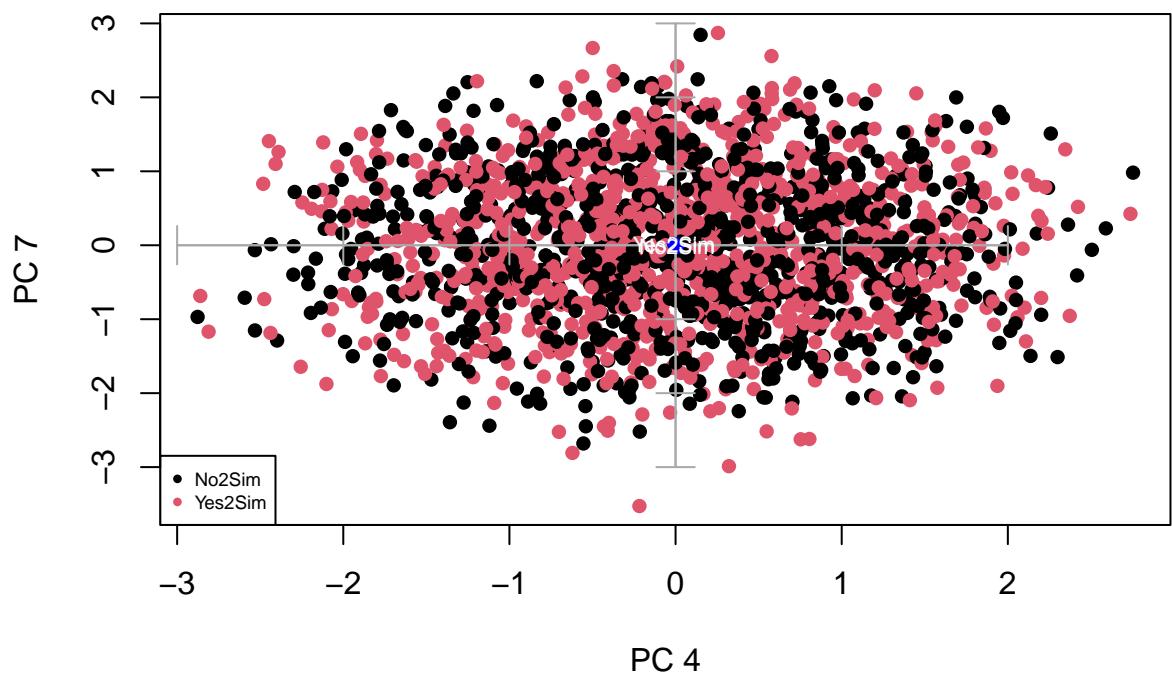


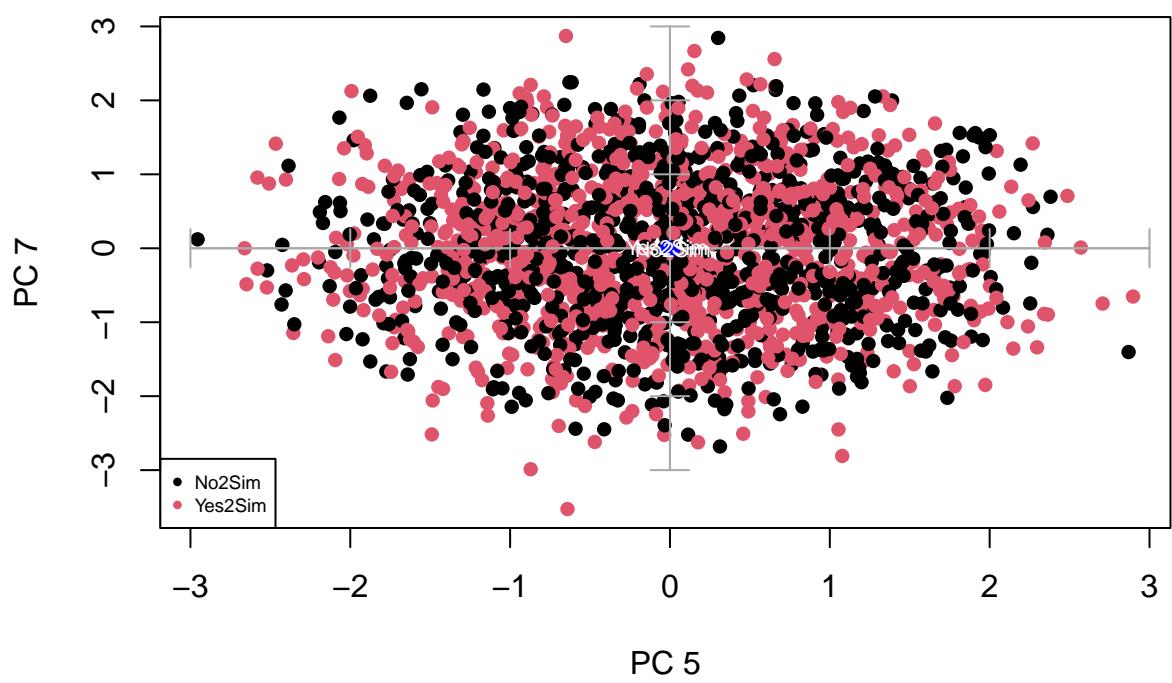
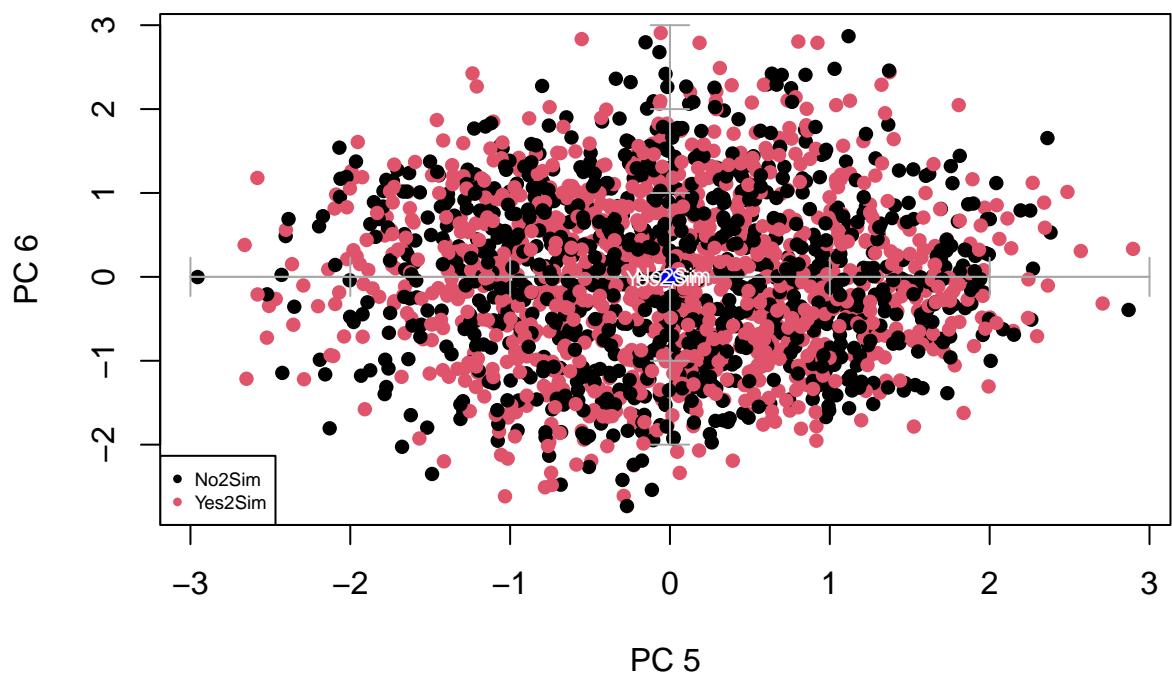


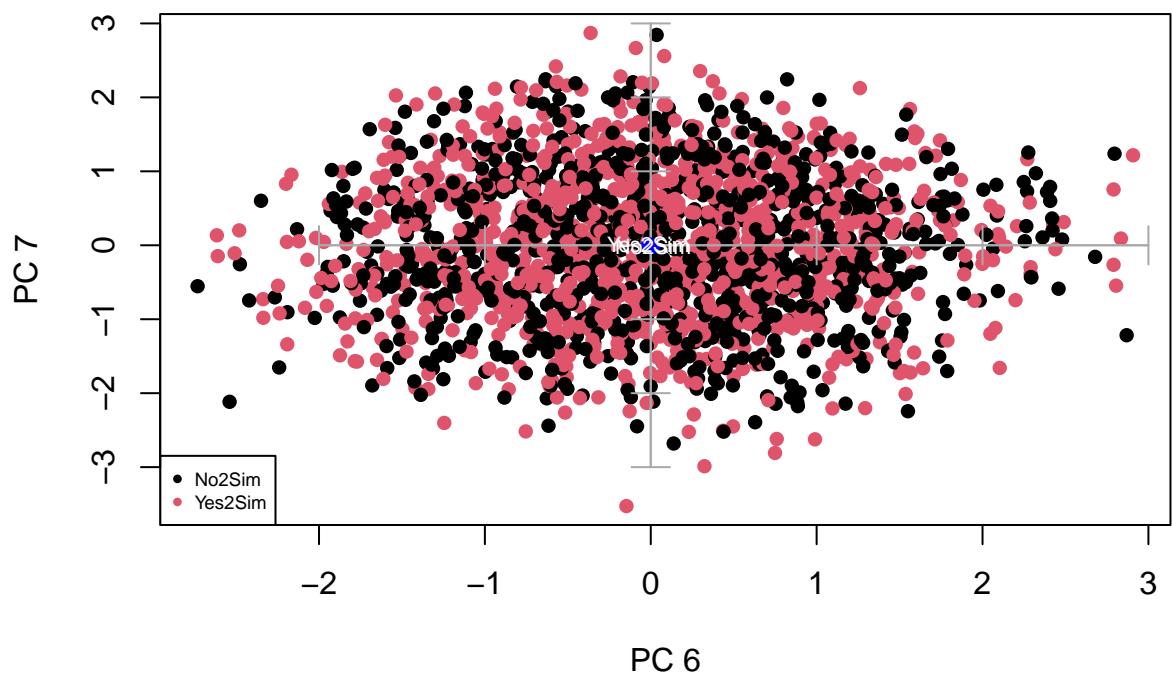
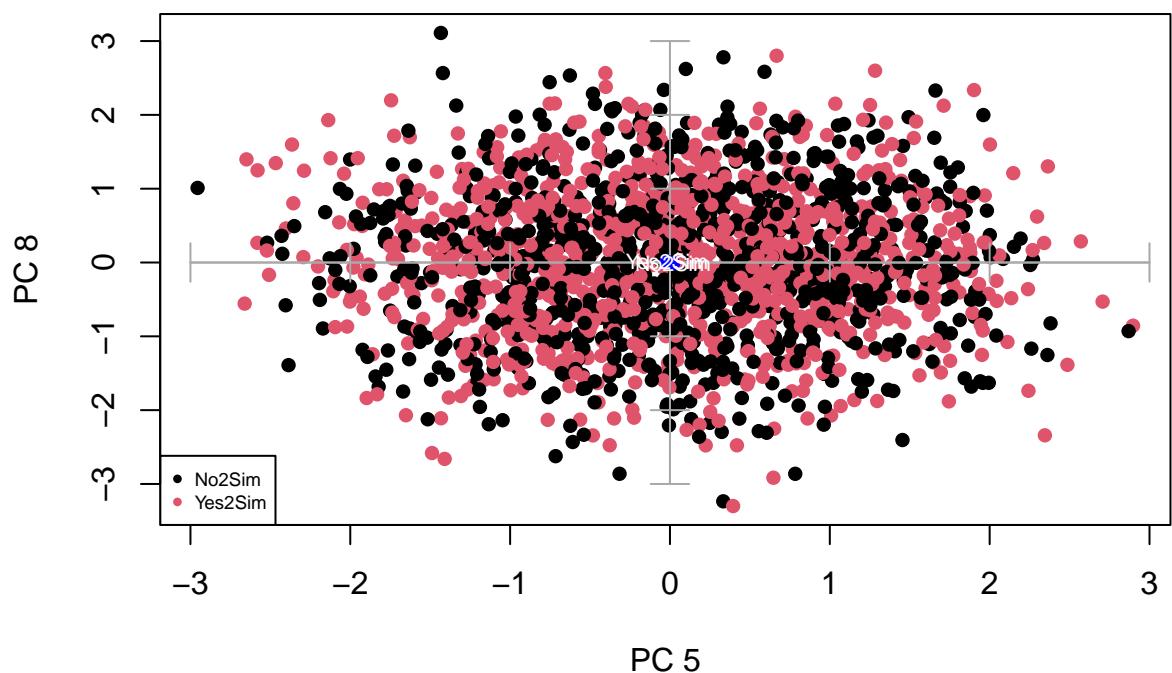


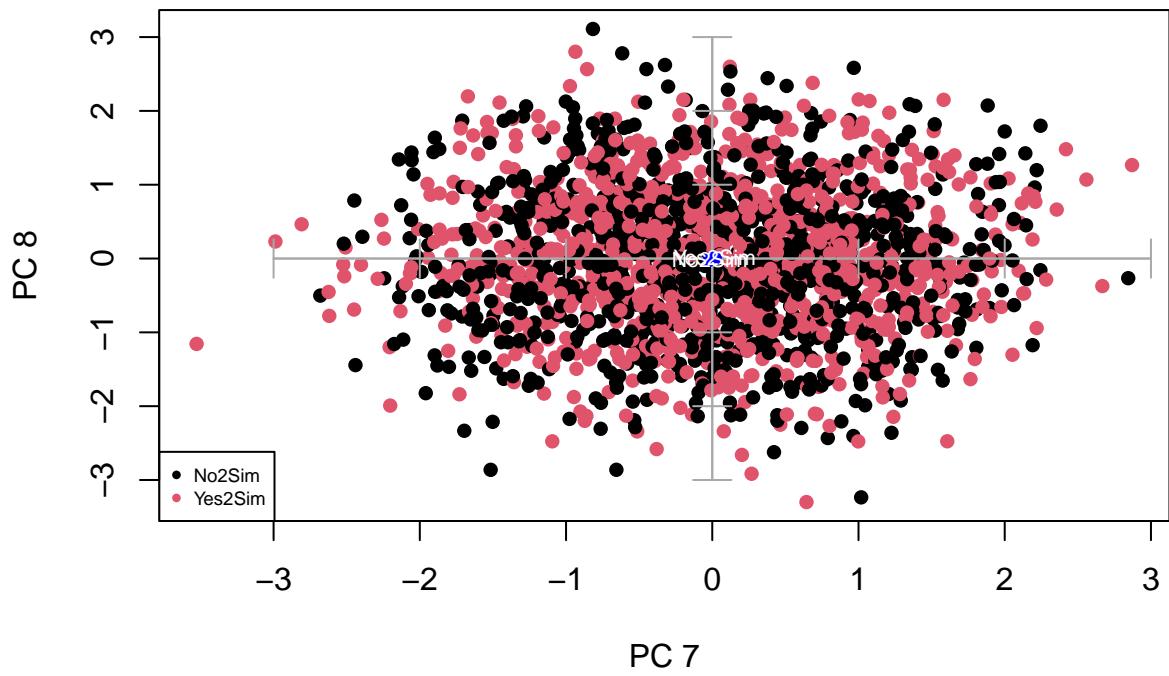
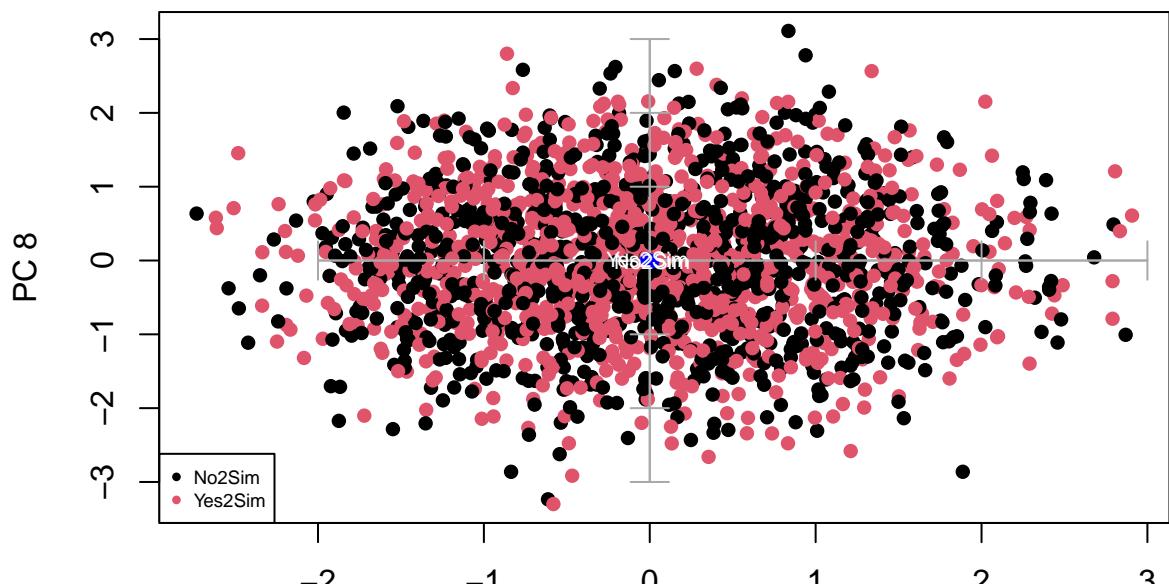












```

for(i in 1:7) {
  for (j in (i+1):8) {
    varcat<-df$blue
    plot(Psi[,i],Psi[,j], col = varcat, pch = 16, xlab=paste("PC",toString(i)) , ylab=paste("PC", toString(j)))
    axis(side=1, pos= 0, labels = F, col="darkgray")
    axis(side=3, pos= 0, labels = F, col="darkgray")
    axis(side=2, pos= 0, labels = F, col="darkgray")
    axis(side=4, pos= 0, labels = F, col="darkgray")
  }
}
  
```

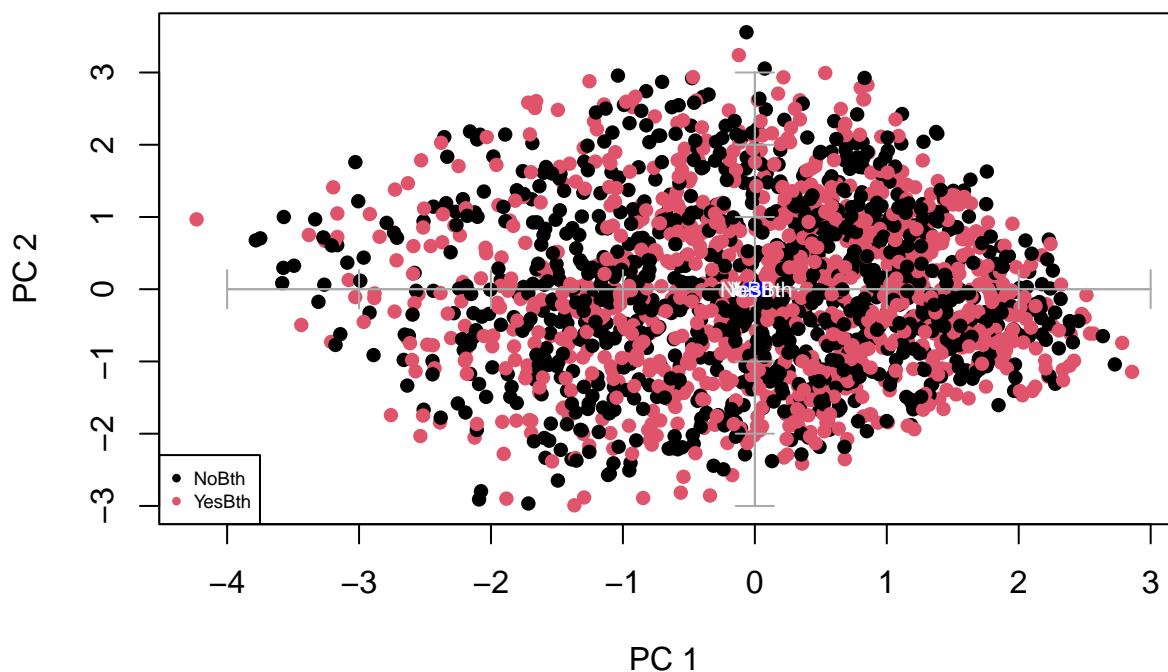
```

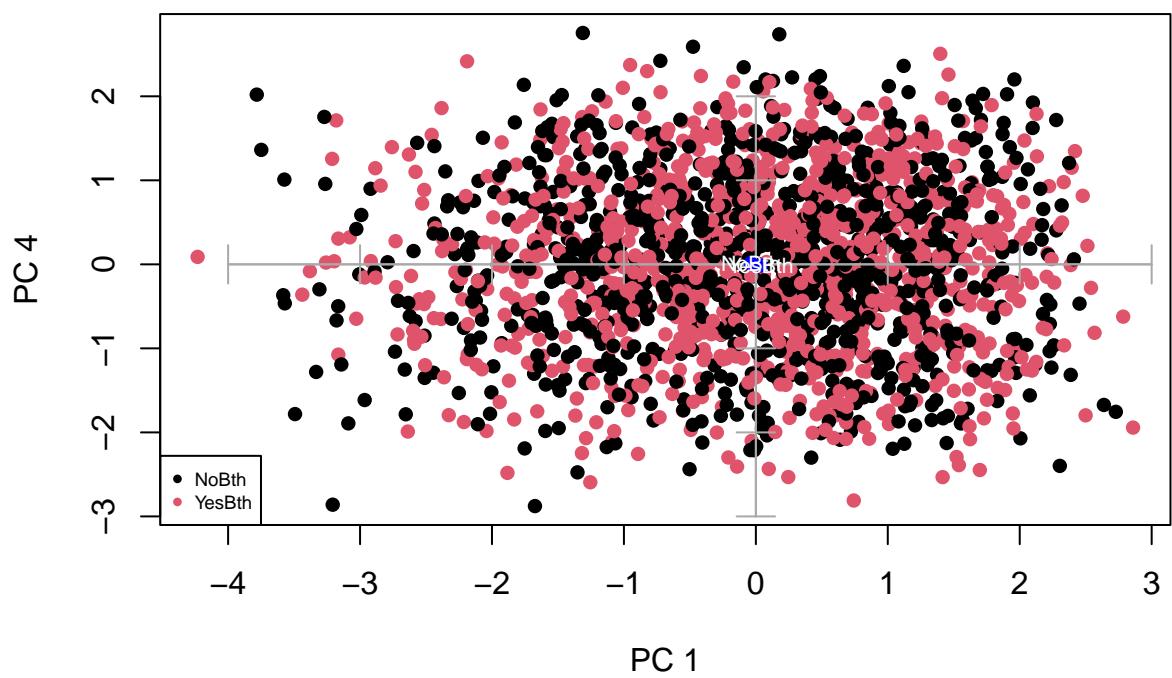
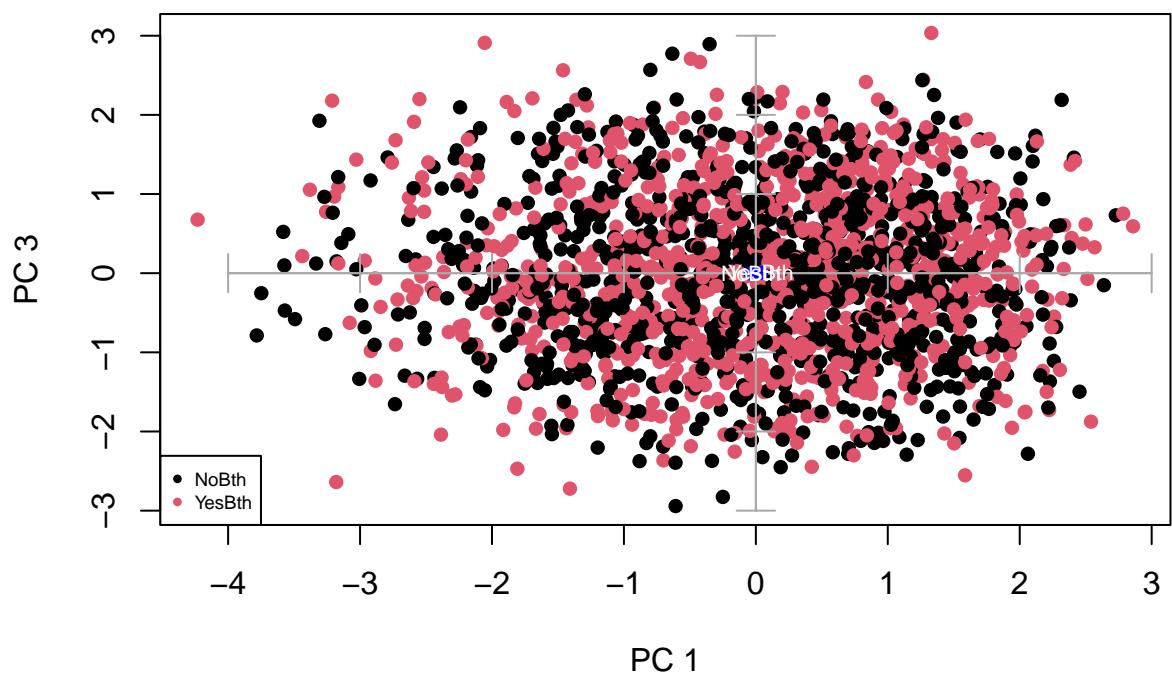
legend("bottomleft",levels(varcat),pch=16,col=c(1:2), cex=0.6)

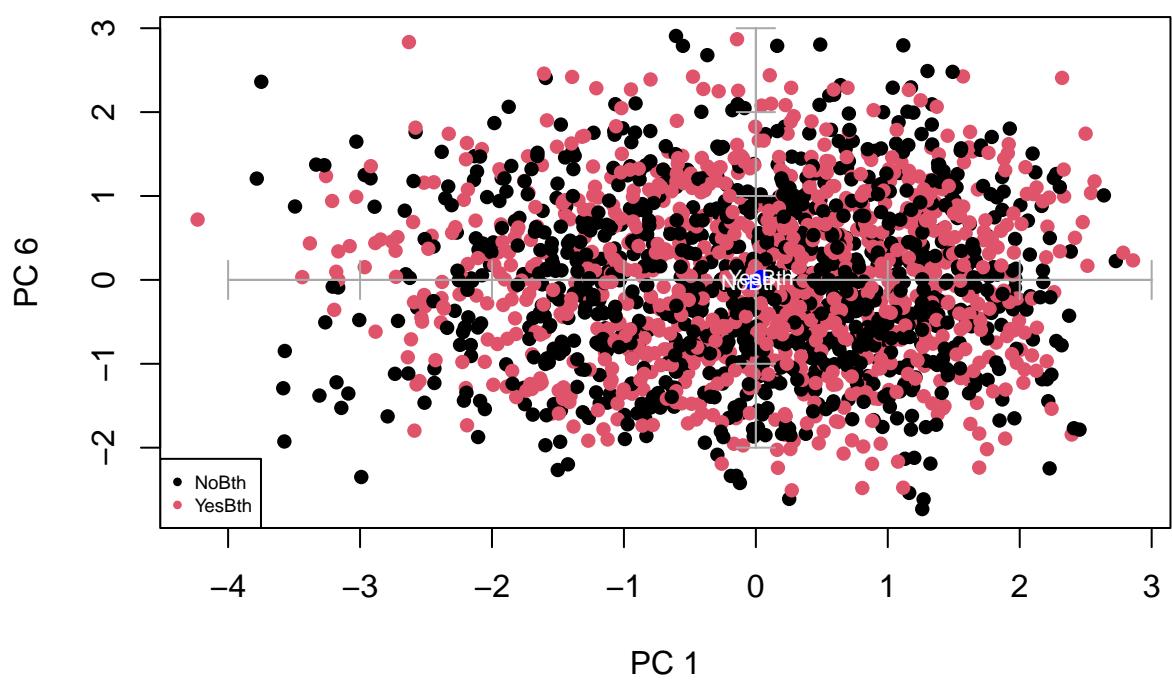
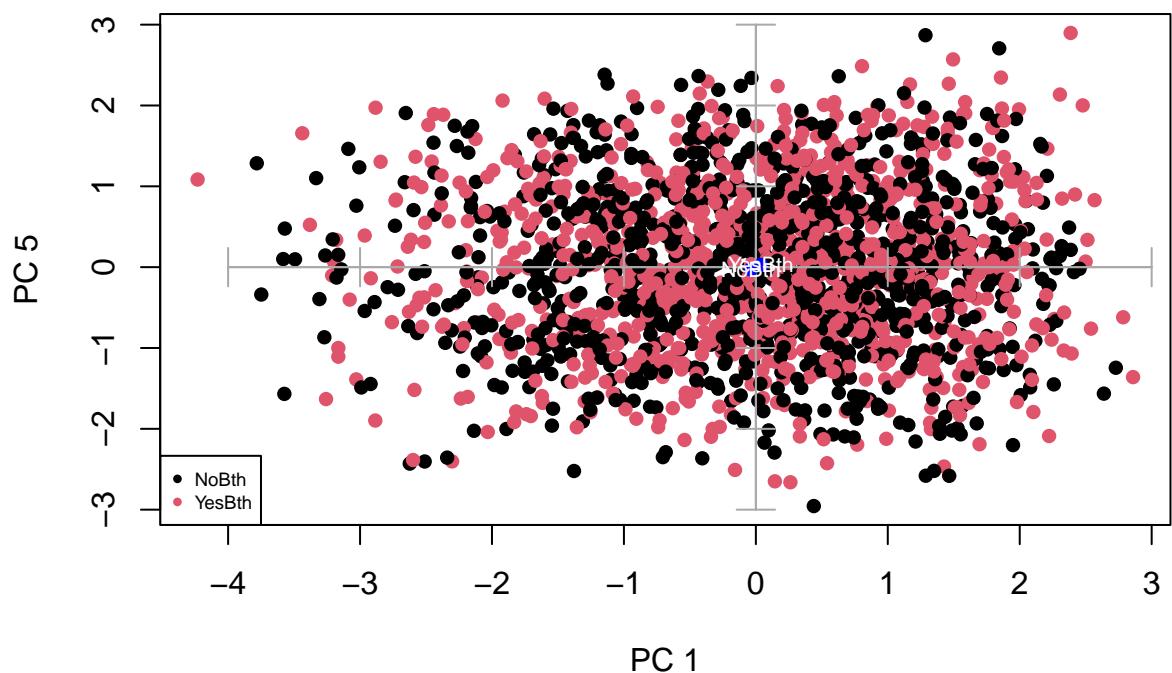
#select your qualitative variable
fdic1 = tapply(Psi[,i],varcat,mean)
fdic2 = tapply(Psi[,j],varcat,mean)
points(fdic1,fdic2,pch=16,col="blue")
text(fdic1,fdic2,labels=levels(varcat),col="white", cex=0.7)
}

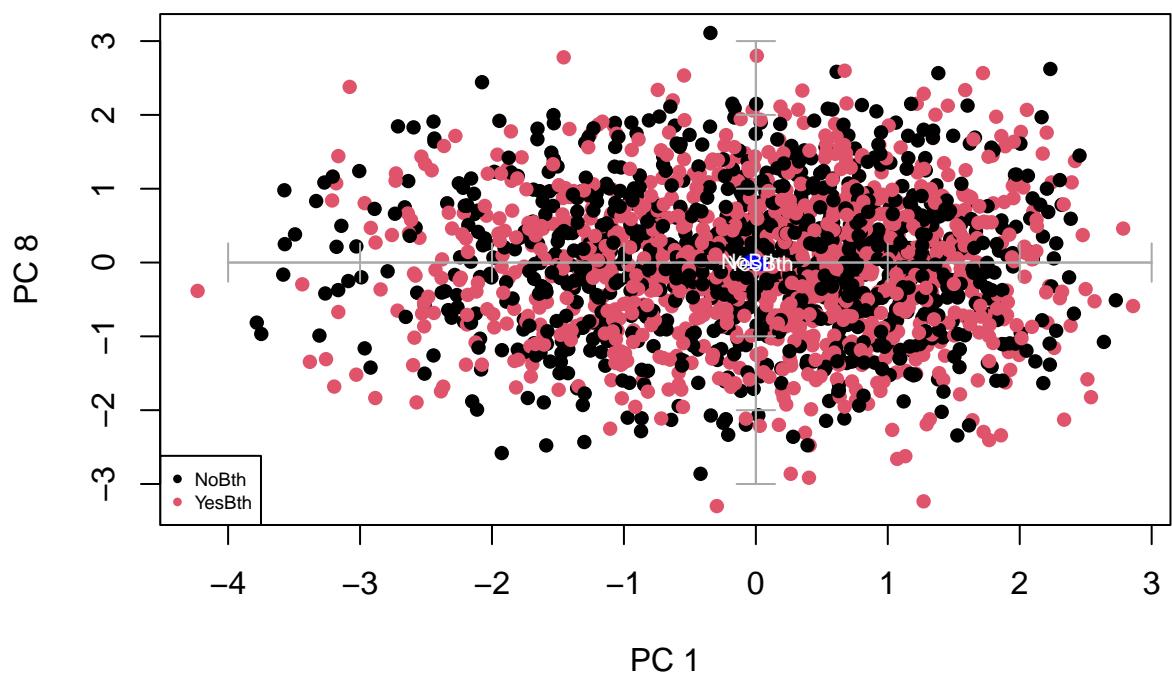
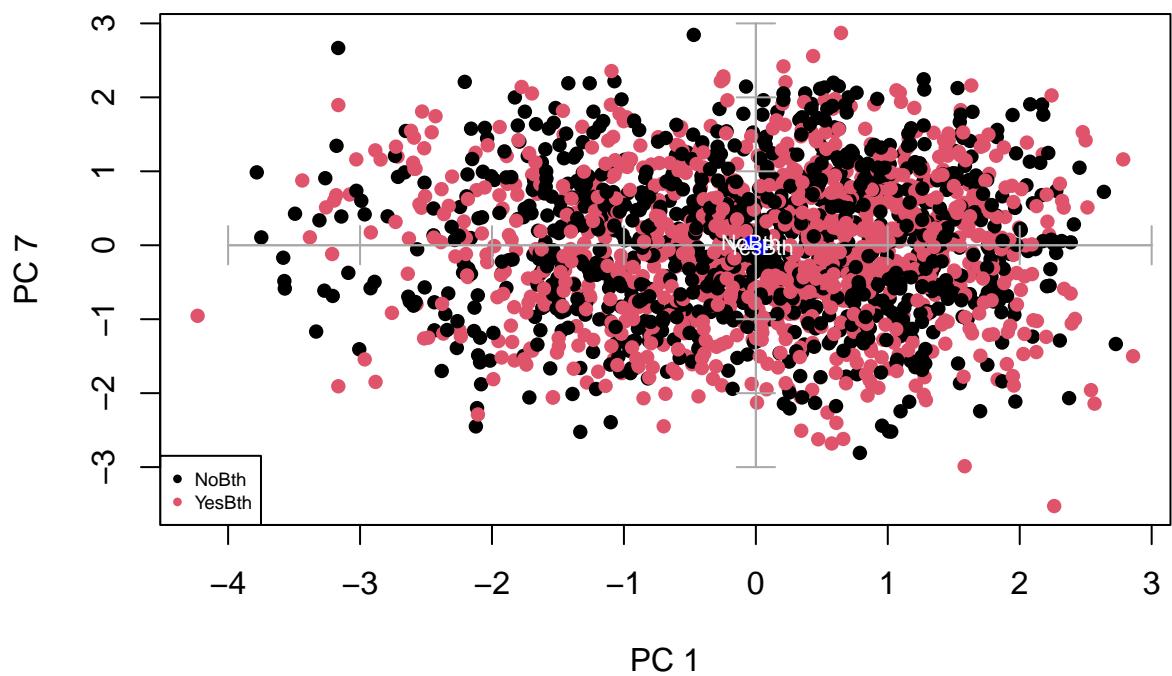
}

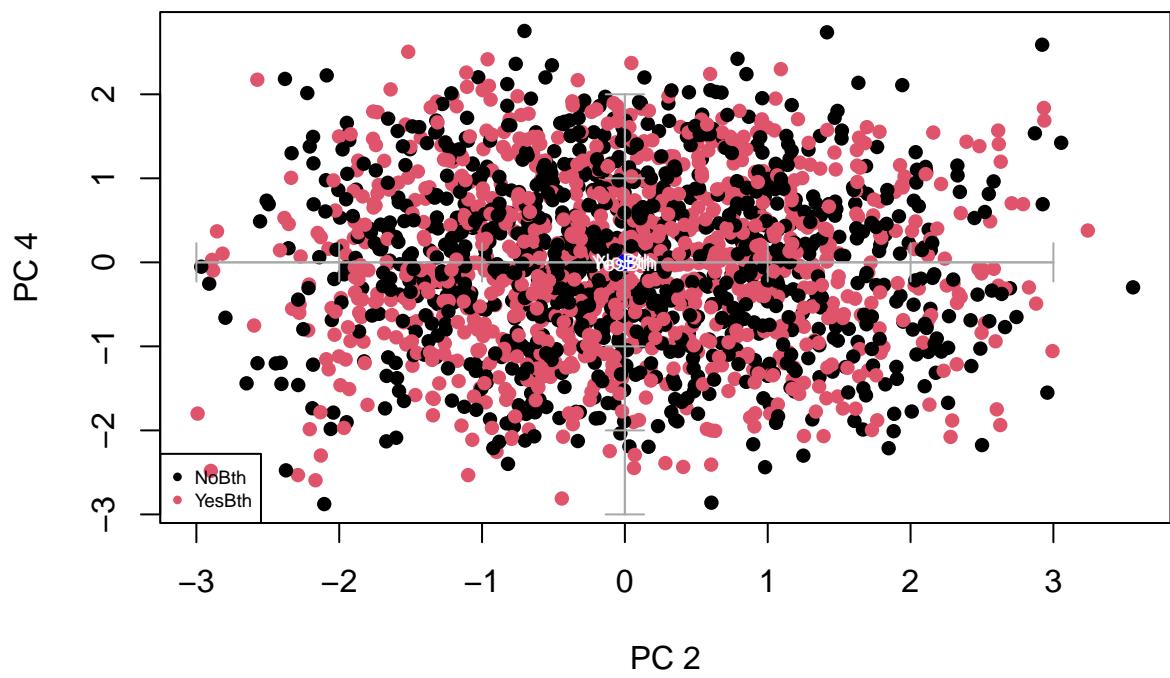
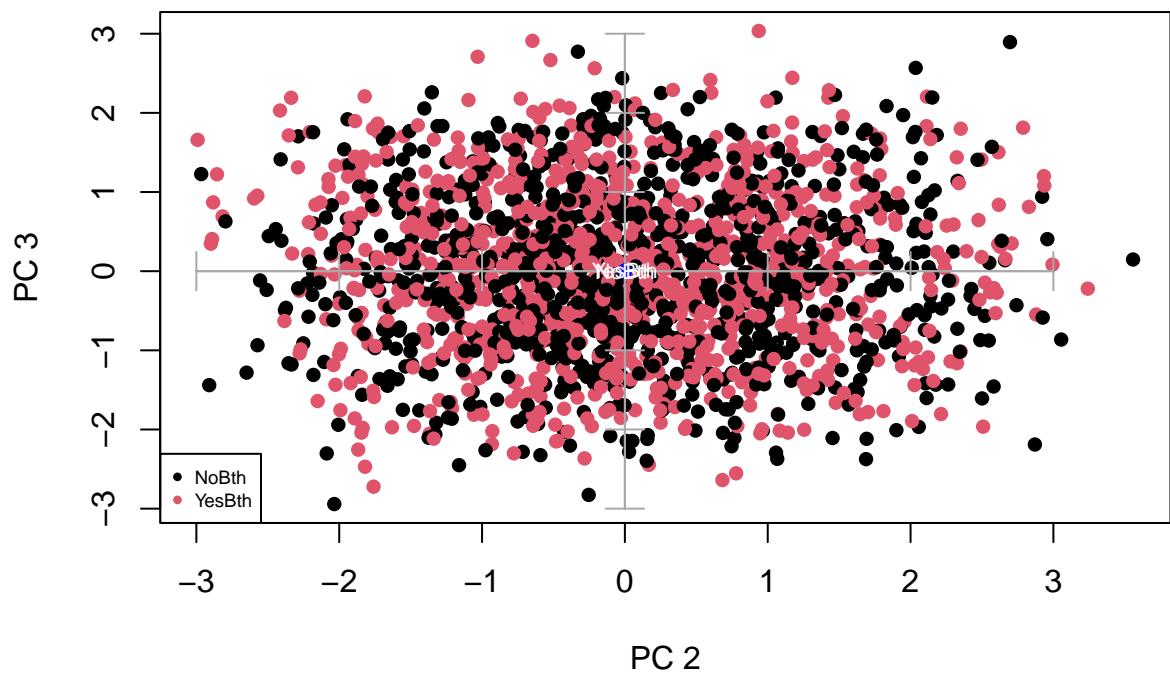
```

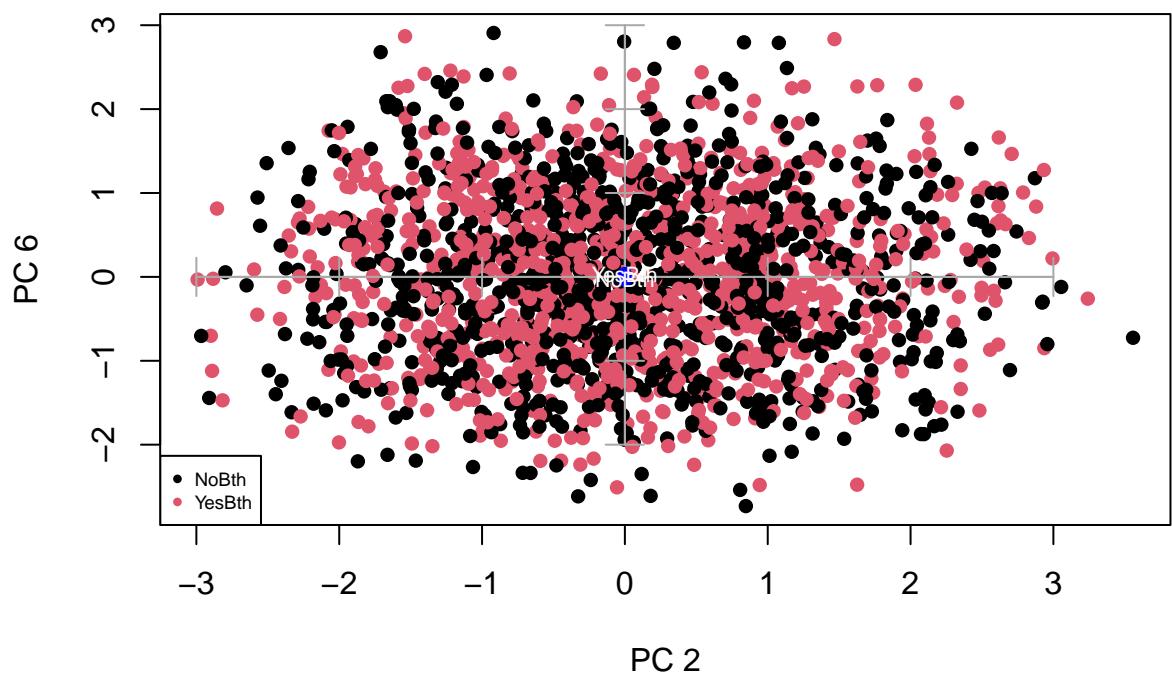
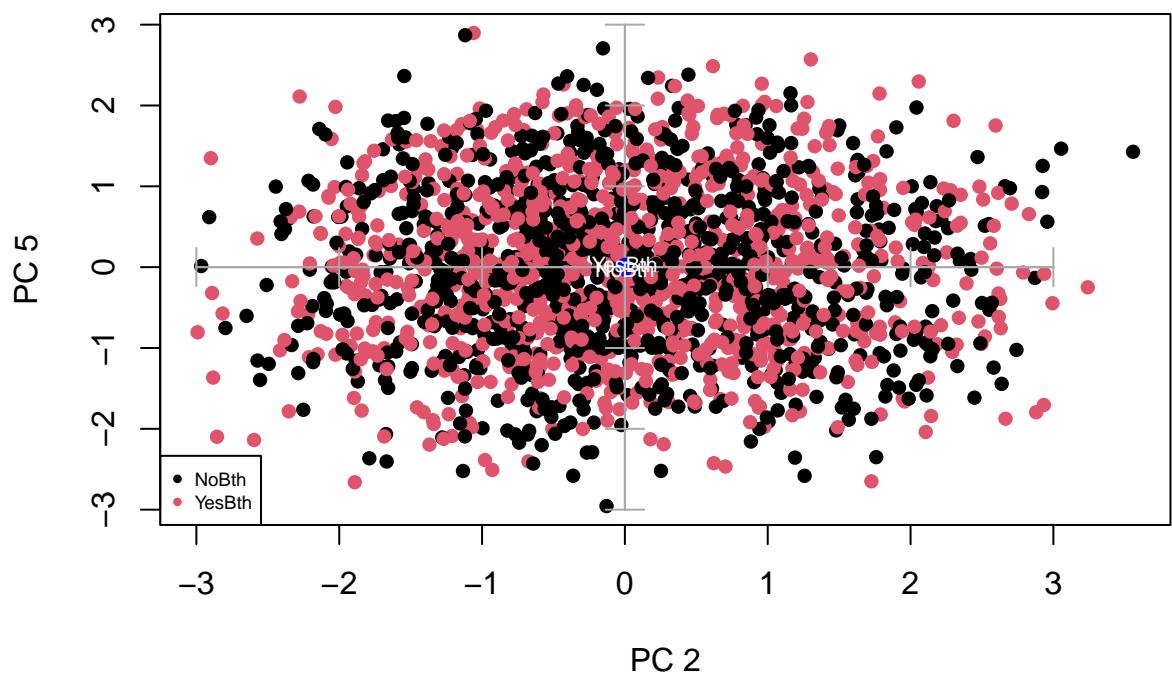


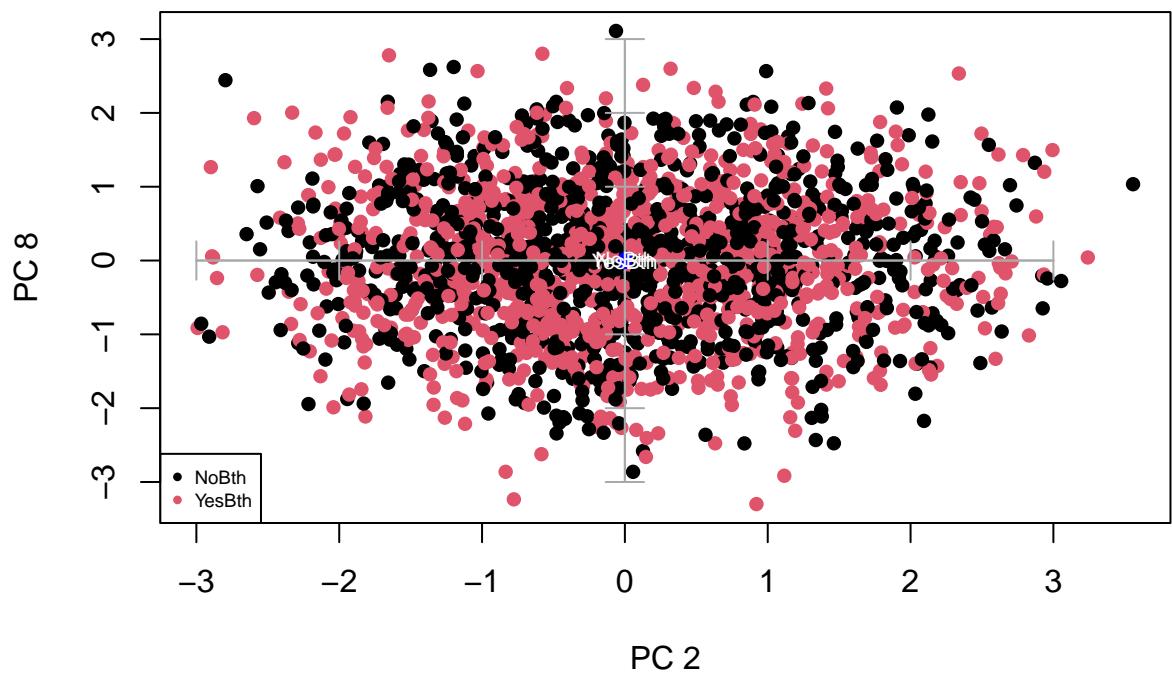
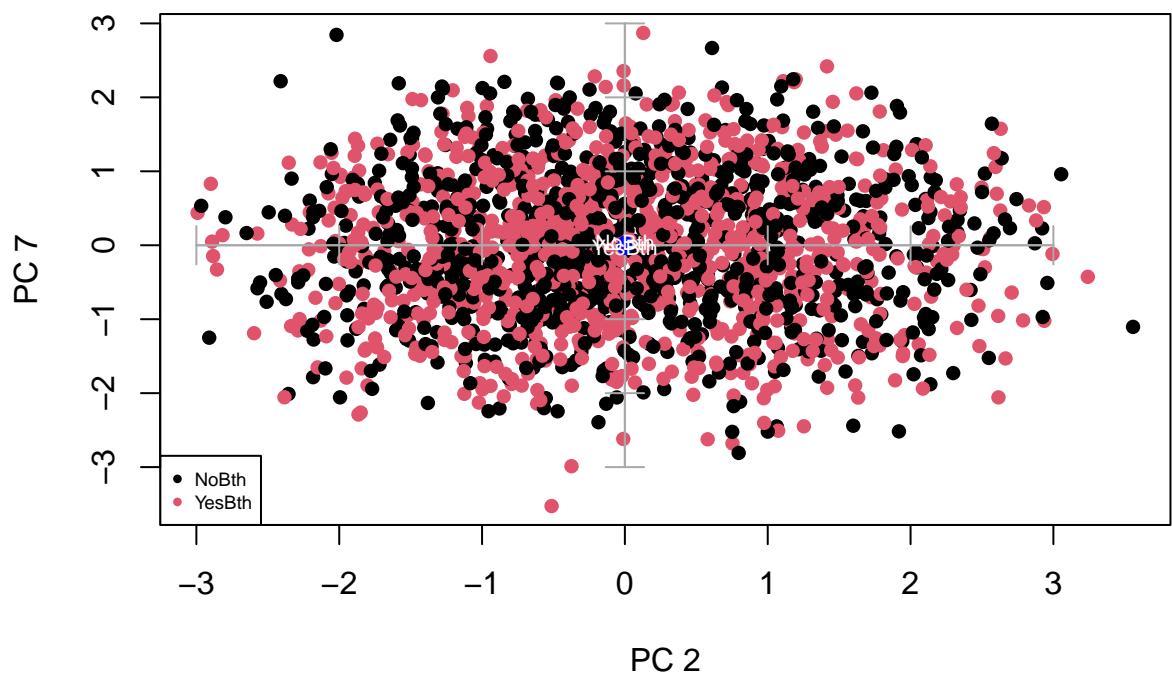


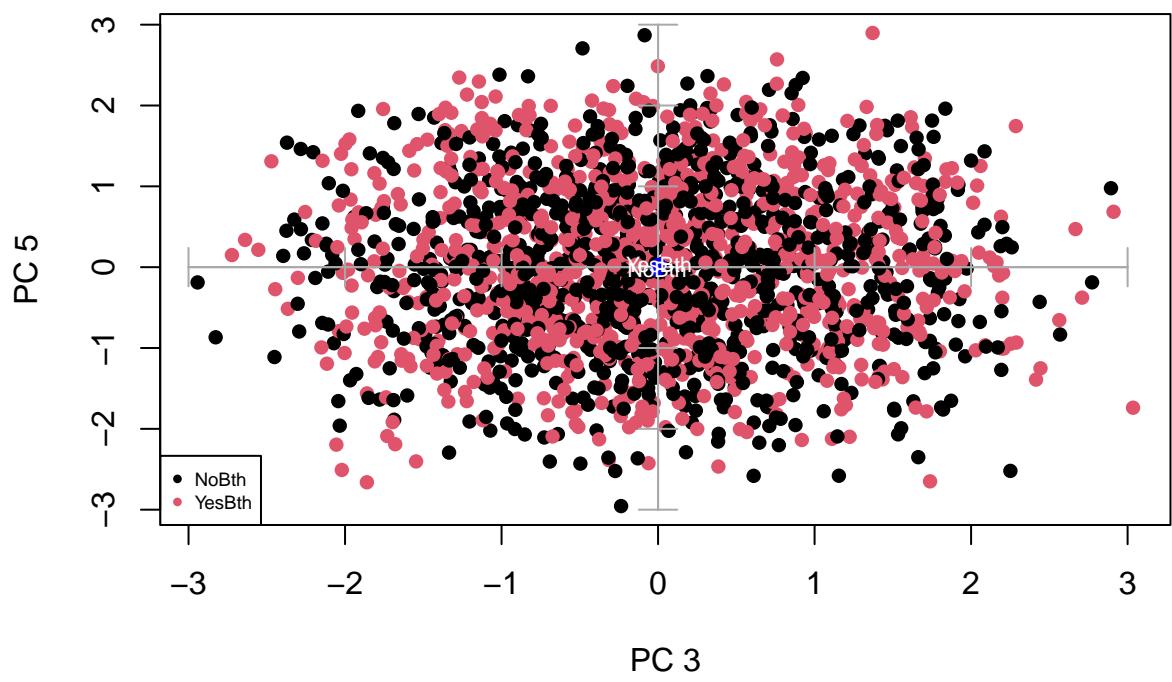
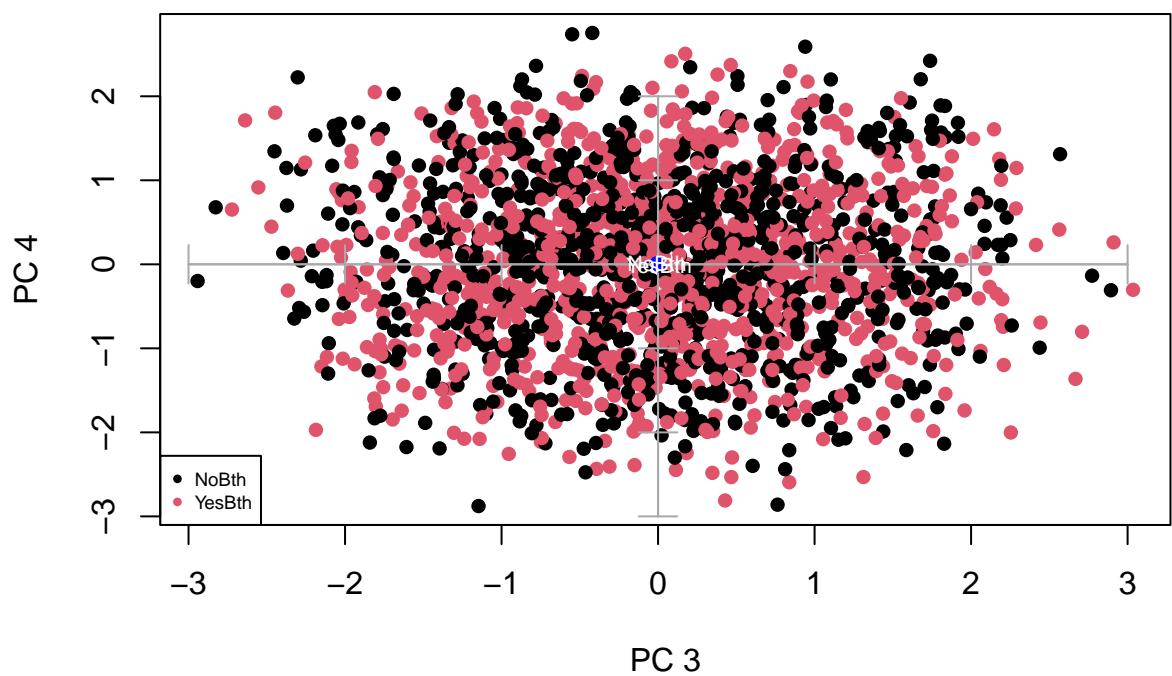


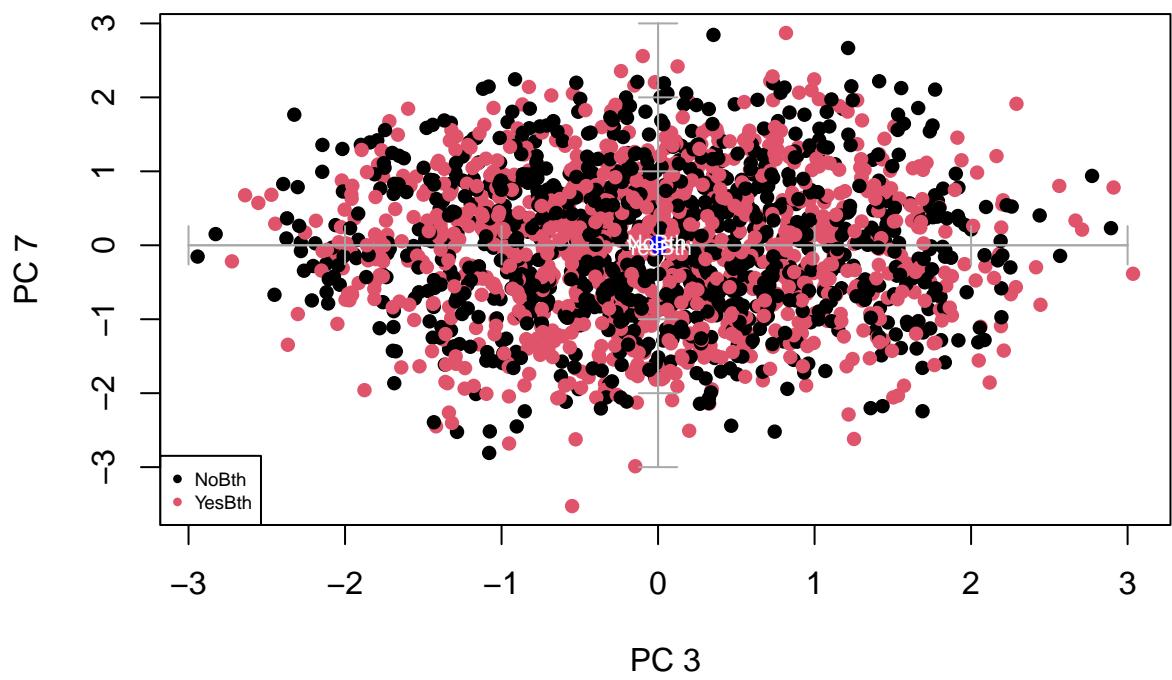
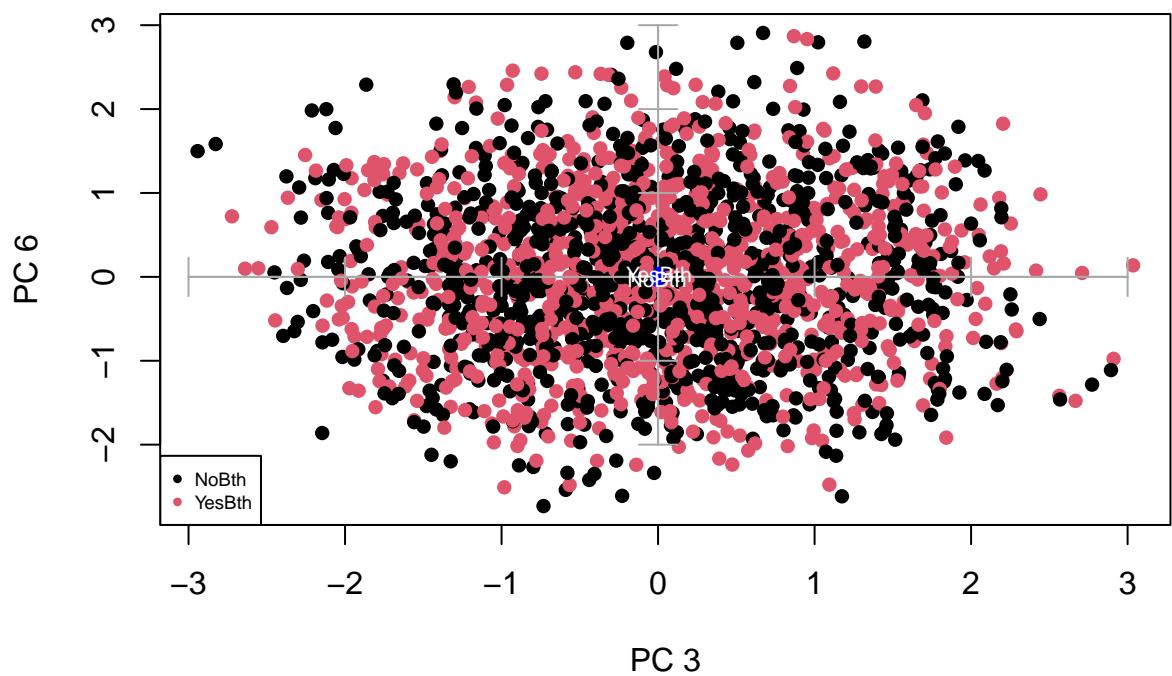


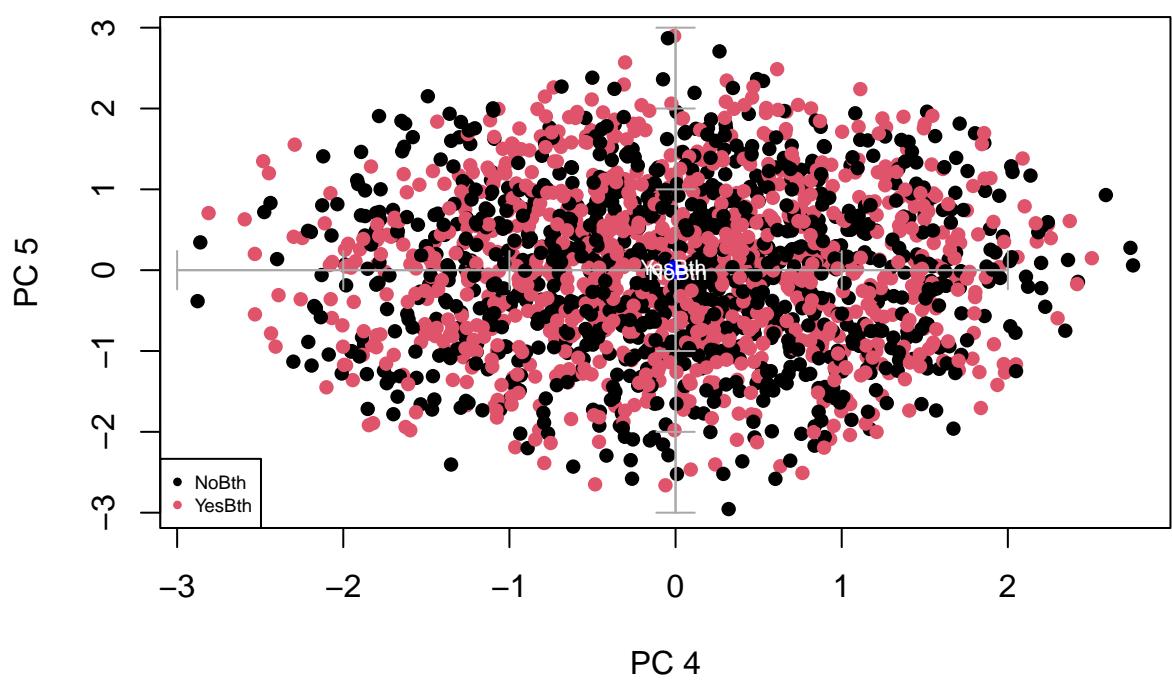
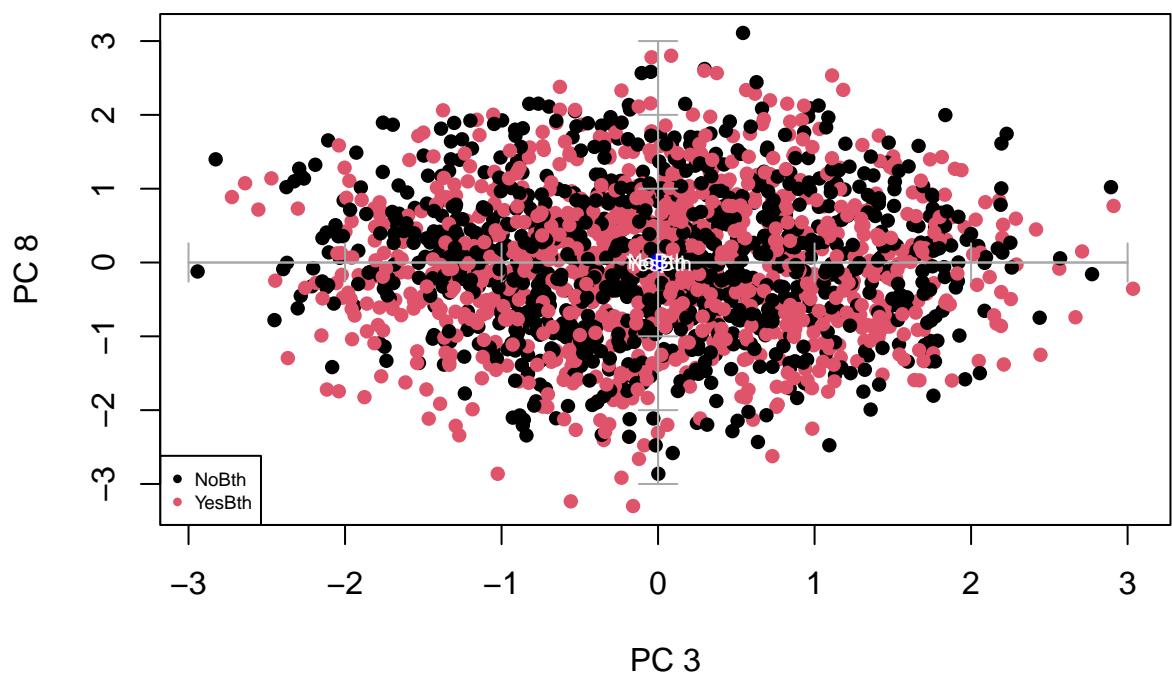


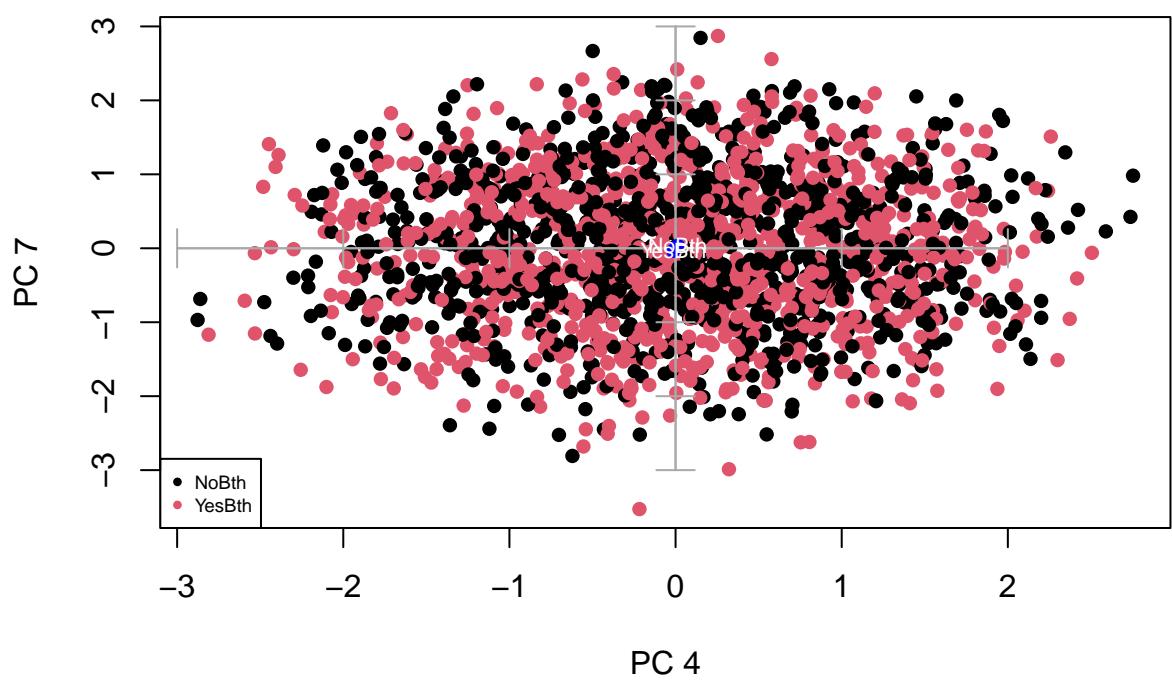
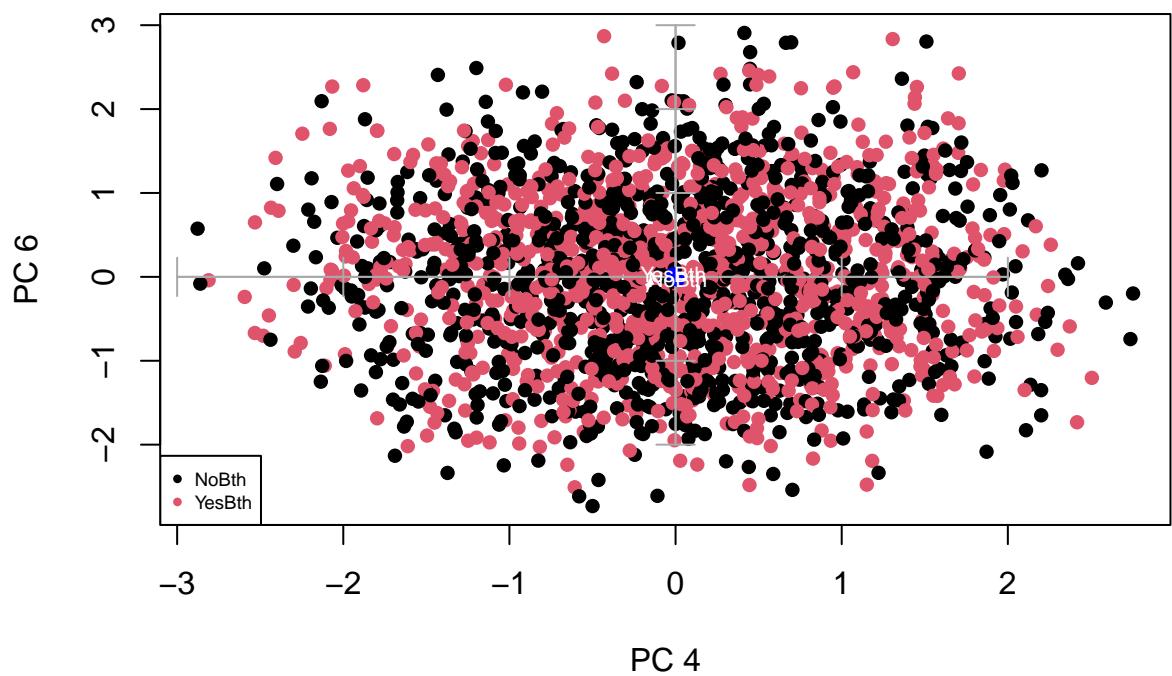


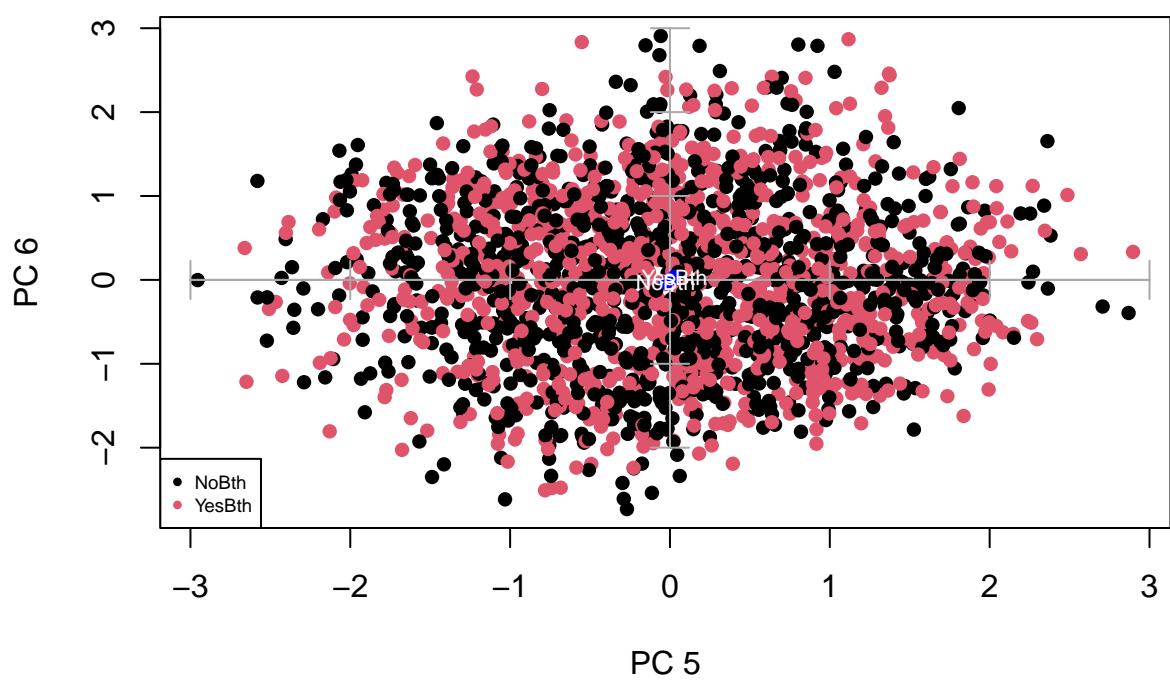
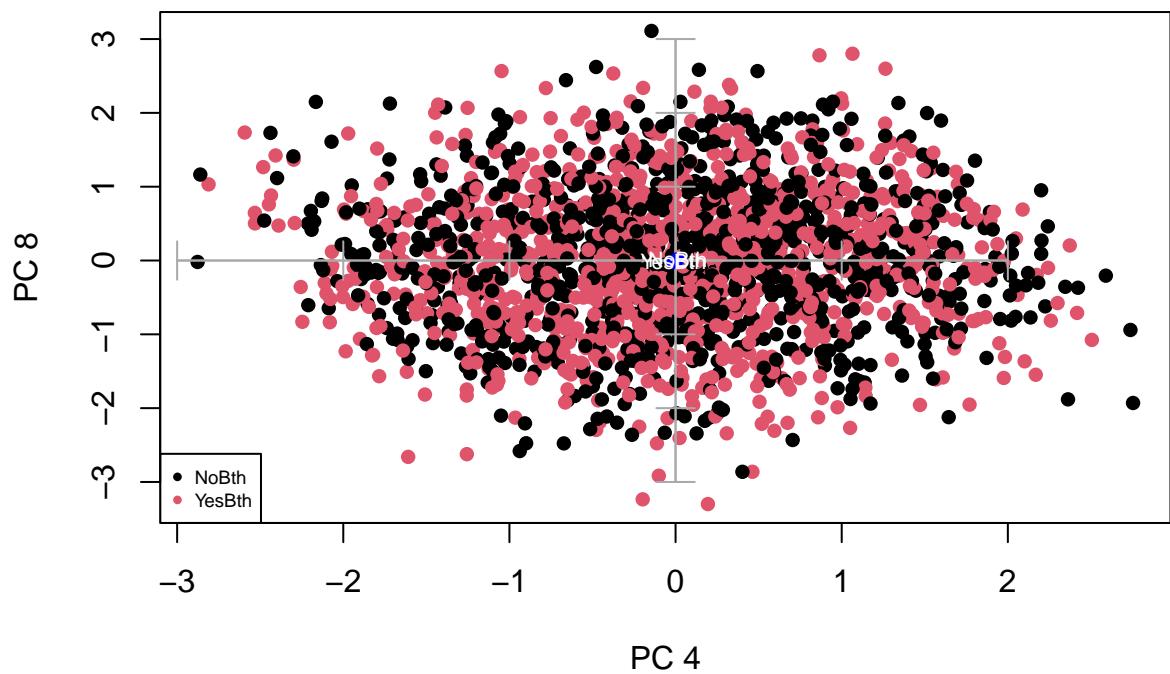


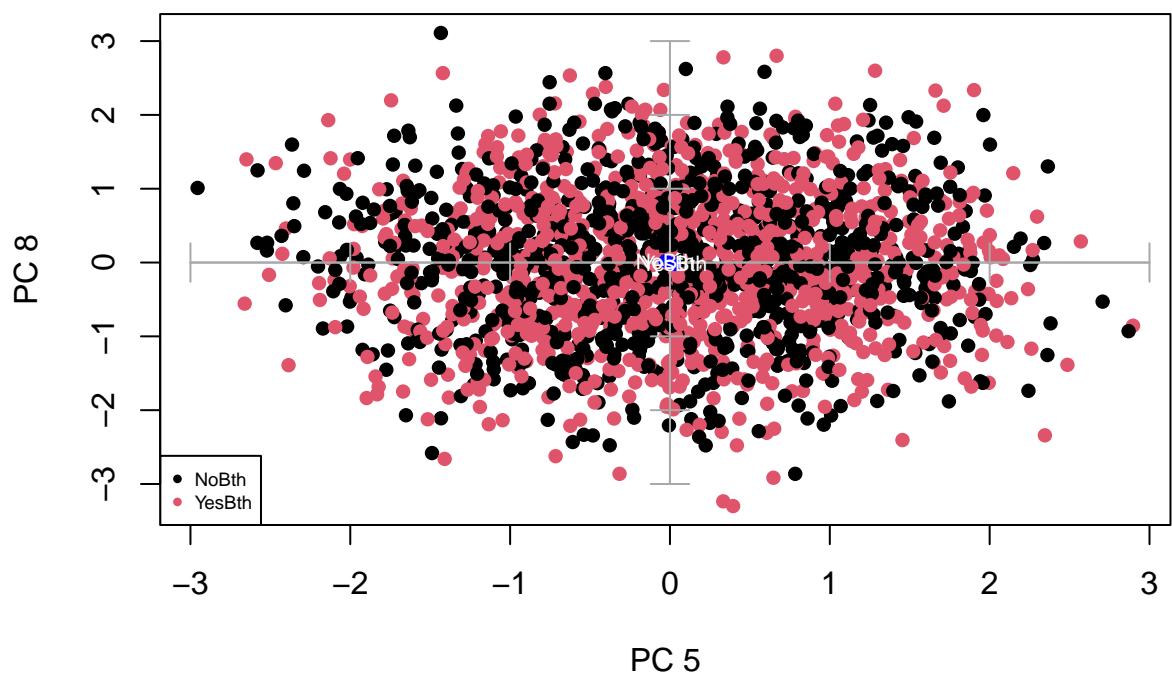
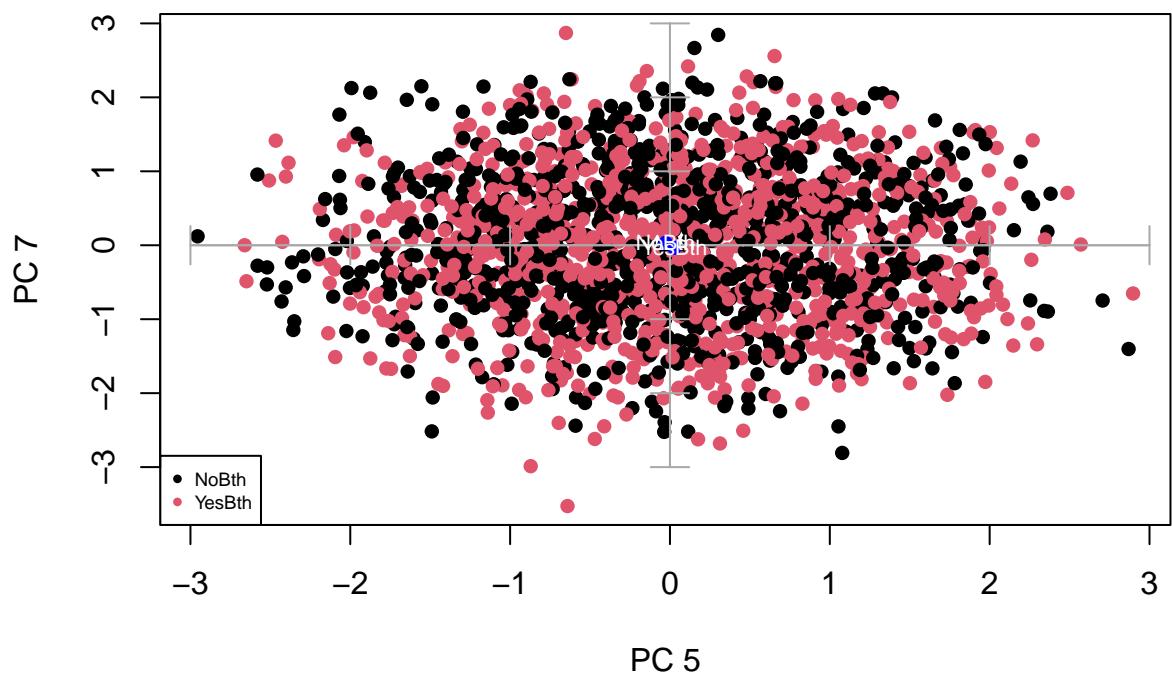


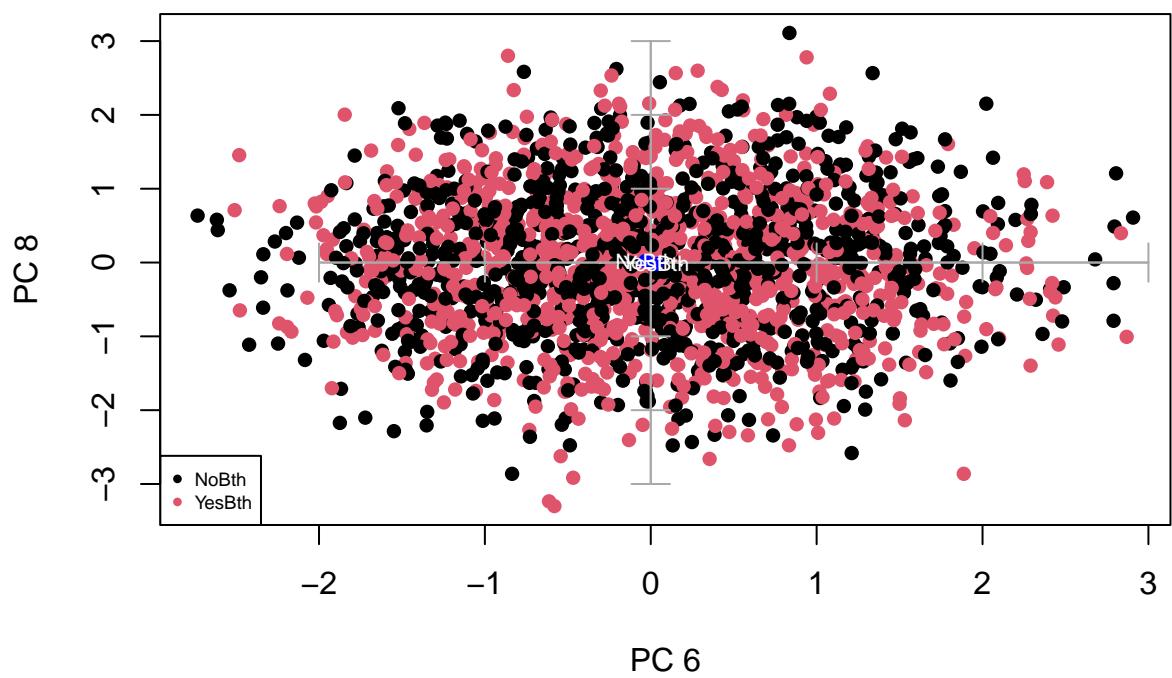
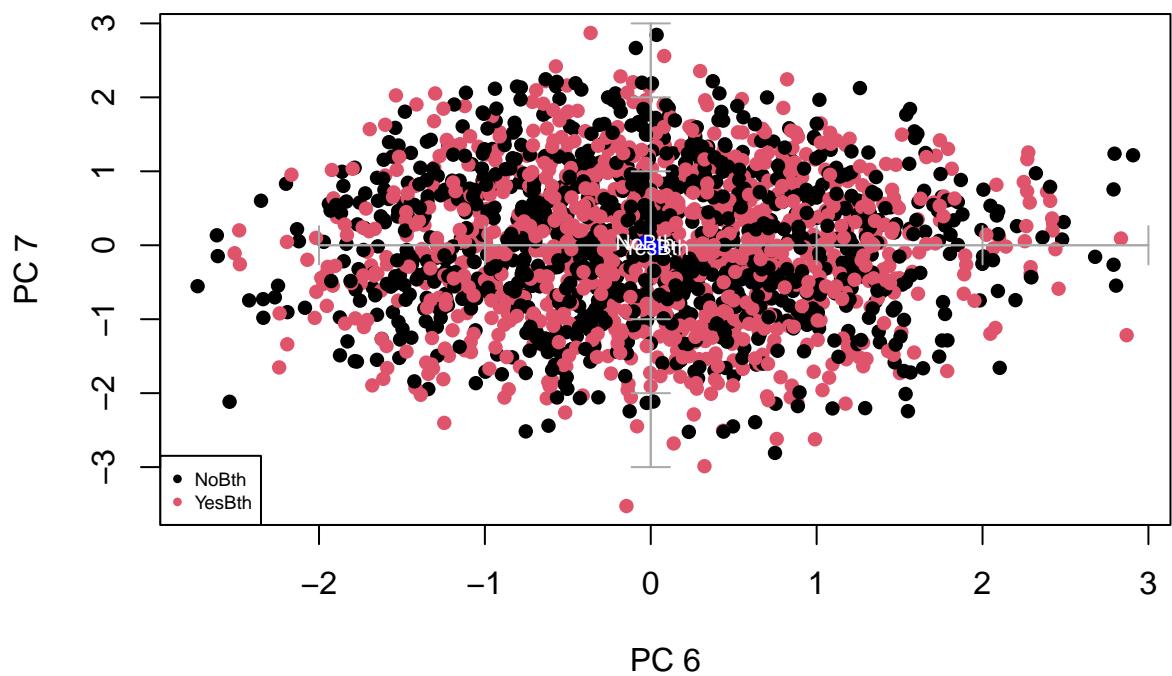


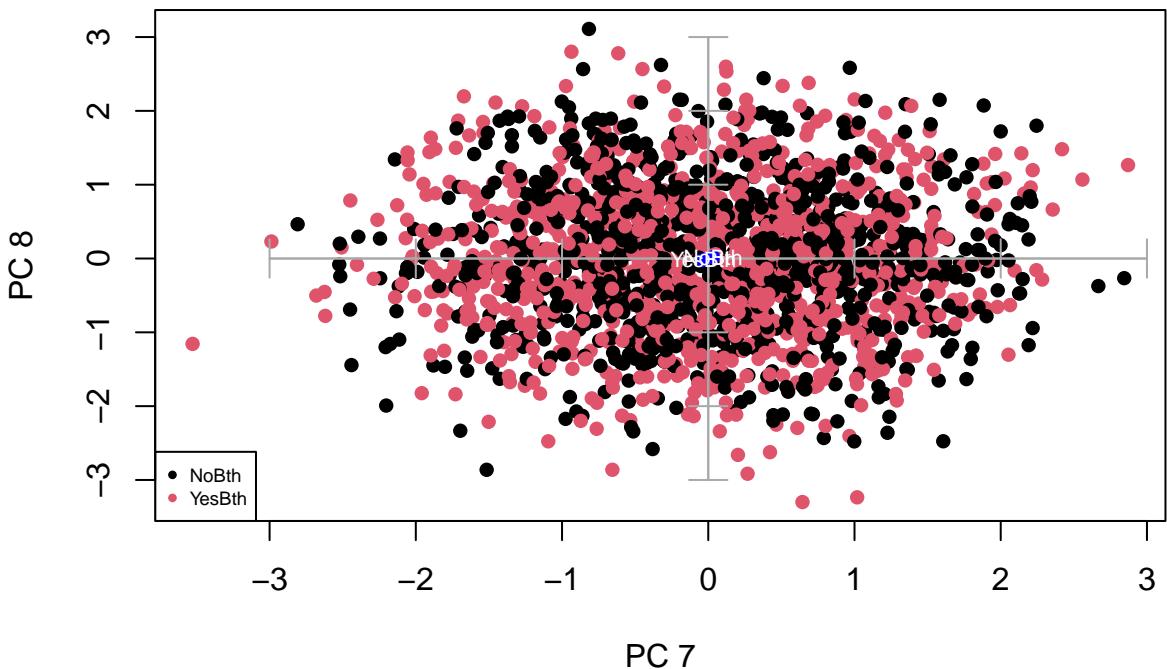












10.5 Profiling plots

```

for(k in 1:K){
  print(paste("Variable", names(dades)[k]))
  if (is.numeric(dades[,k])){
    barplot(tapply(dades[,k], P, mean),main=paste("Means of", names(dades)[k], "by", nameP))
    abline(h=mean(dades[[k]]))
    legend(0,mean(dades[[k]]),"global mean",bty="n")

    pvalk[,1] <- ValorTestXnum(dades[,1], P)
    #p-values ValorsTest:
    pvalk[,1]

  } else{
    table<-table(P,dades[,k])
    colperc<-prop.table(table,2)
    colperc

    #diagrammes de barres adosades
    barplot(table(dades[,k], as.factor(P)), beside=TRUE, col=paleta )
    legend("topright",levels(as.factor(dades[,k])),pch=1,cex=0.5, col=paleta)

    #with legend
    plot(marg,type="l",ylim=c(0,1),main=paste("Prop. of pos & neg by",names(dades)[k]))
    paleta<-rainbow(length(levels(dades[,k])))
    for(c in 1:length(levels(dades[,k]))) {lines(colperc[,c],col=paleta[c])}
    legend("topright", levels(dades[,k]), col=paleta, lty=2, cex=0.6)

    #amb variable en eix d'abscisses
    marg <-table(dades[,k])/n
    print(append("Categories=",levels(dades[,k])))
    plot(marg,type="l",ylim=c(0,1),main=paste("Prop. of pos & neg by",names(dades)[k]), las=3)
  }
}

```

```

paleta<-rainbow(length(levels(as.factor(P))))
for(c in 1:length(levels(as.factor(P)))){lines(rowperc[c,],col=paleta[c]) }
legend("topright", levels(as.factor(P)), col=paleta, lty=2, cex=0.6)

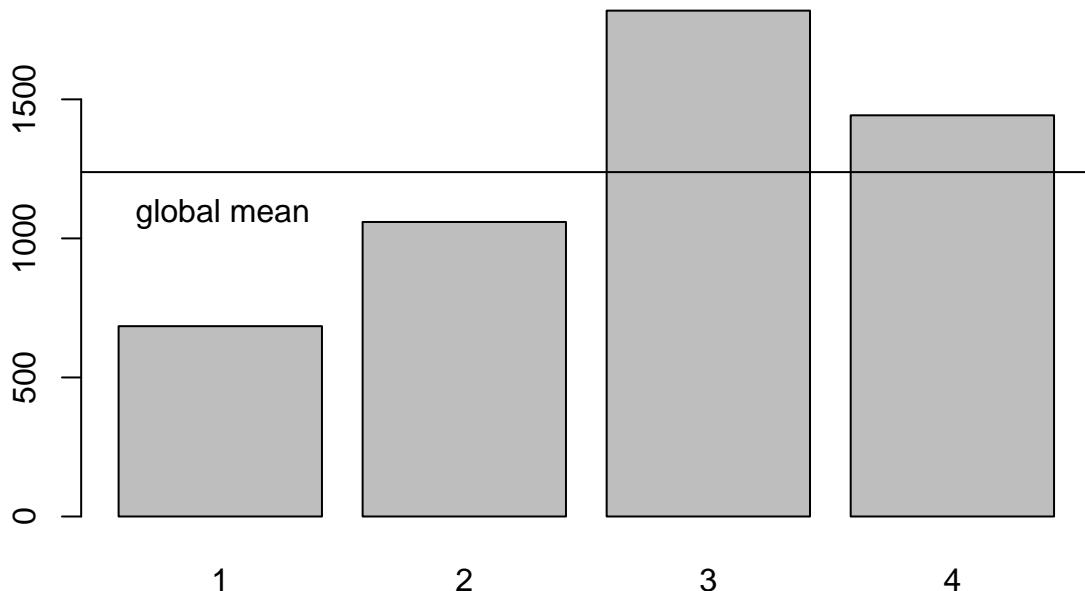
#condicionadas a columna
plot(marg,type="n",ylim=c(0,1),main=paste("Prop. of pos & neg by",names(dades)[k]), las=3)
paleta<-rainbow(length(levels(as.factor(P))))
for(c in 1:length(levels(as.factor(P)))){lines(colperc[c,],col=paleta[c]) }
legend("topright", levels(as.factor(P)), col=paleta, lty=2, cex=0.6)

}

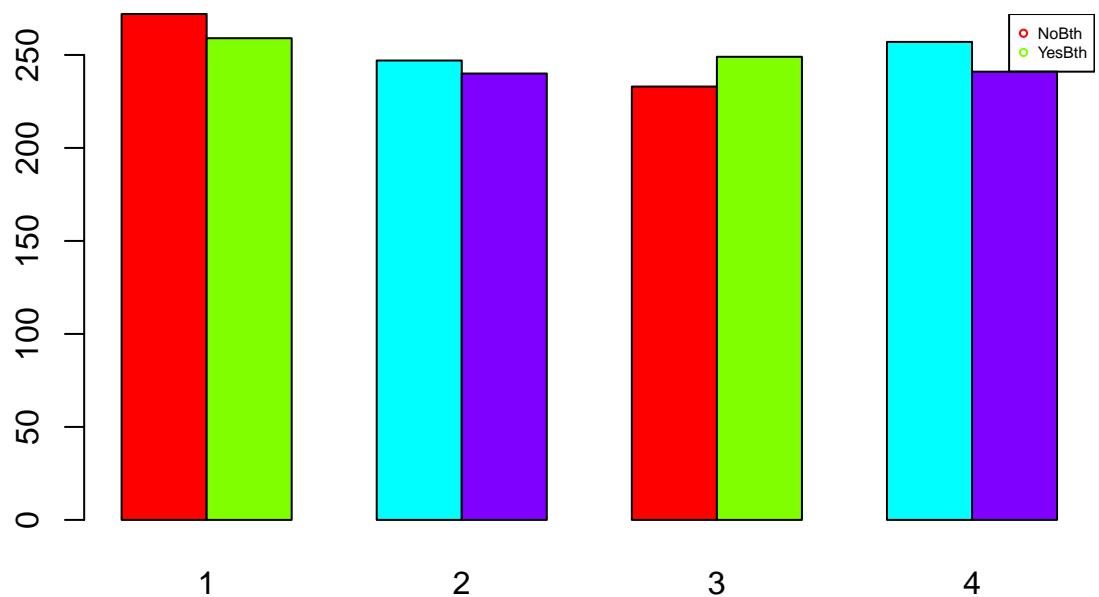
## [1] "Variable battery_power"

```

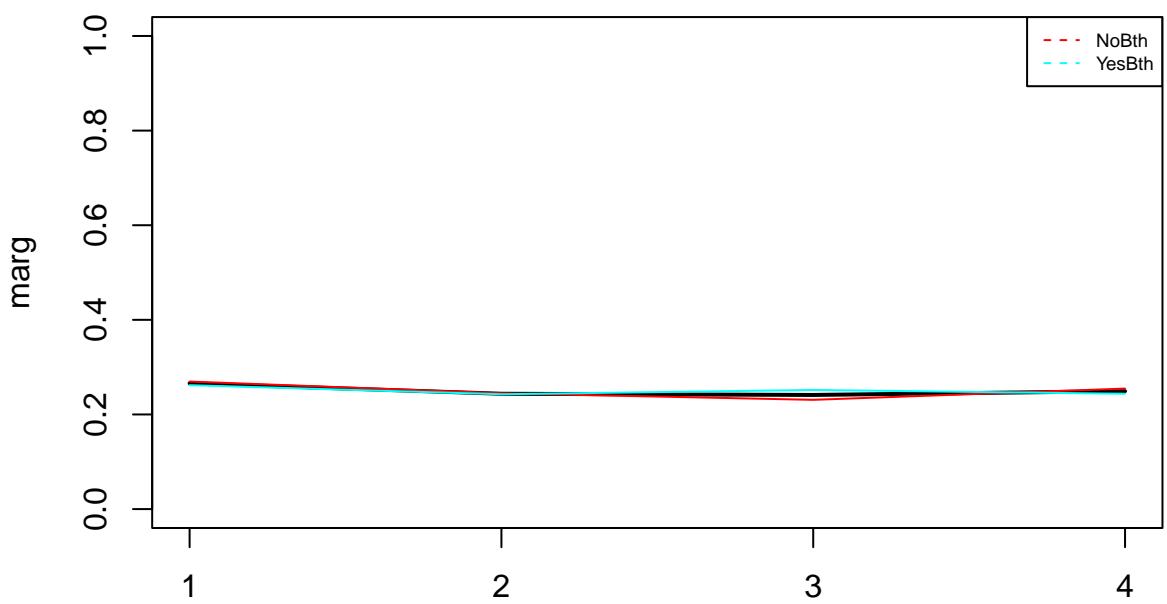
Means of battery_power by Clustering classes



```
## [1] "Variable blue"
```

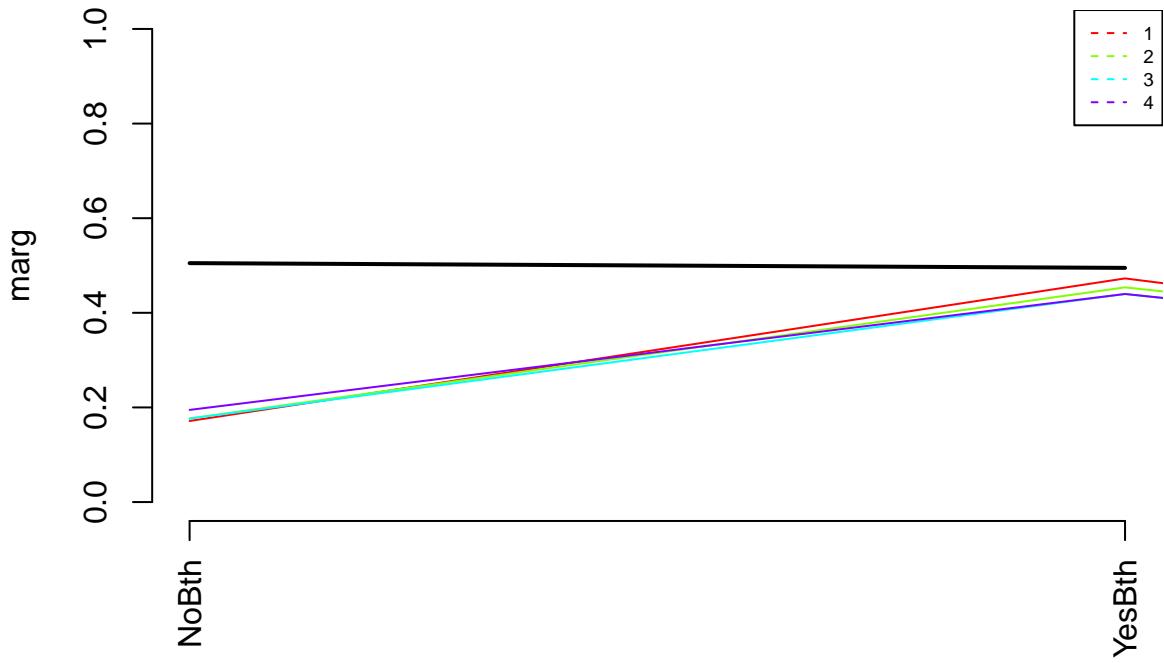


Prop. of pos & neg by blue

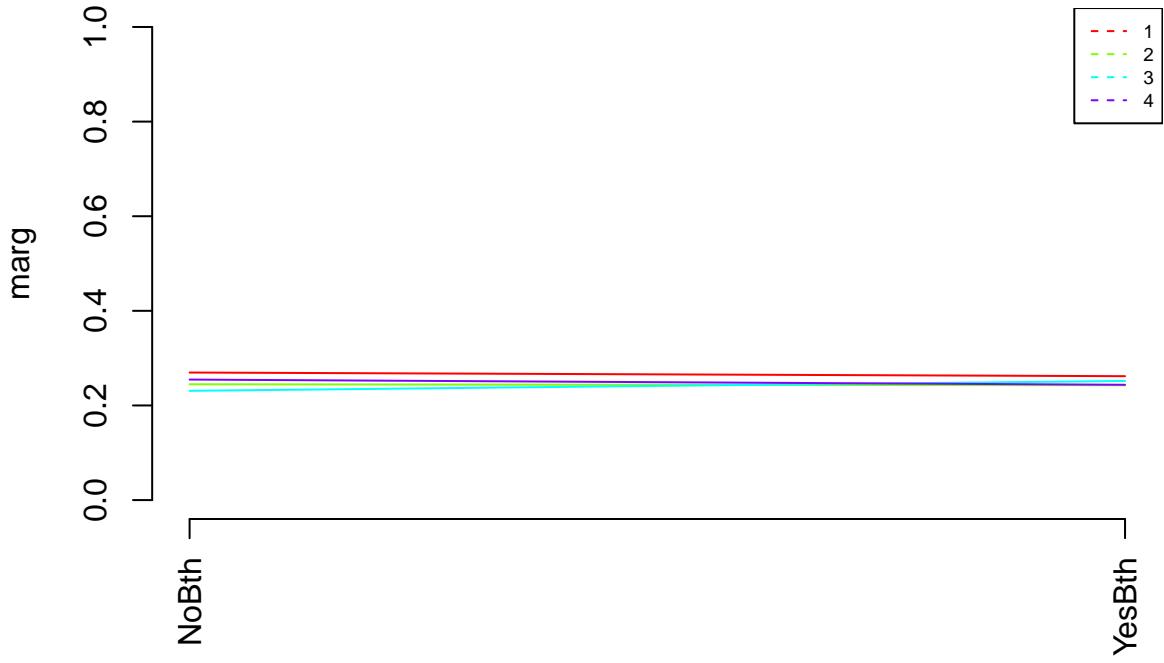


```
## [1] "Categories=" "NoBth"      "YesBth"
```

Prop. of pos & neg by blue

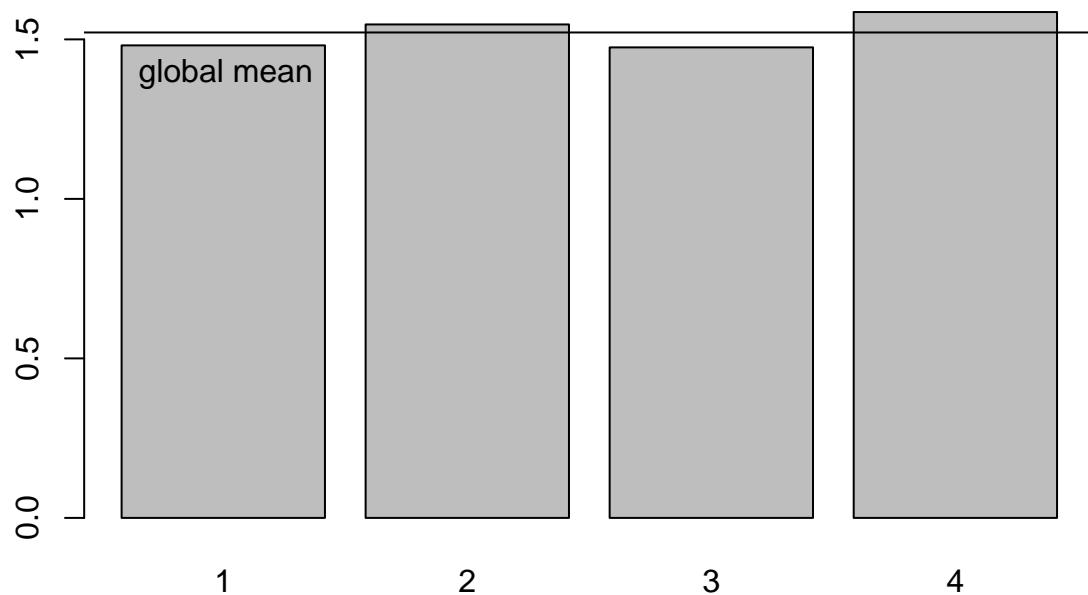


Prop. of pos & neg by blue

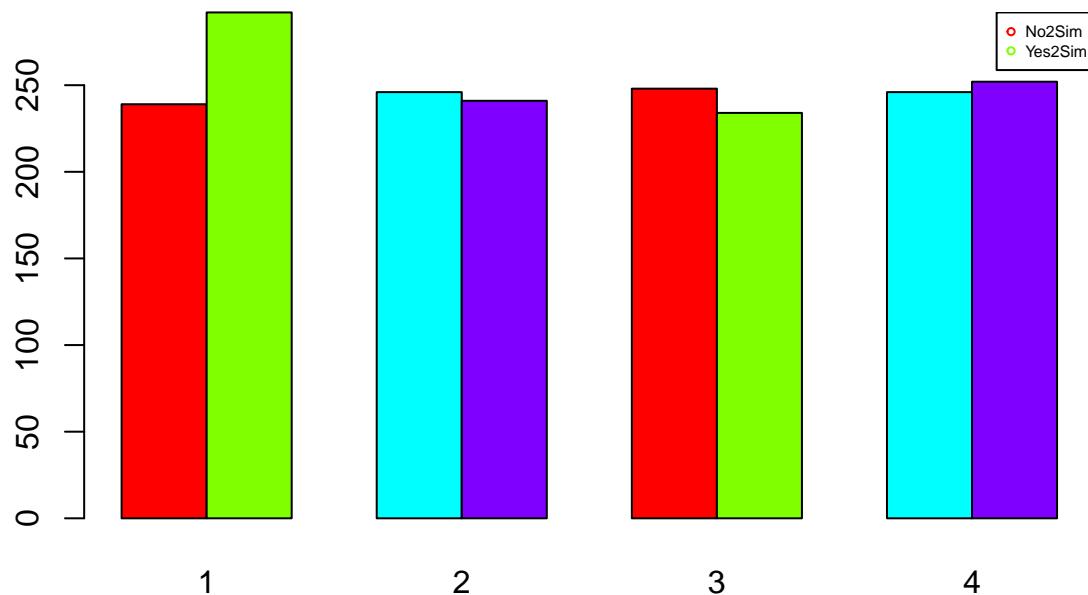


```
## [1] "Variable clock_speed"
```

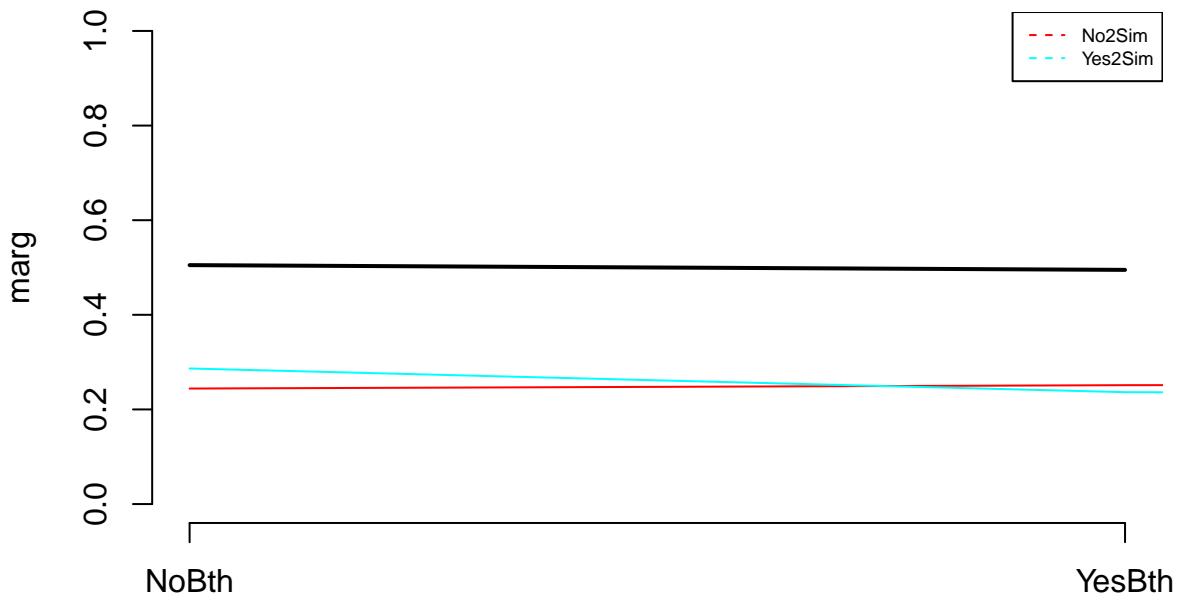
Means of clock_speed by Clustering classes



```
## [1] "Variable dual_sim"
```

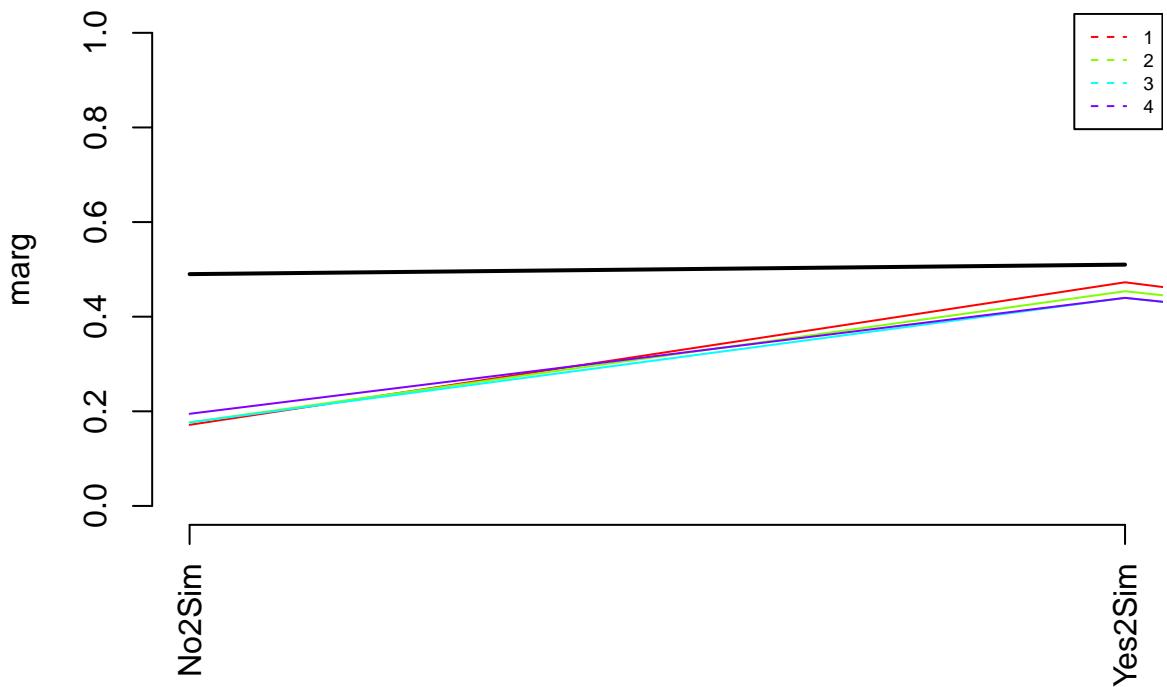


Prop. of pos & neg by dual_sim

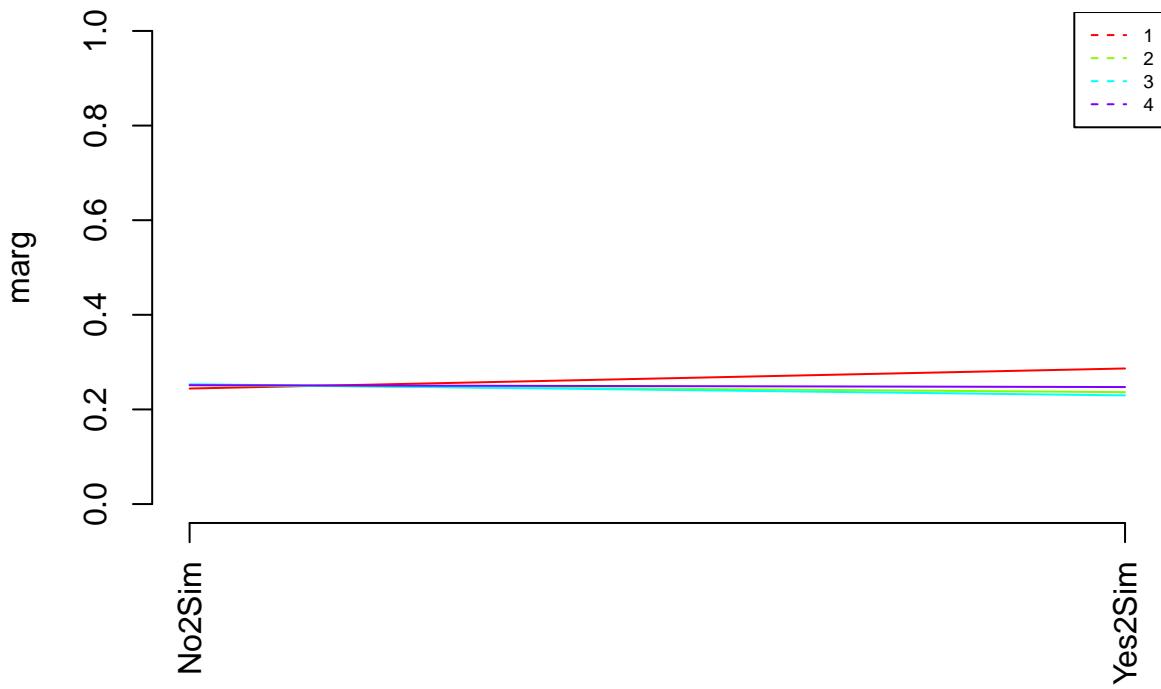


```
## [1] "Categories=" "No2Sim"      "Yes2Sim"
```

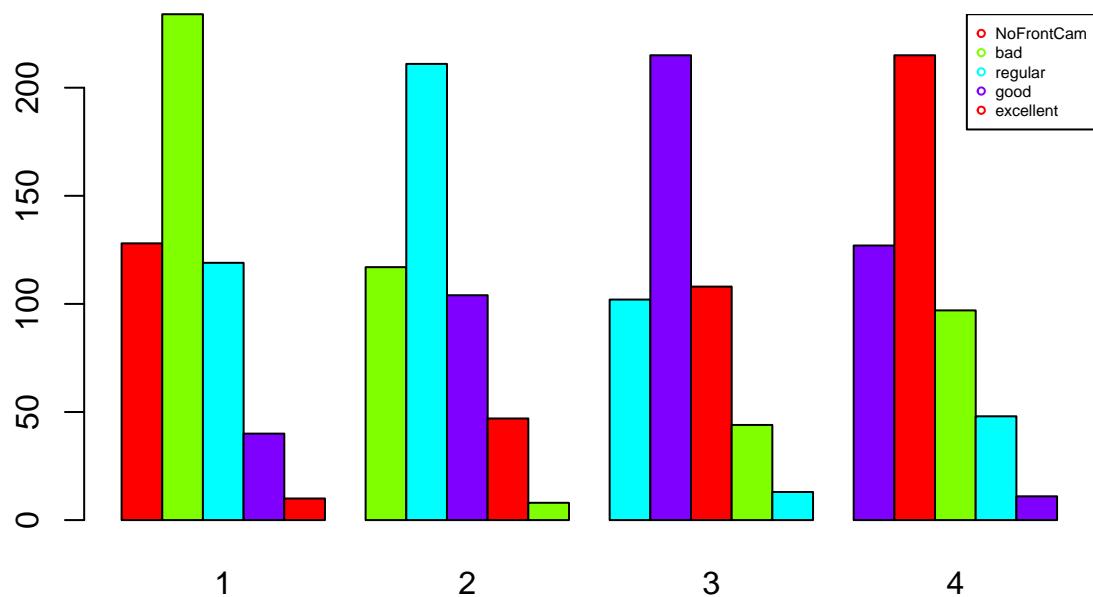
Prop. of pos & neg by dual_sim



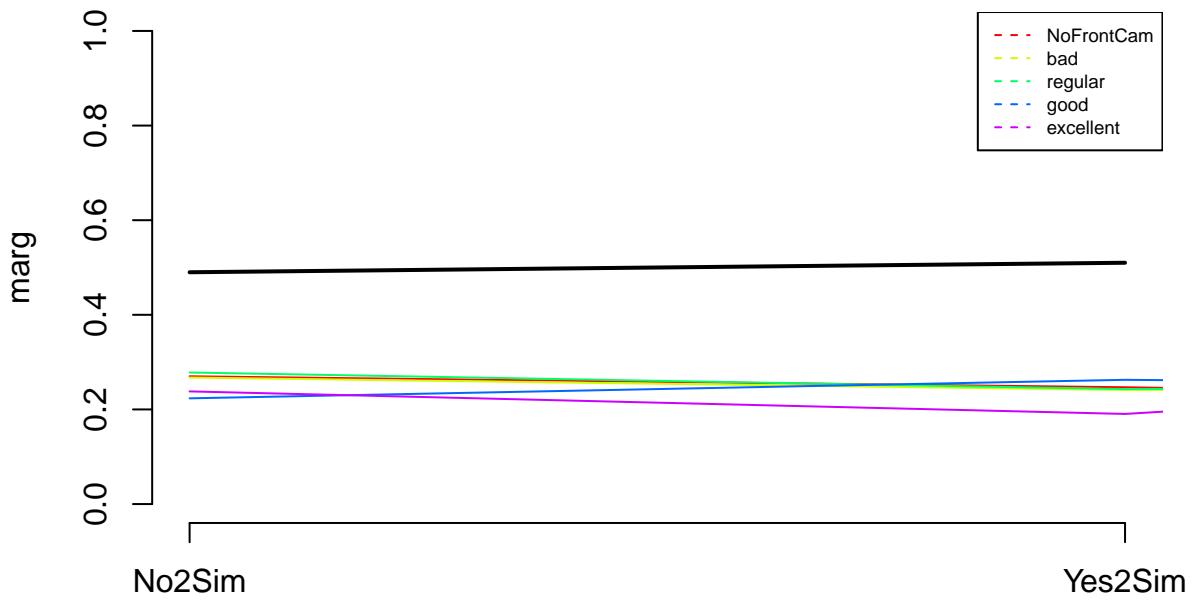
Prop. of pos & neg by dual_sim



```
## [1] "Variable fc"
```

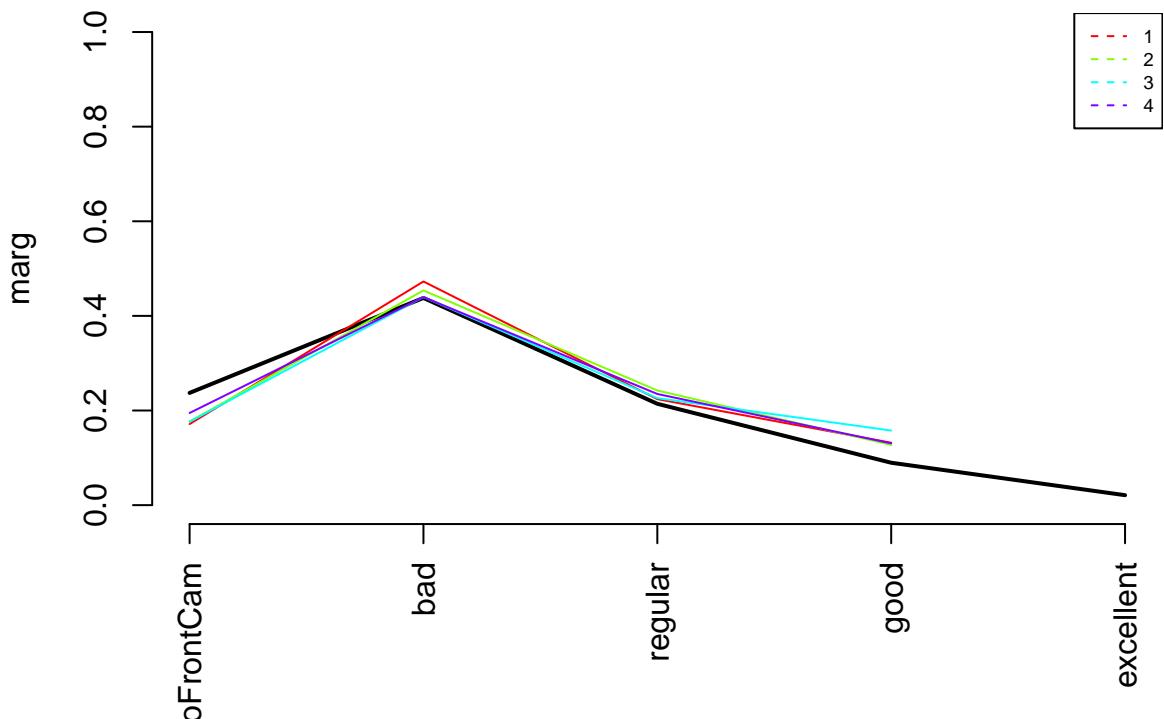


Prop. of pos & neg by fc

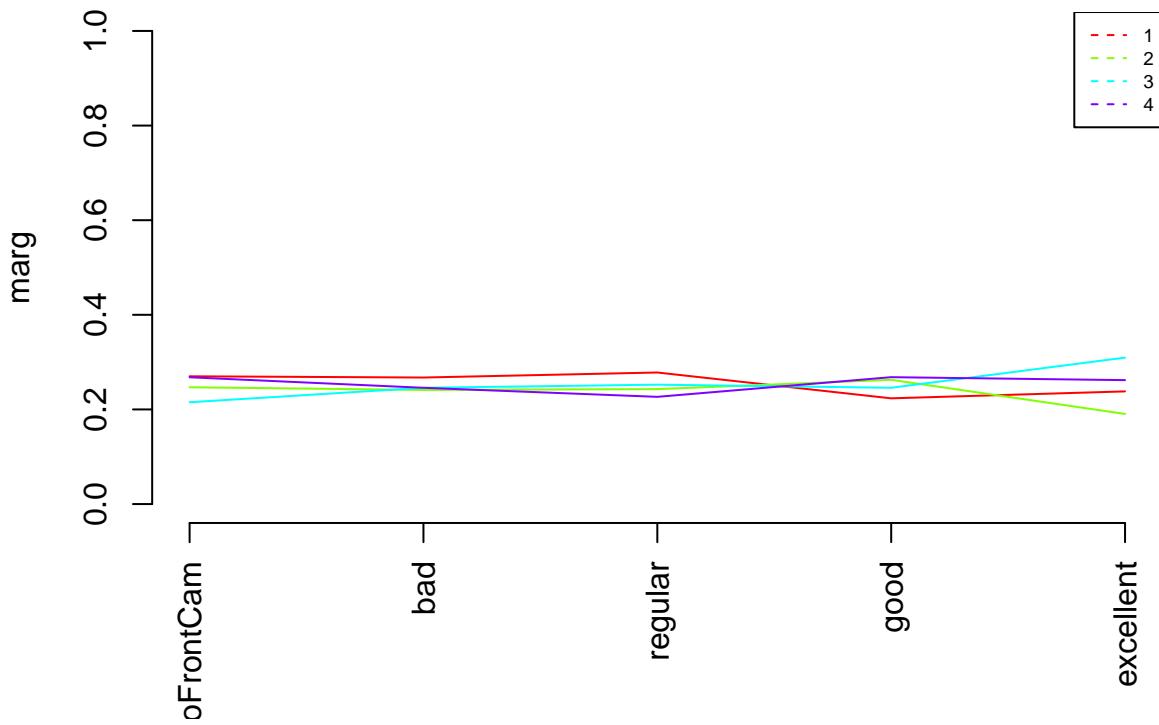


```
## [1] "Categories=" "NoFrontCam"   "bad"          "regular"      "good"        "excellent"
```

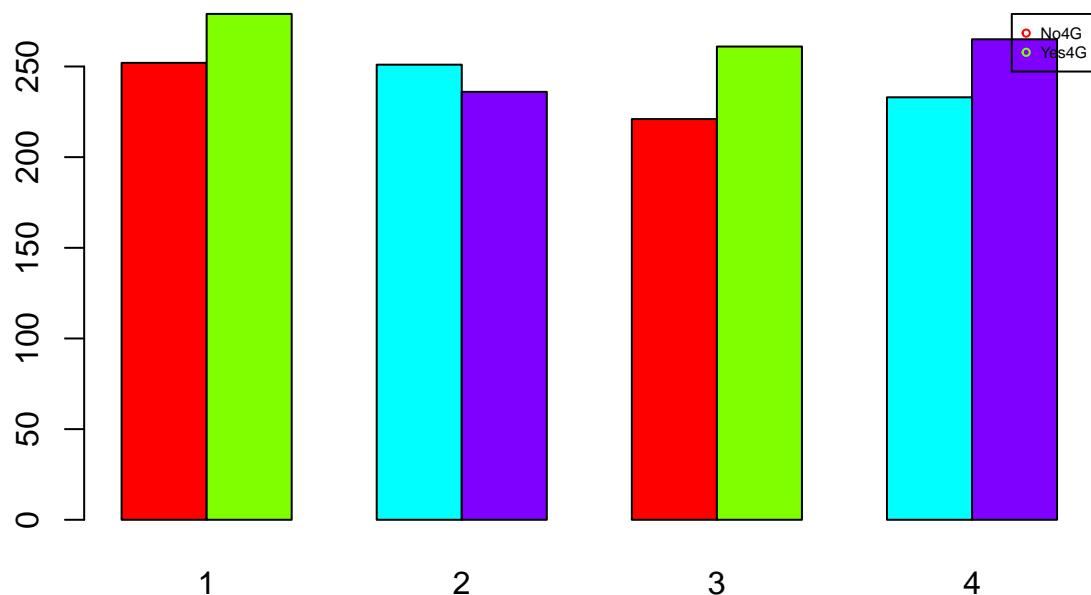
Prop. of pos & neg by fc

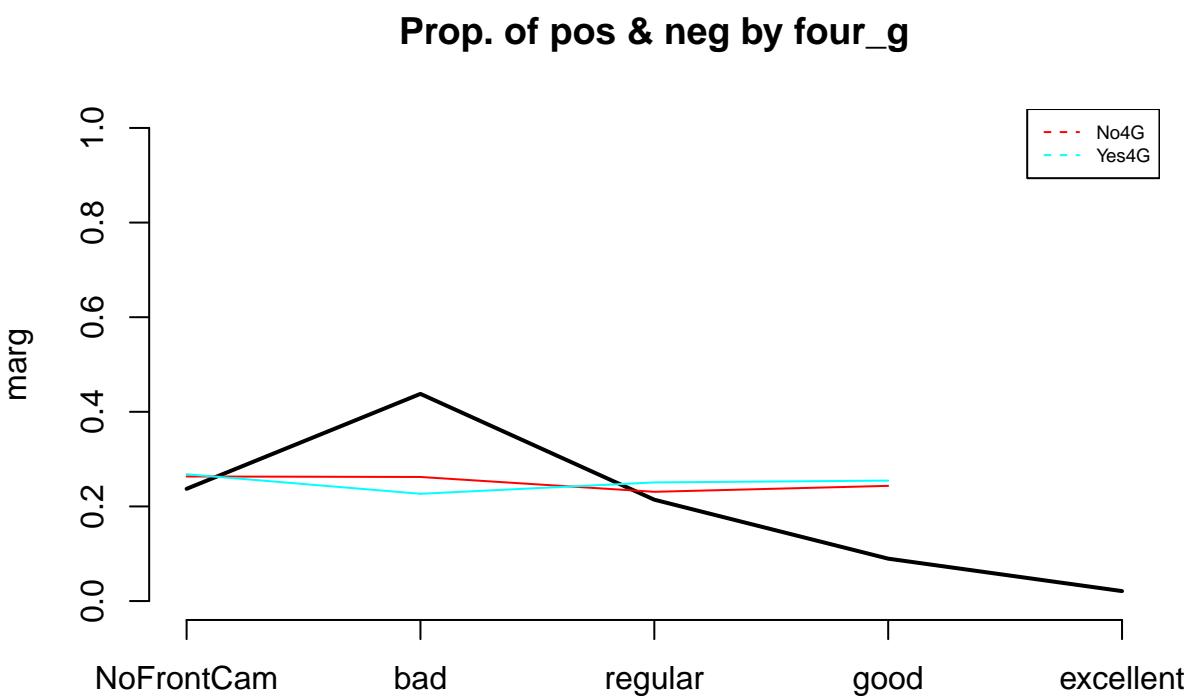


Prop. of pos & neg by fc

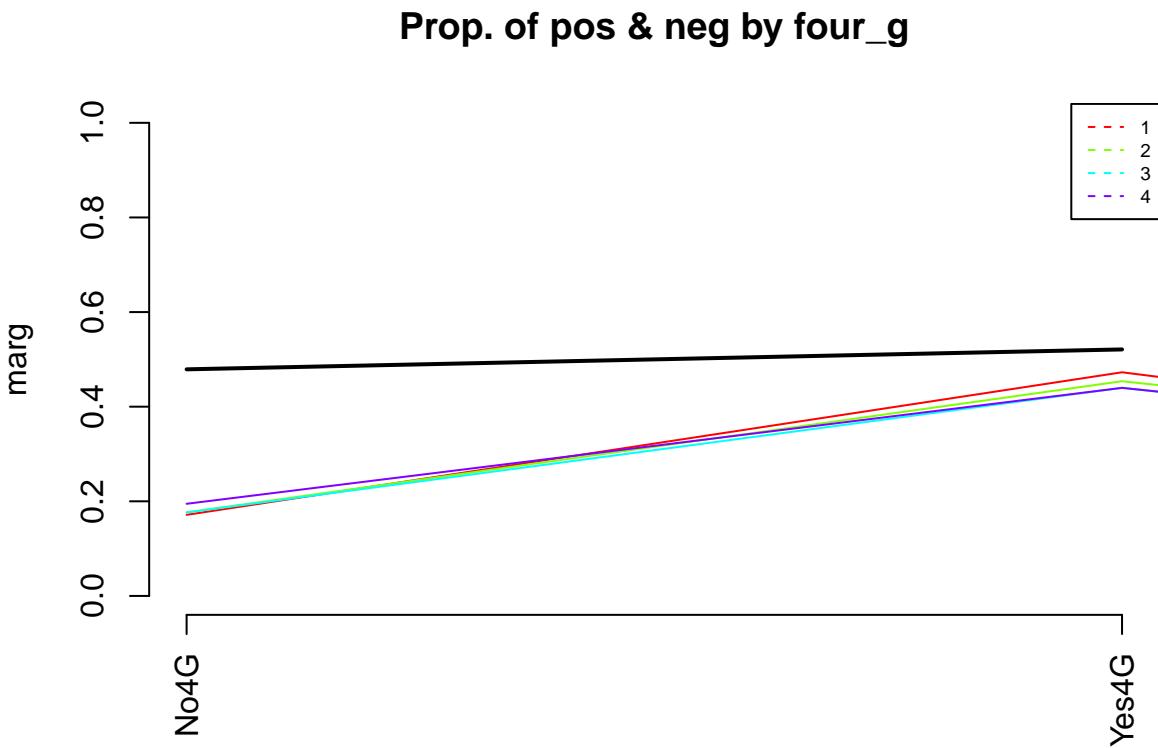


```
## [1] "Variable four_g"
```

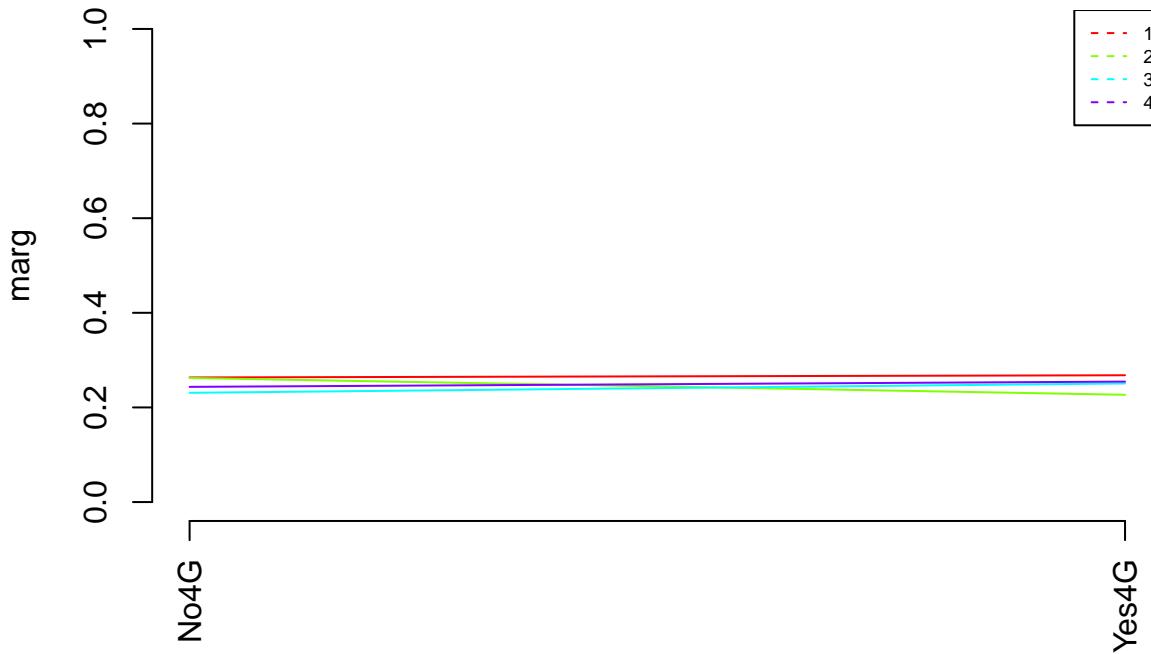




```
## [1] "Categories=" "No4G"           "Yes4G"
```

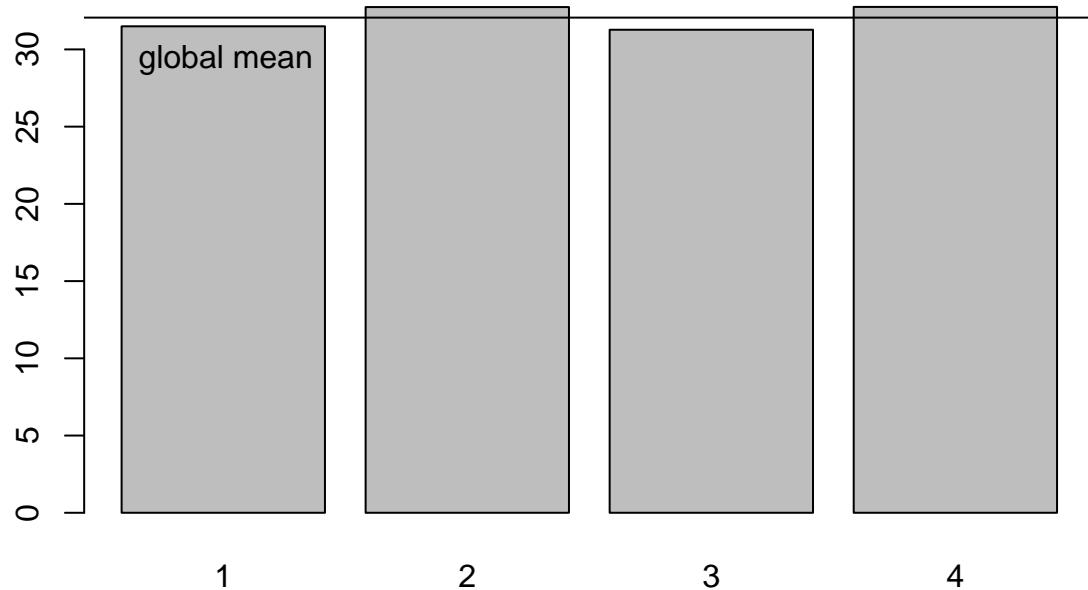


Prop. of pos & neg by four_g



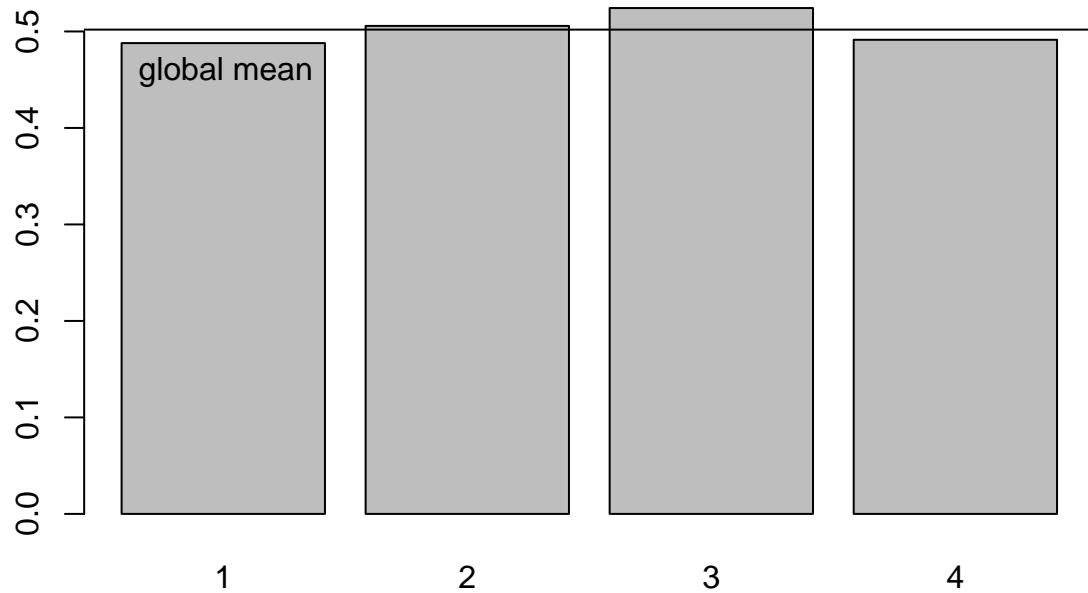
```
## [1] "Variable int_memory"
```

Means of int_memory by Clustering classes



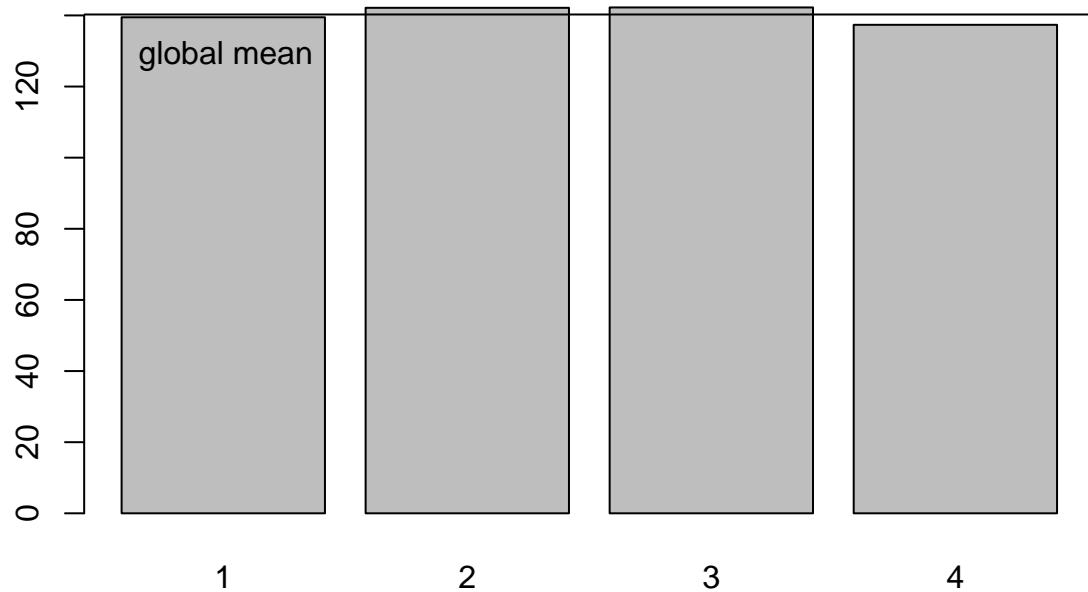
```
## [1] "Variable m_dep"
```

Means of m_dep by Clustering classes



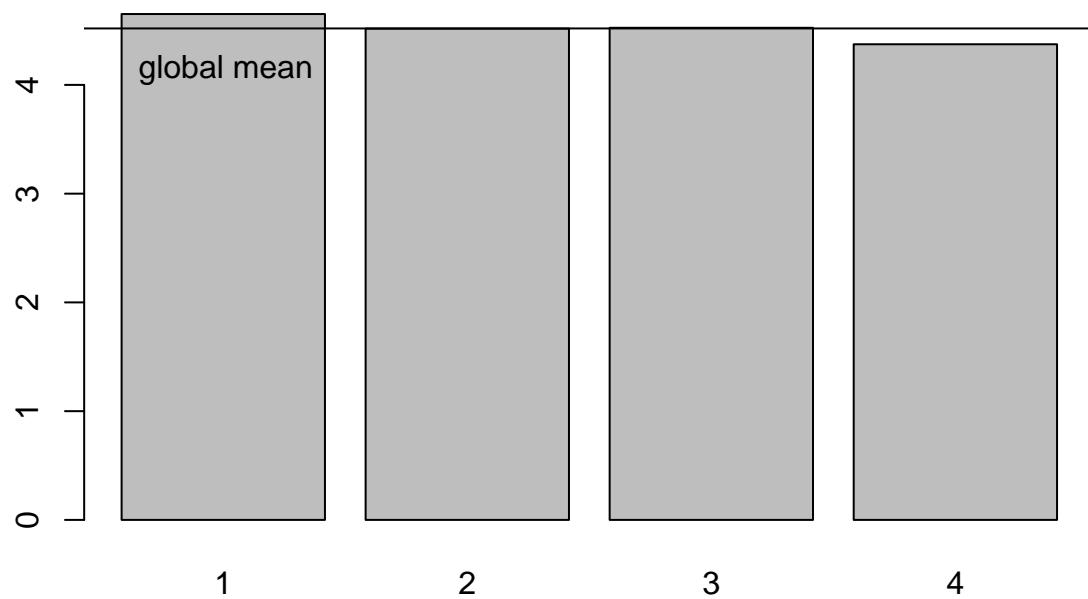
```
## [1] "Variable mobile_wt"
```

Means of mobile_wt by Clustering classes

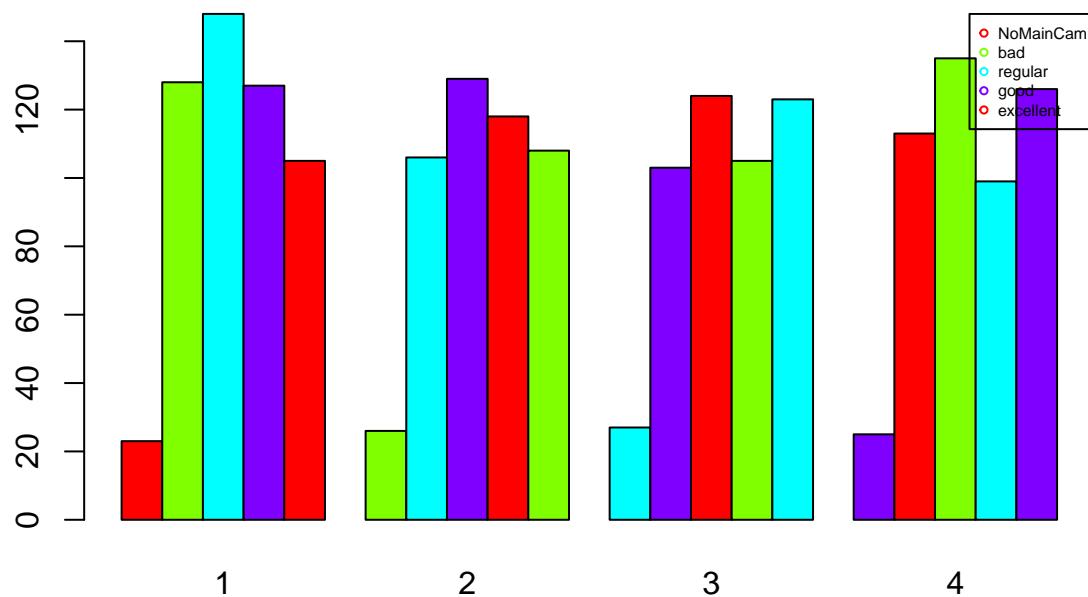


```
## [1] "Variable n_cores"
```

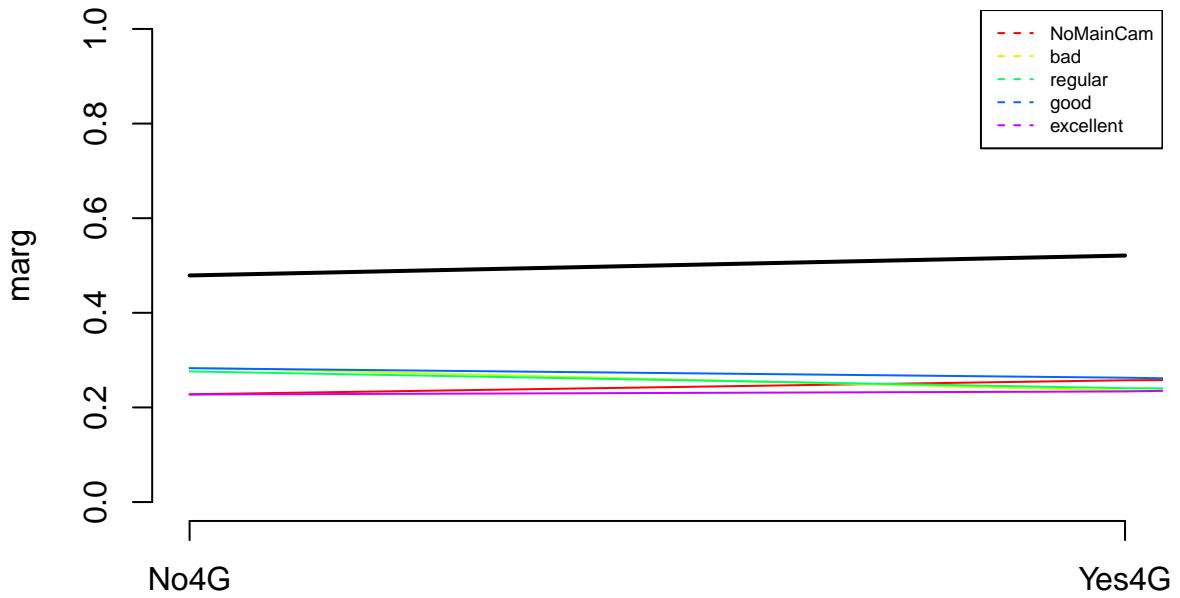
Means of n_cores by Clustering classes



```
## [1] "Variable pc"
```

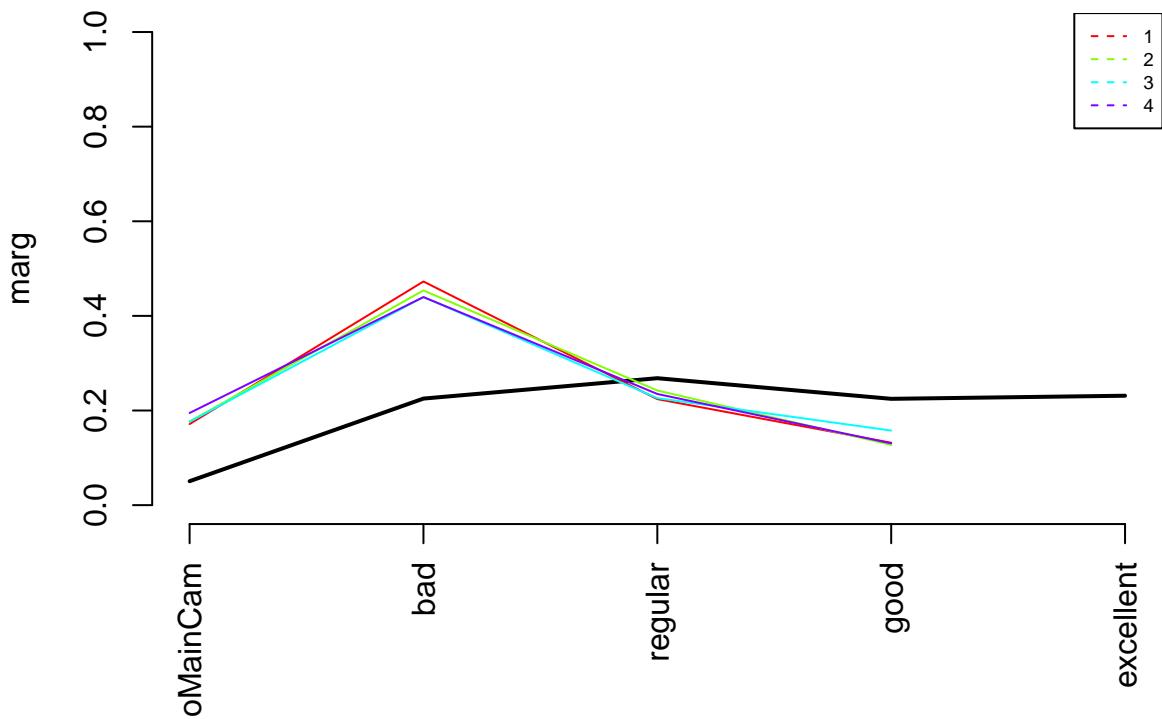


Prop. of pos & neg by pc

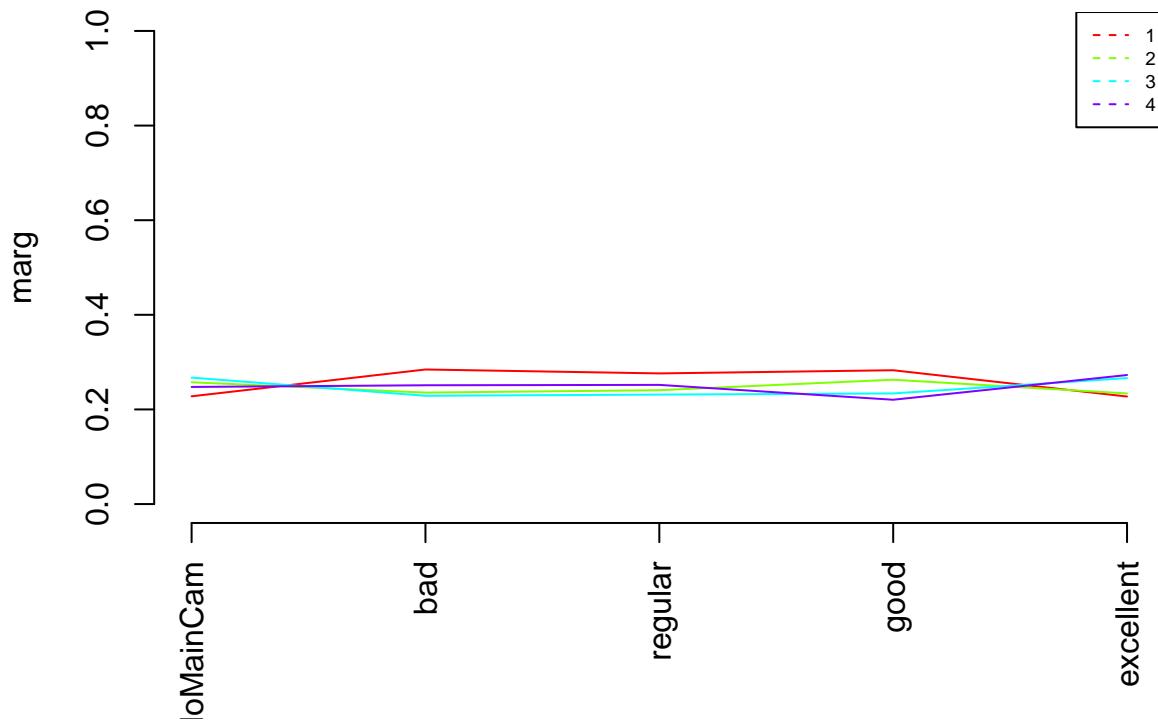


```
## [1] "Categories=" "NoMainCam"    "bad"          "regular"      "good"        "excellent"  
## [6] "excellent"
```

Prop. of pos & neg by pc

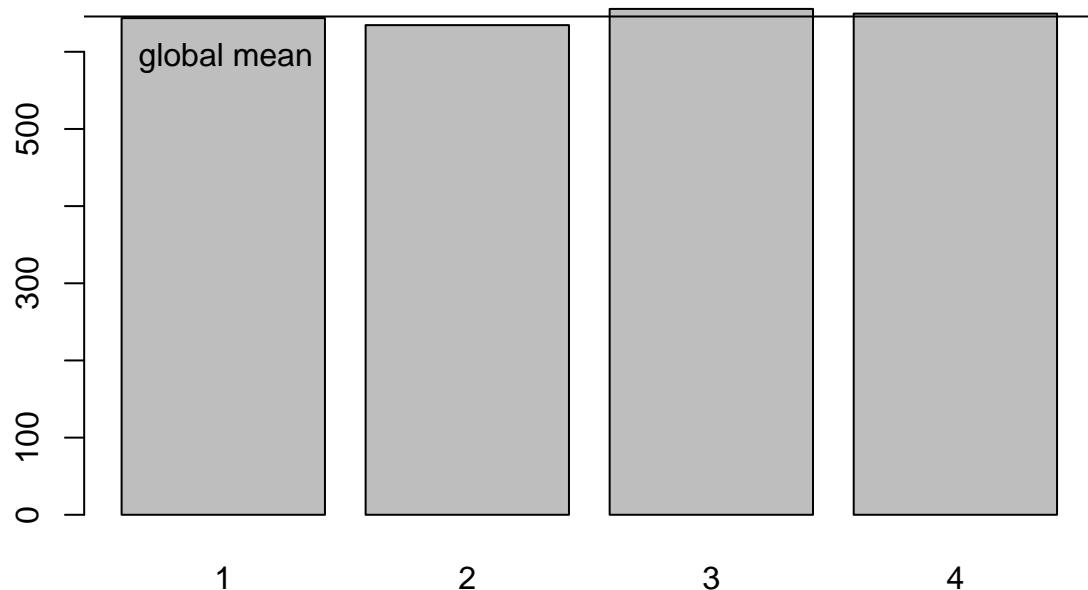


Prop. of pos & neg by pc



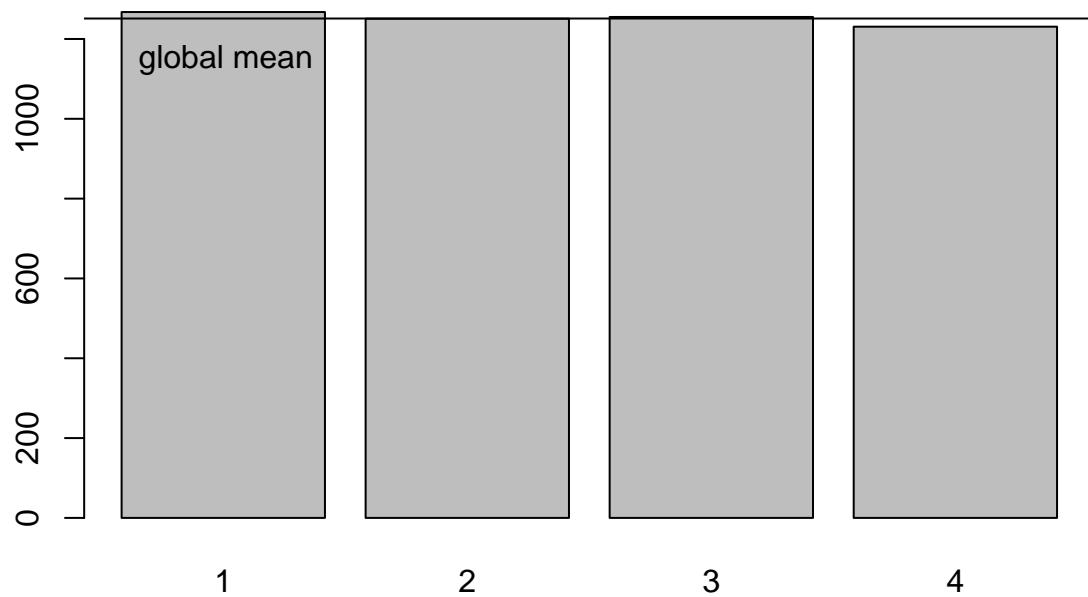
```
## [1] "Variable px_height"
```

Means of px_height by Clustering classes

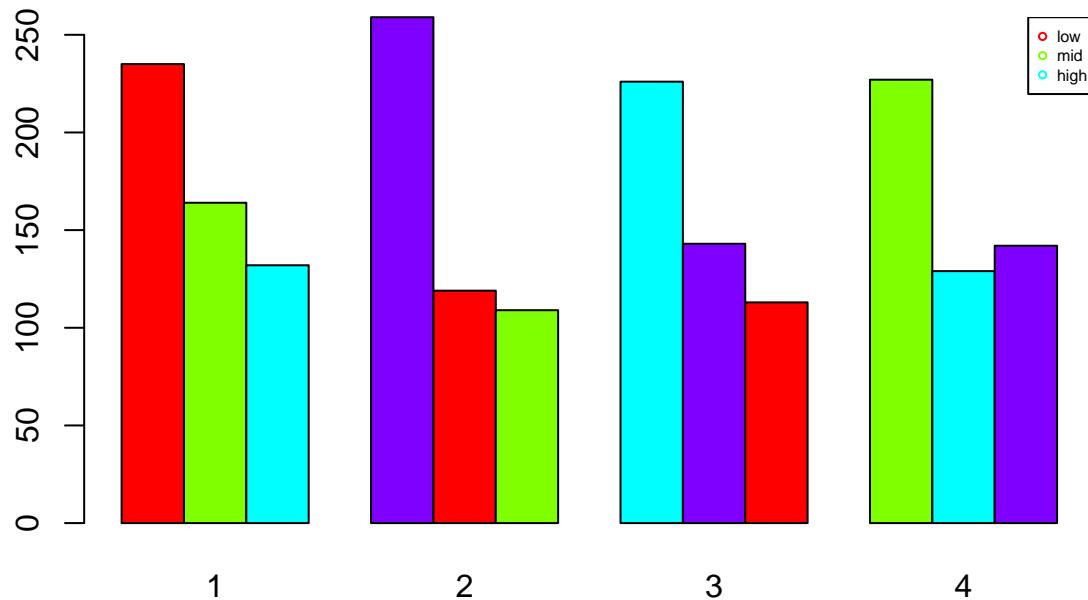


```
## [1] "Variable px_width"
```

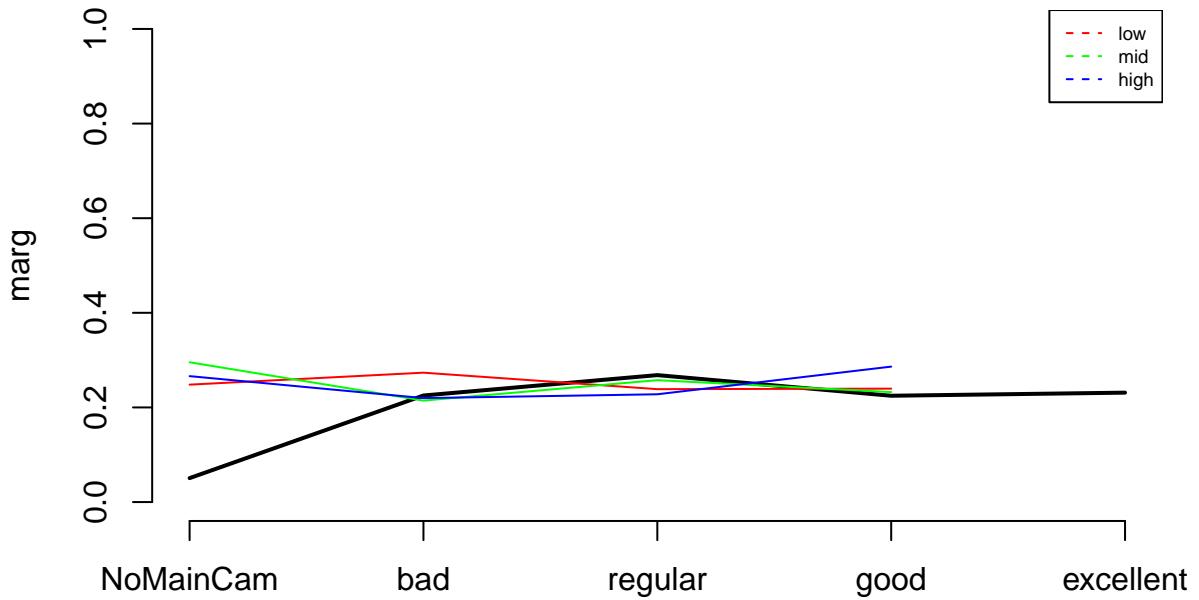
Means of px_width by Clustering classes



```
## [1] "Variable ram"
```

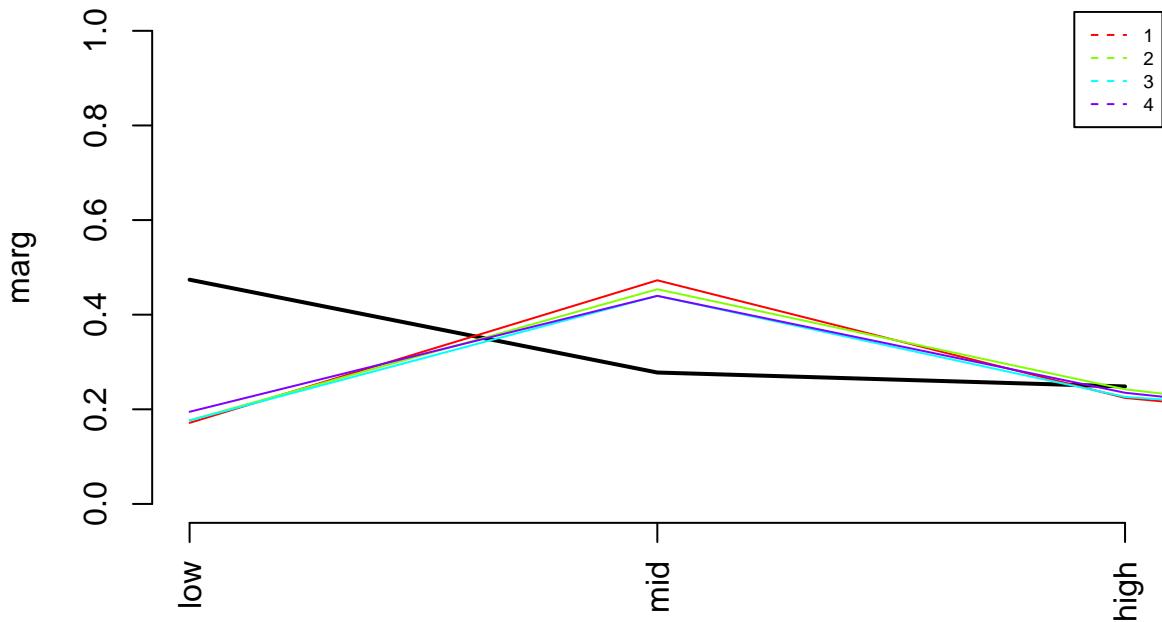


Prop. of pos & neg by ram

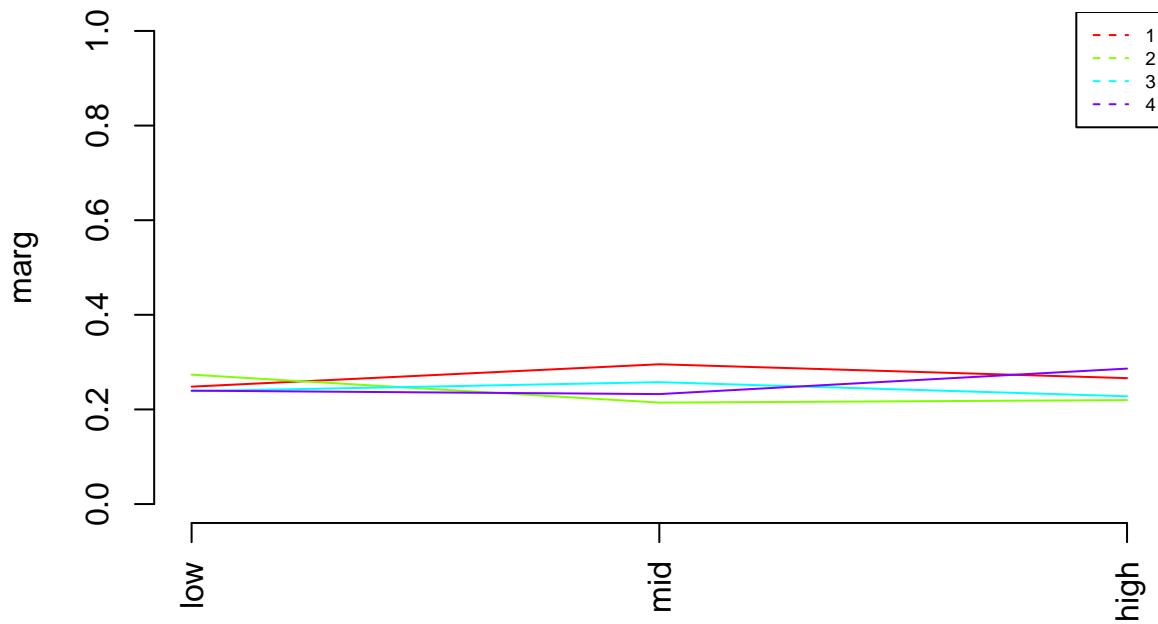


```
## [1] "Categories=" "low"      "mid"      "high"
```

Prop. of pos & neg by ram

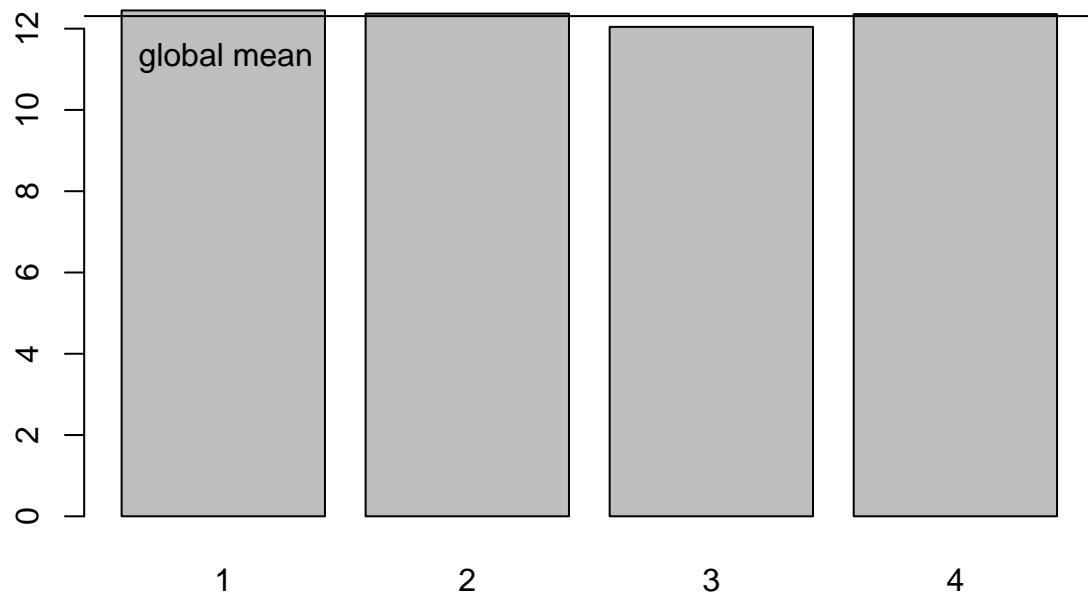


Prop. of pos & neg by ram



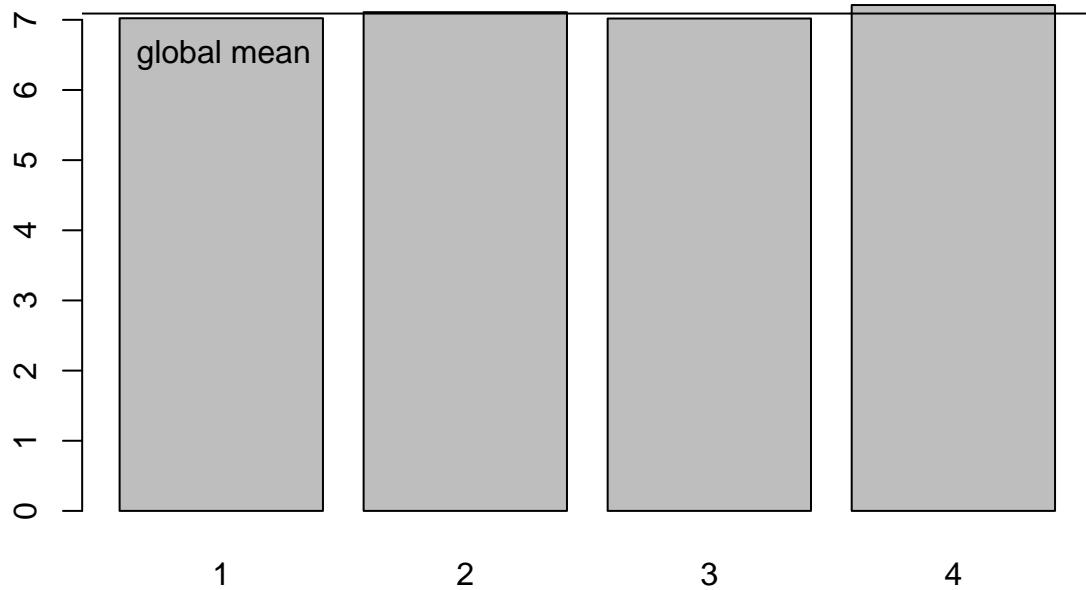
```
## [1] "Variable sc_h"
```

Means of sc_h by Clustering classes



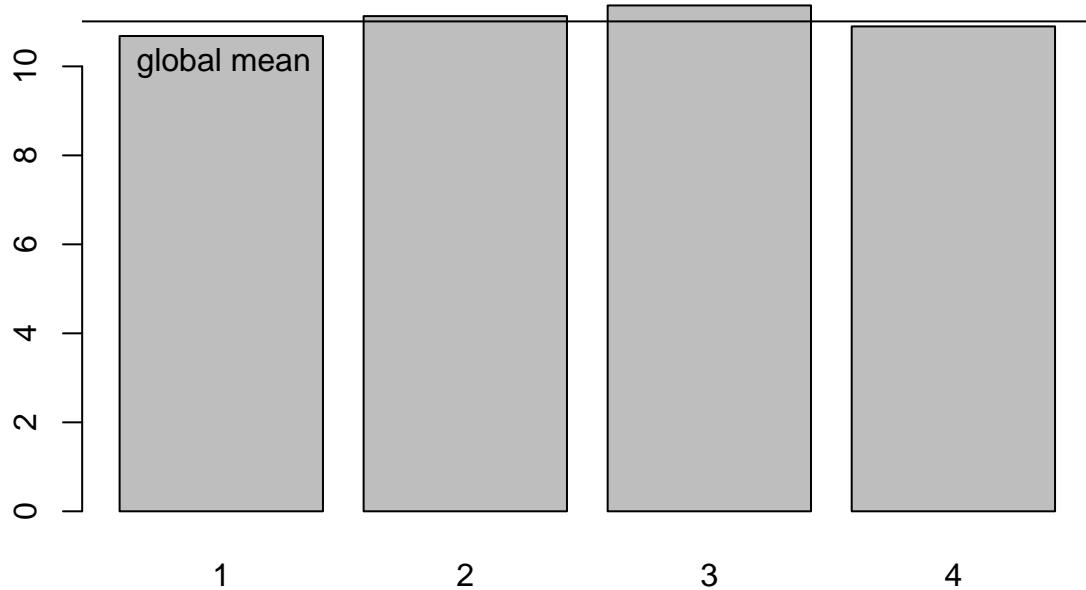
```
## [1] "Variable sc_w"
```

Means of sc_w by Clustering classes

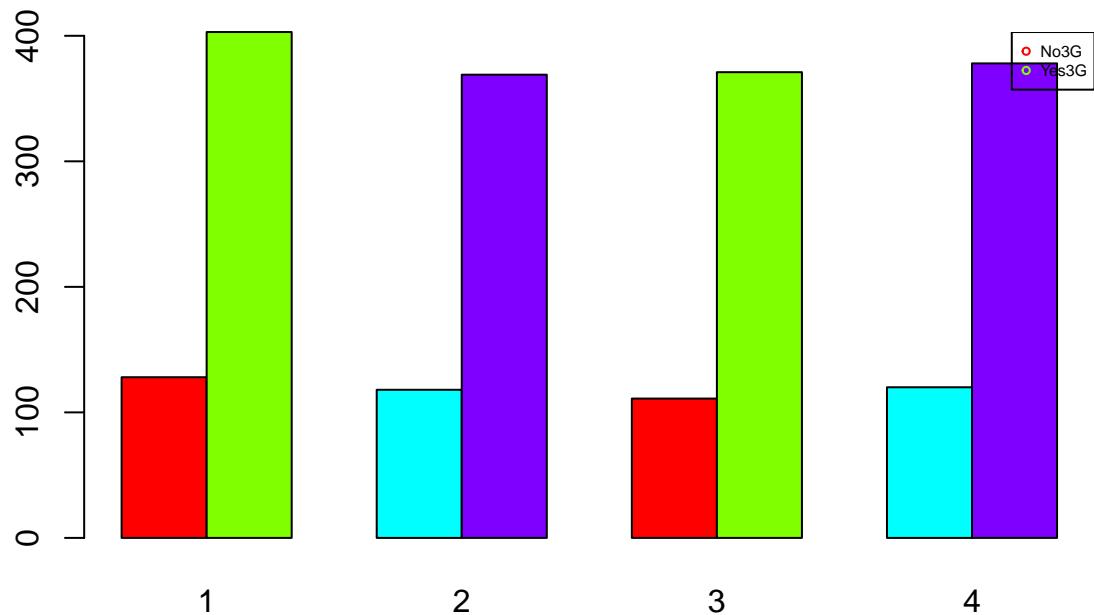


```
## [1] "Variable talk_time"
```

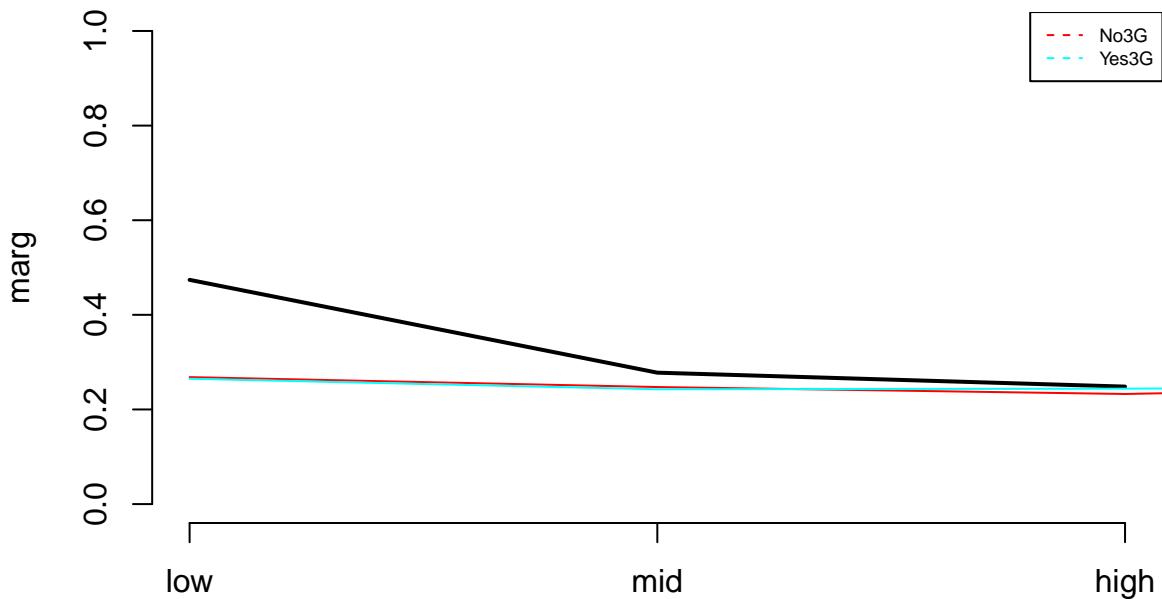
Means of talk_time by Clustering classes



```
## [1] "Variable three_g"
```

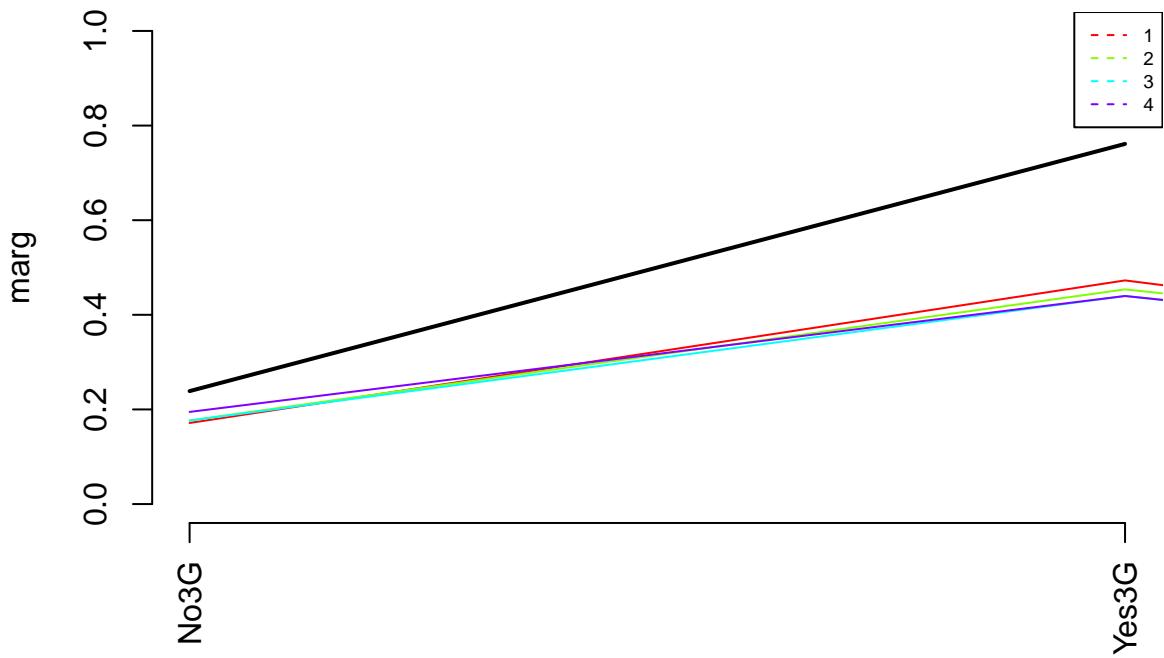


Prop. of pos & neg by three_g

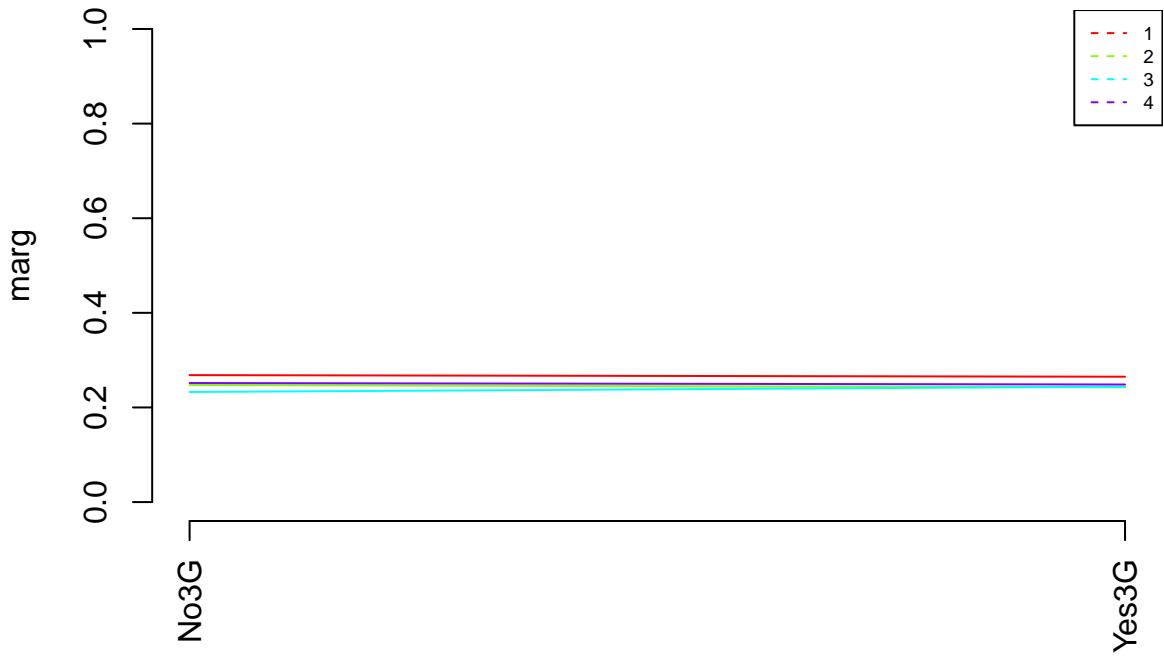


```
## [1] "Categories=" "No3G"      "Yes3G"
```

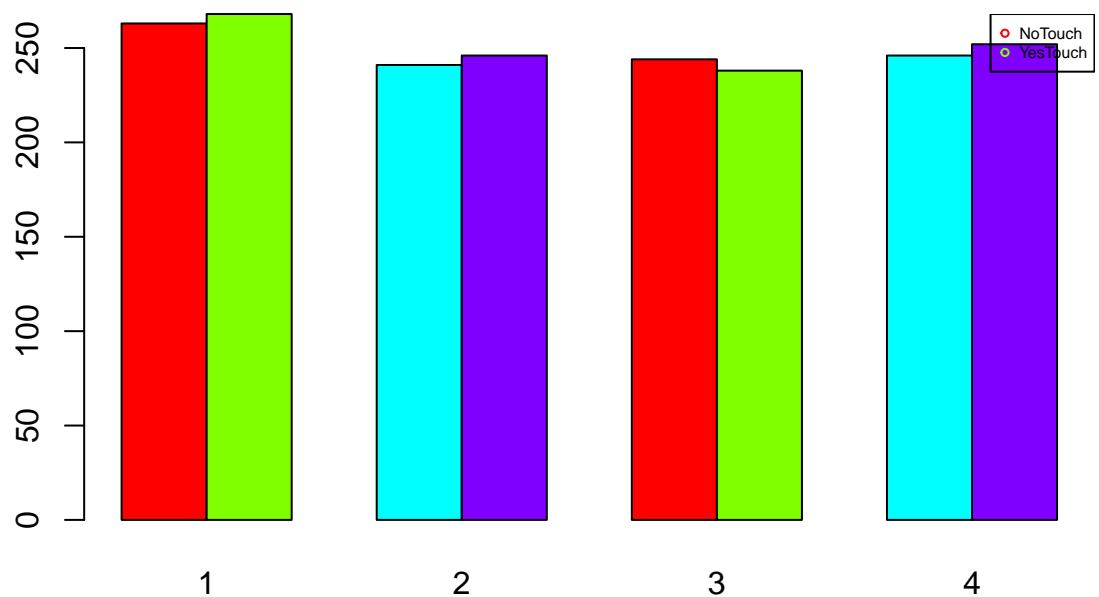
Prop. of pos & neg by three_g



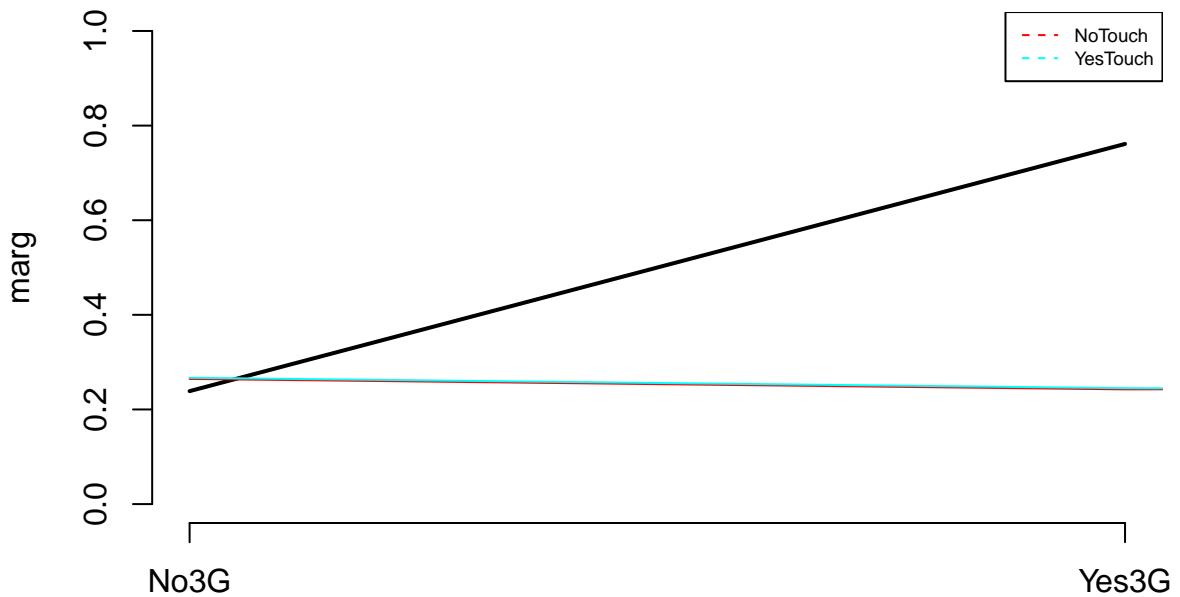
Prop. of pos & neg by three_g



```
## [1] "Variable touch_screen"
```

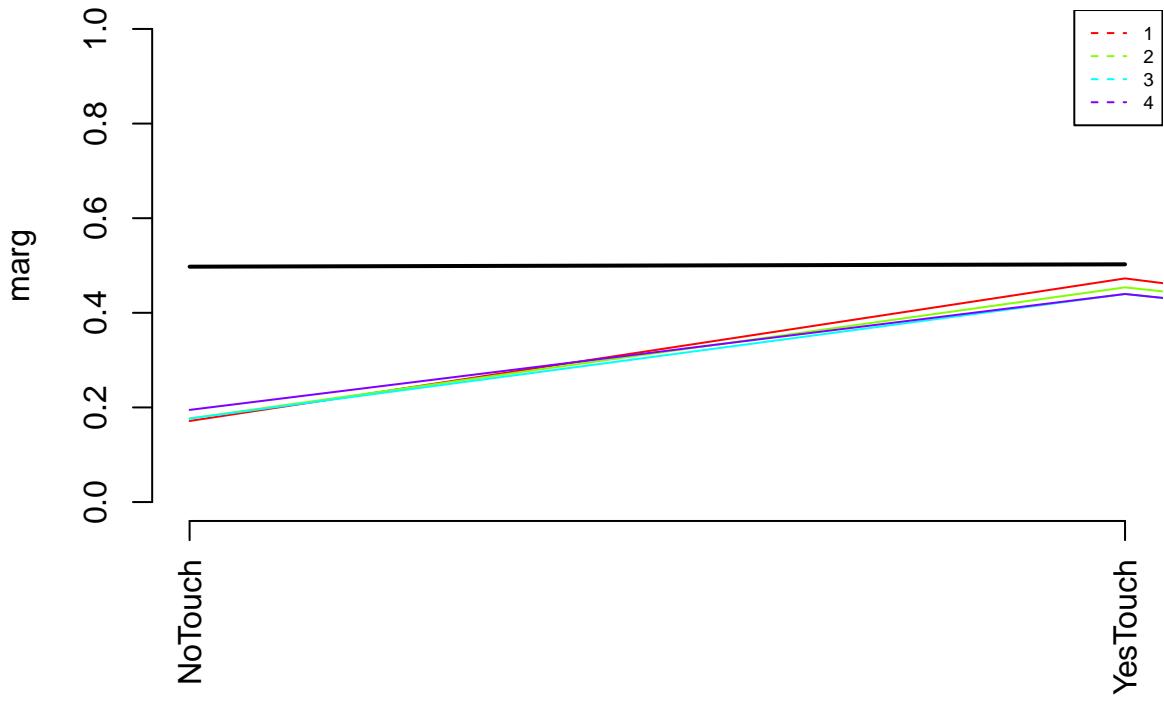


Prop. of pos & neg by touch_screen



```
## [1] "Categories=" "NoTouch"      "YesTouch"
```

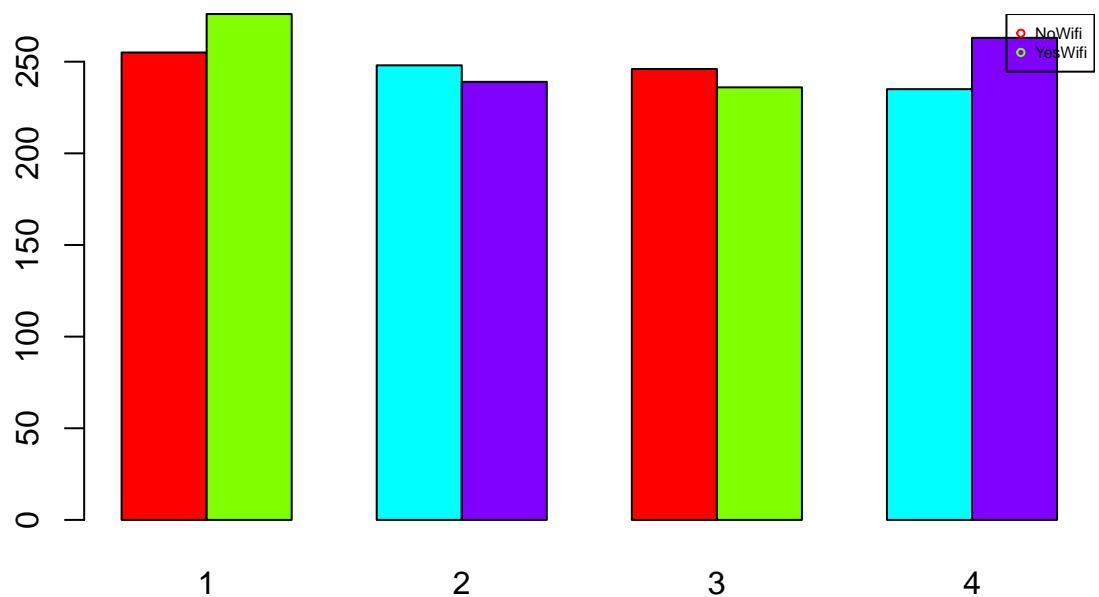
Prop. of pos & neg by touch_screen



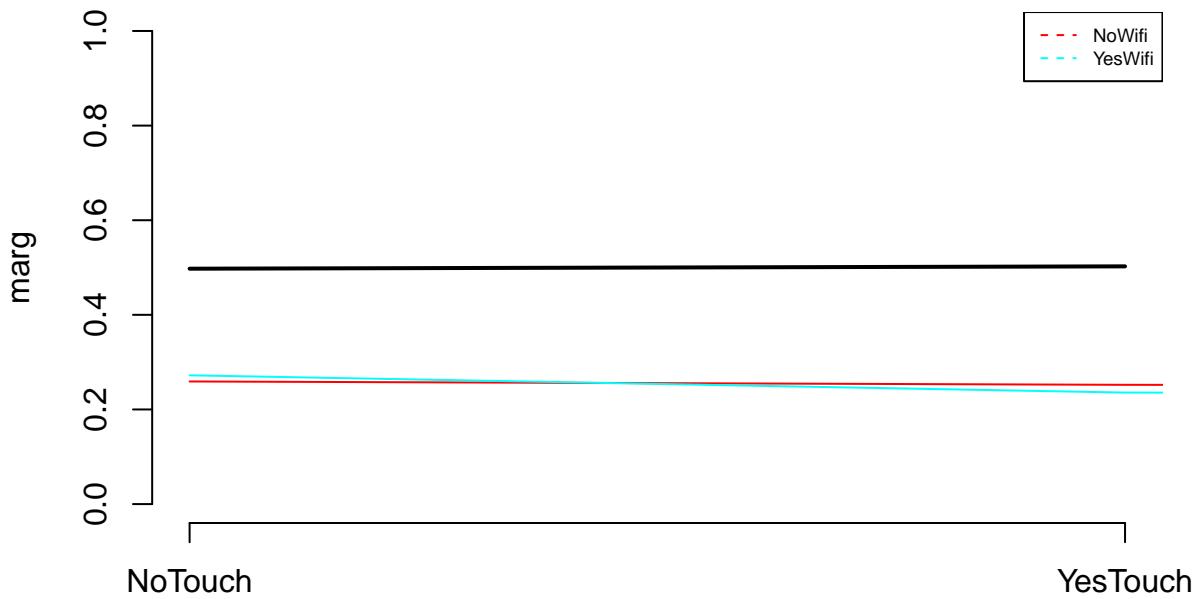
Prop. of pos & neg by touch_screen



```
## [1] "Variable wifi"
```

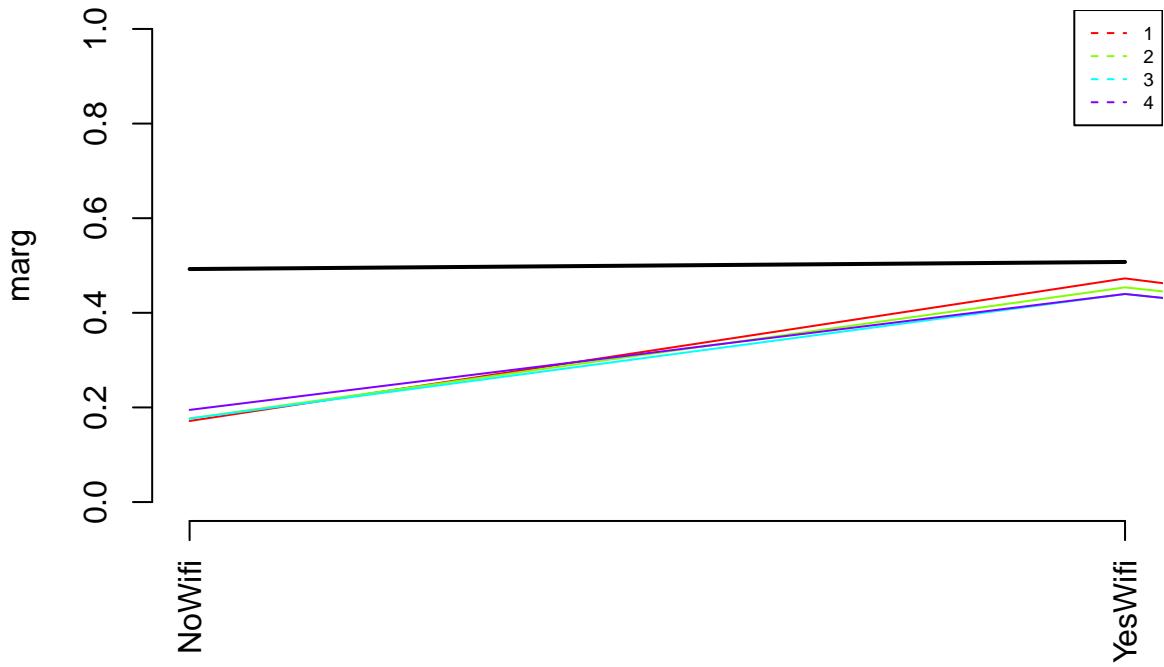


Prop. of pos & neg by wifi

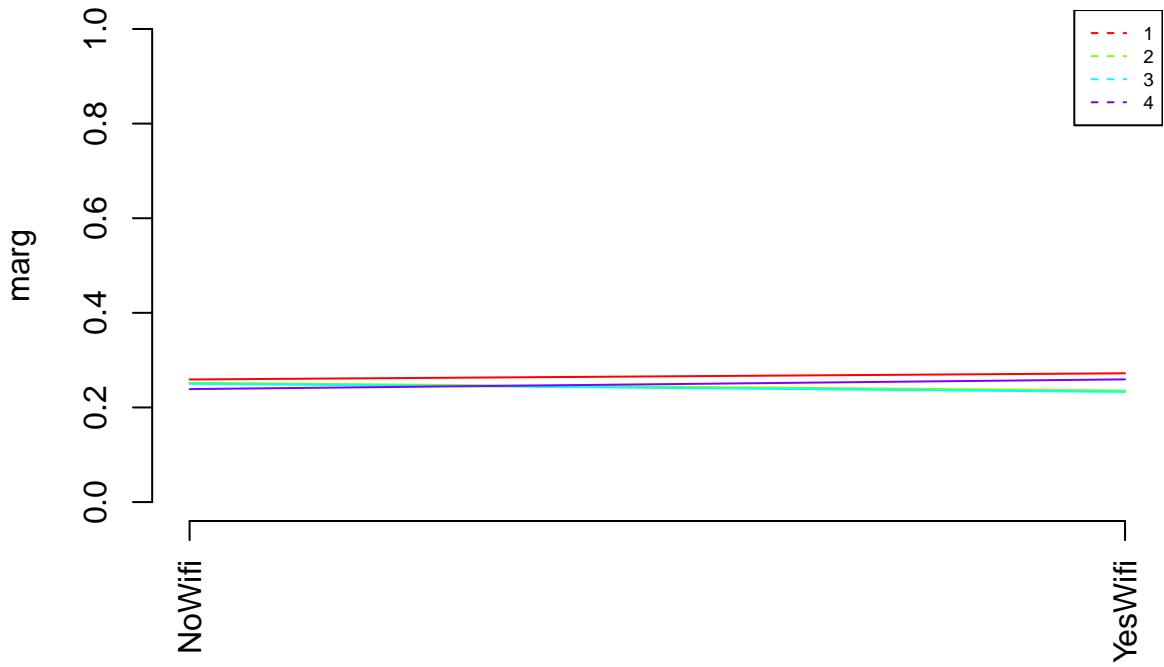


```
## [1] "Categories=" "NoWifi"      "YesWifi"
```

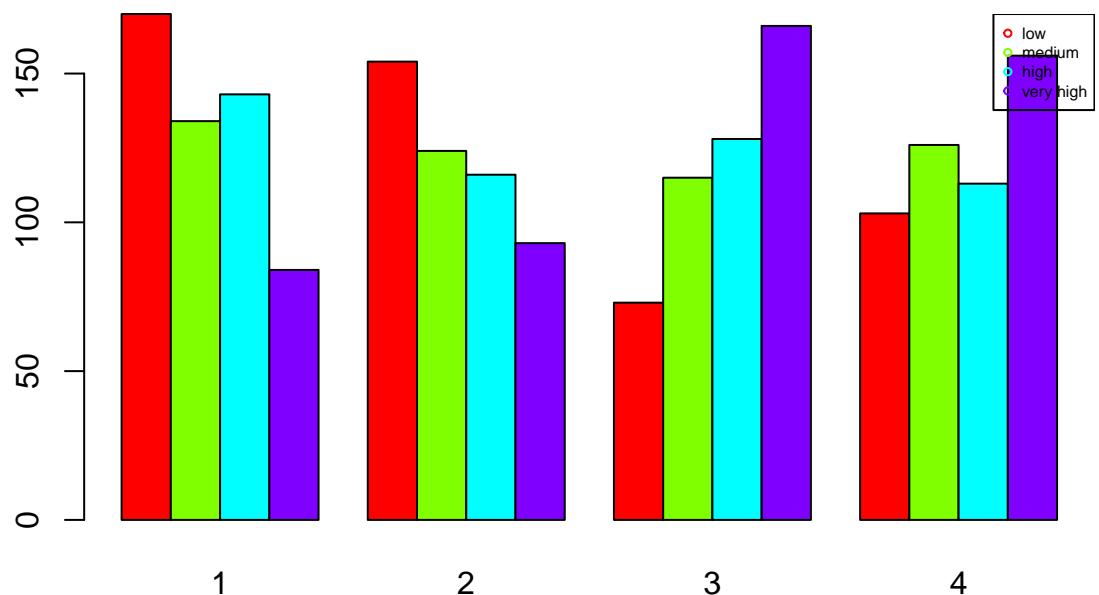
Prop. of pos & neg by wifi



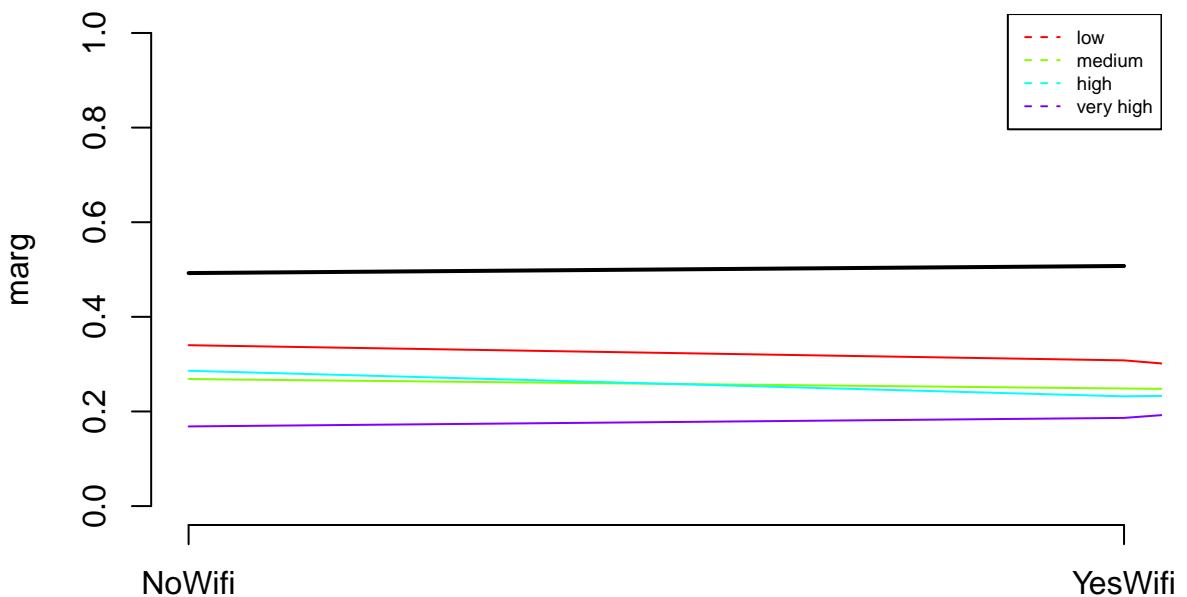
Prop. of pos & neg by wifi



```
## [1] "Variable price_range"
```

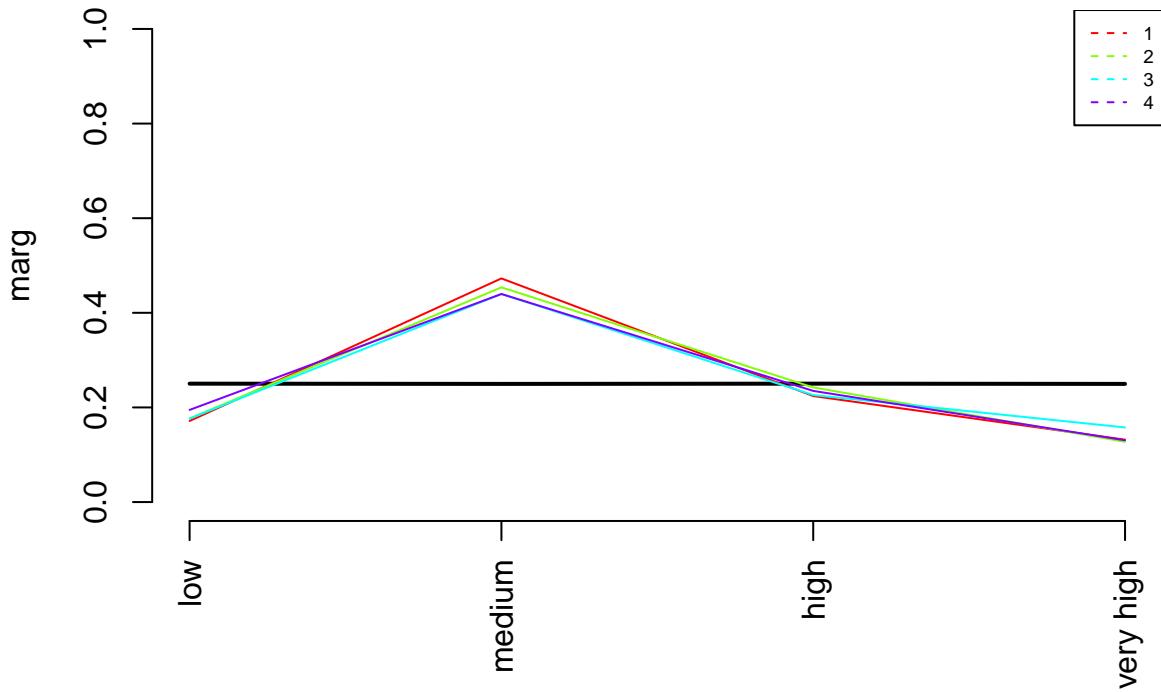


Prop. of pos & neg by price_range

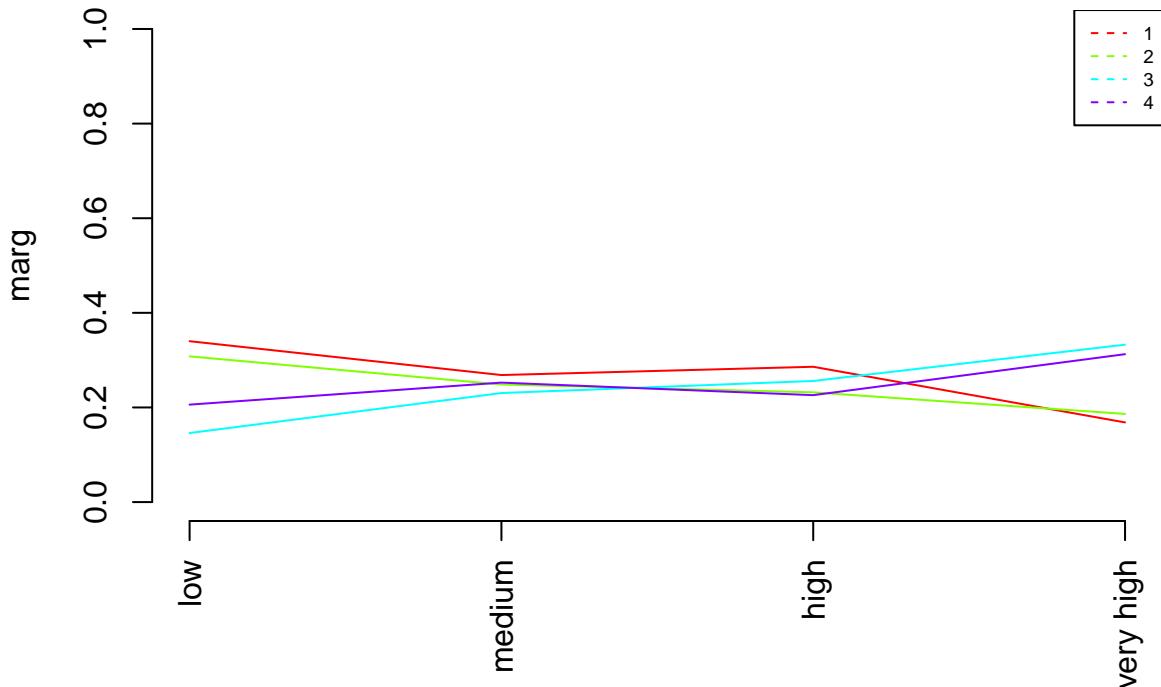


```
## [1] "Categories=" "low"           "medium"        "high"          "very high"
```

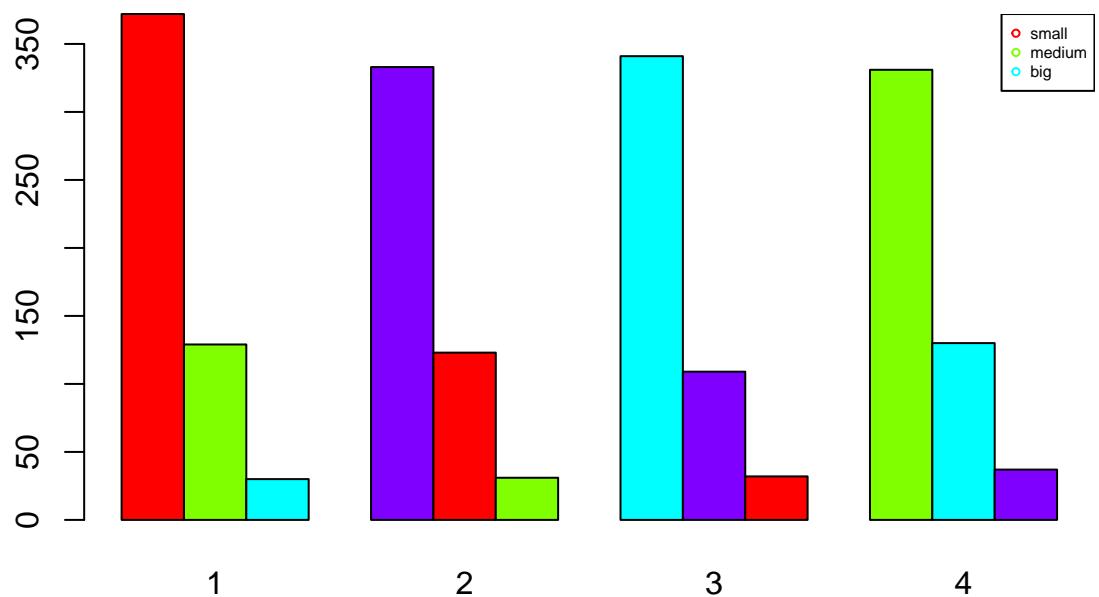
Prop. of pos & neg by price_range



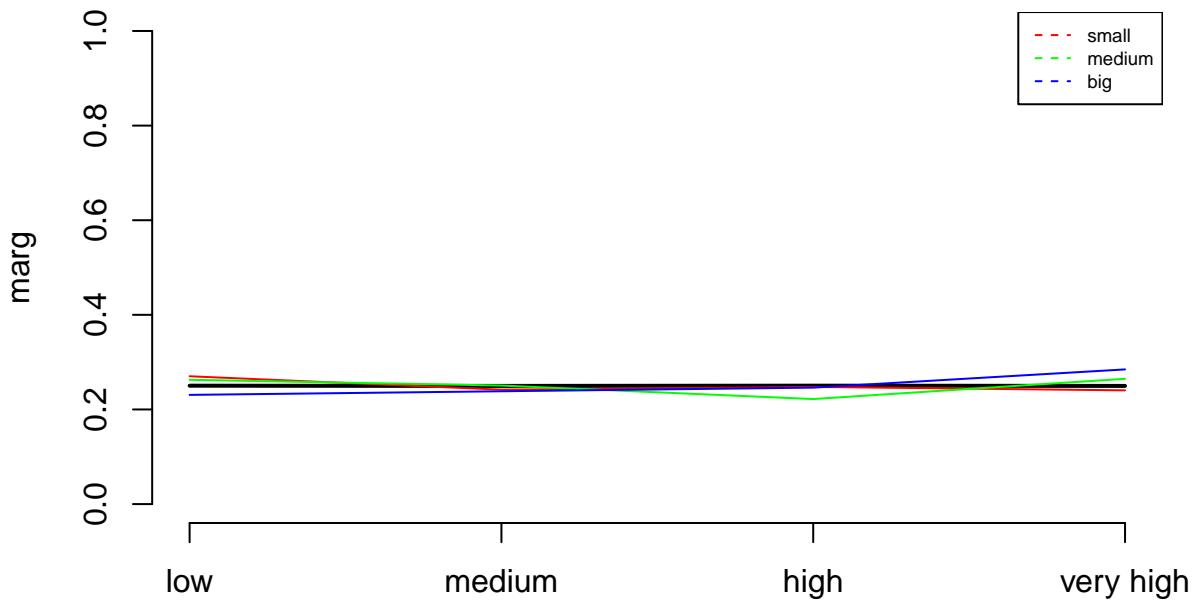
Prop. of pos & neg by price_range



```
## [1] "Variable sc_d"
```

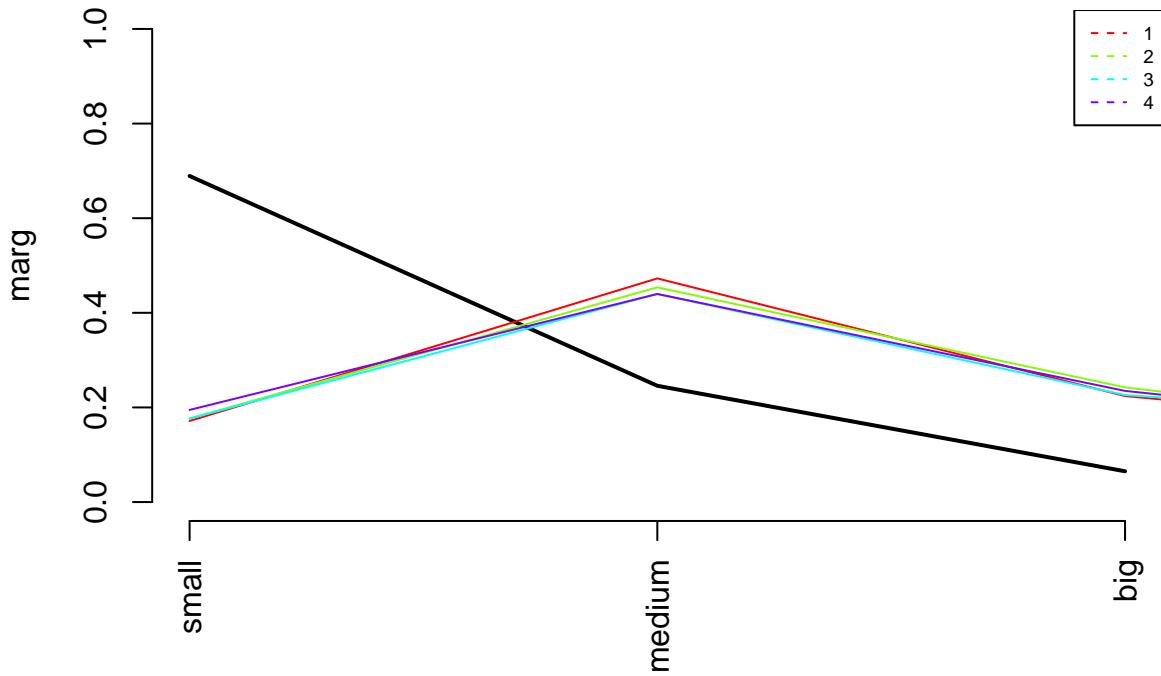


Prop. of pos & neg by sc_d

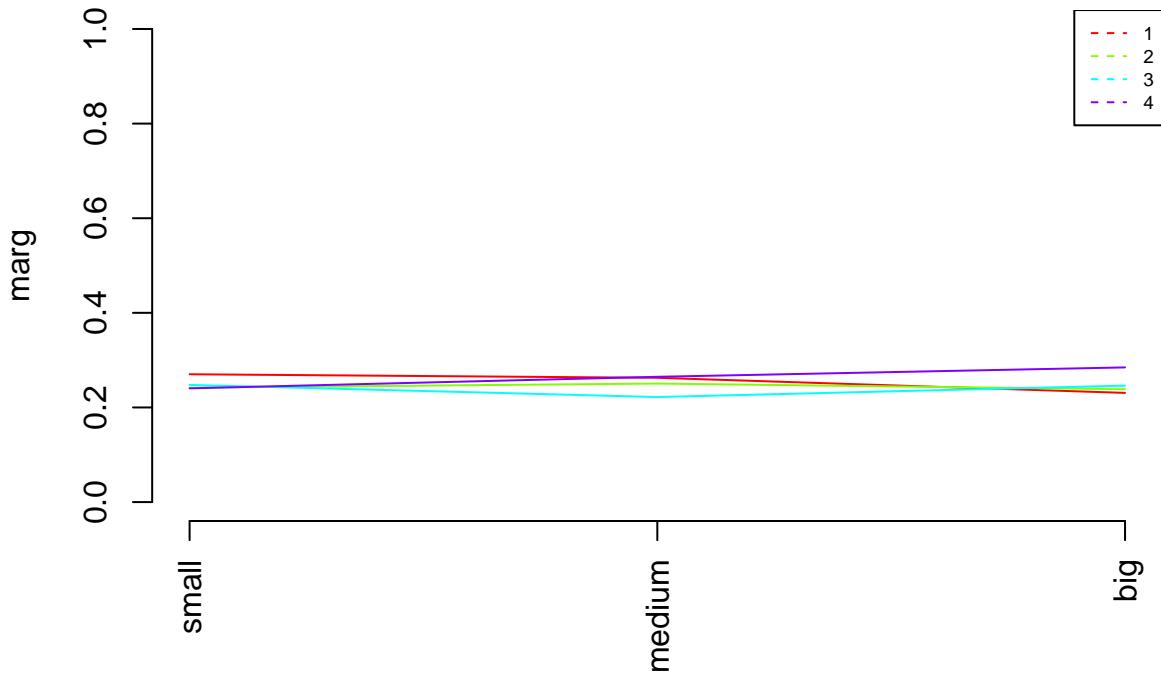


```
## [1] "Categories=" "small"      "medium"      "big"
```

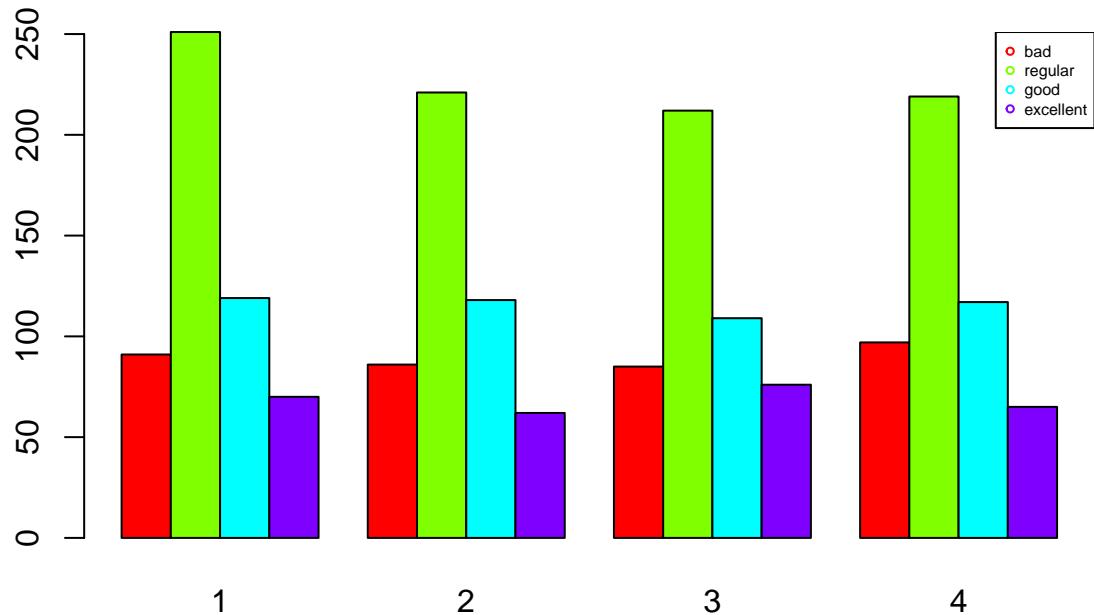
Prop. of pos & neg by sc_d



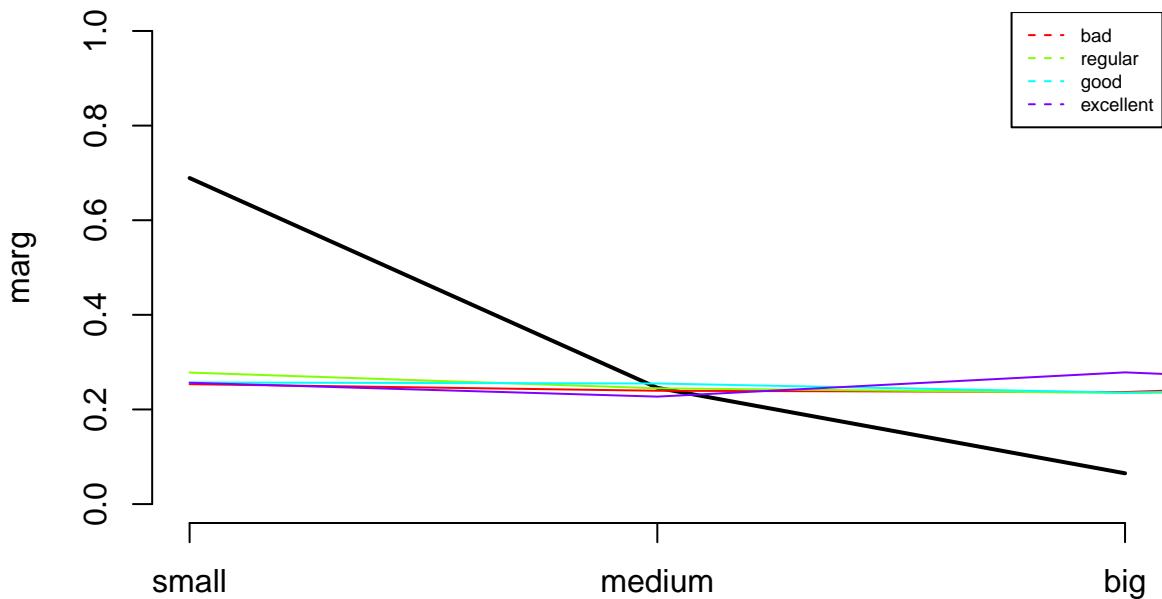
Prop. of pos & neg by sc_d



```
## [1] "Variable px_r"
```

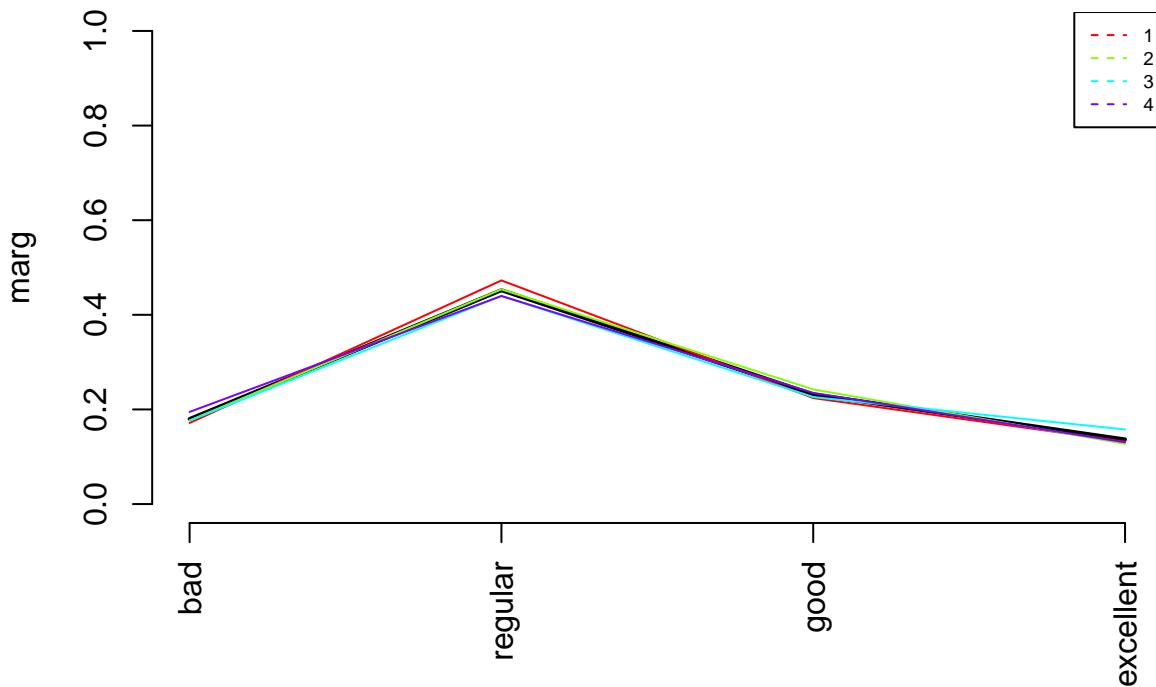


Prop. of pos & neg by px_r

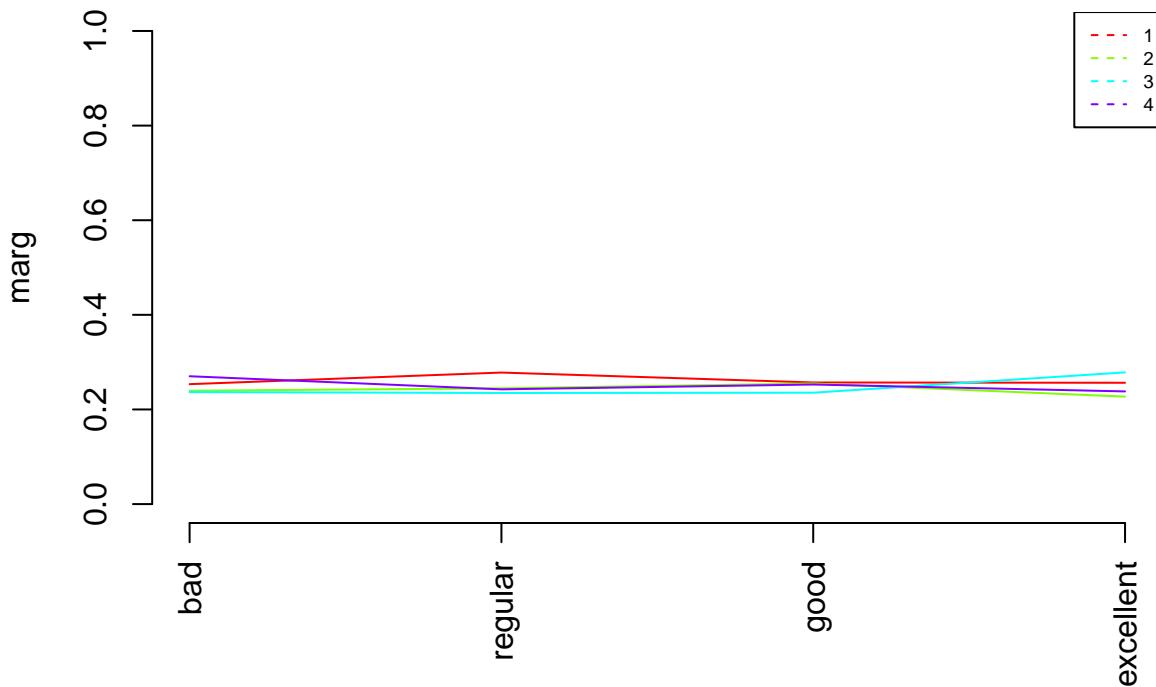


```
## [1] "Categories=" "bad"           "regular"        "good"          "excellent"
```

Prop. of pos & neg by px_r



Prop. of pos & neg by px_r



11 Bibliography

<https://www.multiofera.es/smartphones-financiados/preguntas-frecuentes/caracteristicas-moviles-gama-alta-media-baja-30.html>

<https://androidayuda.com/ca/como-calcular-la-densidad-de-pixeles-ppi-de-una-pantalla/>

<https://norfipc.com/celulares/medidas-pantalla-resolucion-telefonos-celulares-tabletas.html>

<https://blog.pepephone.com/diferencias-movil-gama-baja-media-alta>

<https://pluscases.com/blogs/battery-case-news/understanding-your-phone-what-does-mah-mean-on-a-batter>