# Statistical-Data-Science-Project-Academic-year-2022-2023

## 0 Prep

### 0.1 libraries

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
library(robustbase)
```

```
## Warning: package 'robustbase' was built under R version 4.1.3
```

```
library(rrcov)
```

```
## Warning: package 'rrcov' was built under R version 4.1.3
```

```
## Scalable Robust Estimators with High Breakdown Point (version 1.7-2)
```

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.1.3
```

```
library(cluster)
library(stats)
library(dbscan)
```

```
## Warning: package 'dbscan' was built under R version 4.1.3
```

```
##
## Attaching package: 'dbscan'
```

```
## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

```
library(lars)
```

```
## Warning: package 'lars' was built under R version 4.1.3
```

```
## Loaded lars 1.3
```

```
library(classmap)
```

```
## Warning: package 'classmap' was built under R version 4.1.3
```

```
library(class)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.3
```

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

### 0.2 loading data set

```
cars_data<-read.csv("~/GitHub/Statistical-Data-Science-Project-Academic-year-2022-2023/cars_data")
```

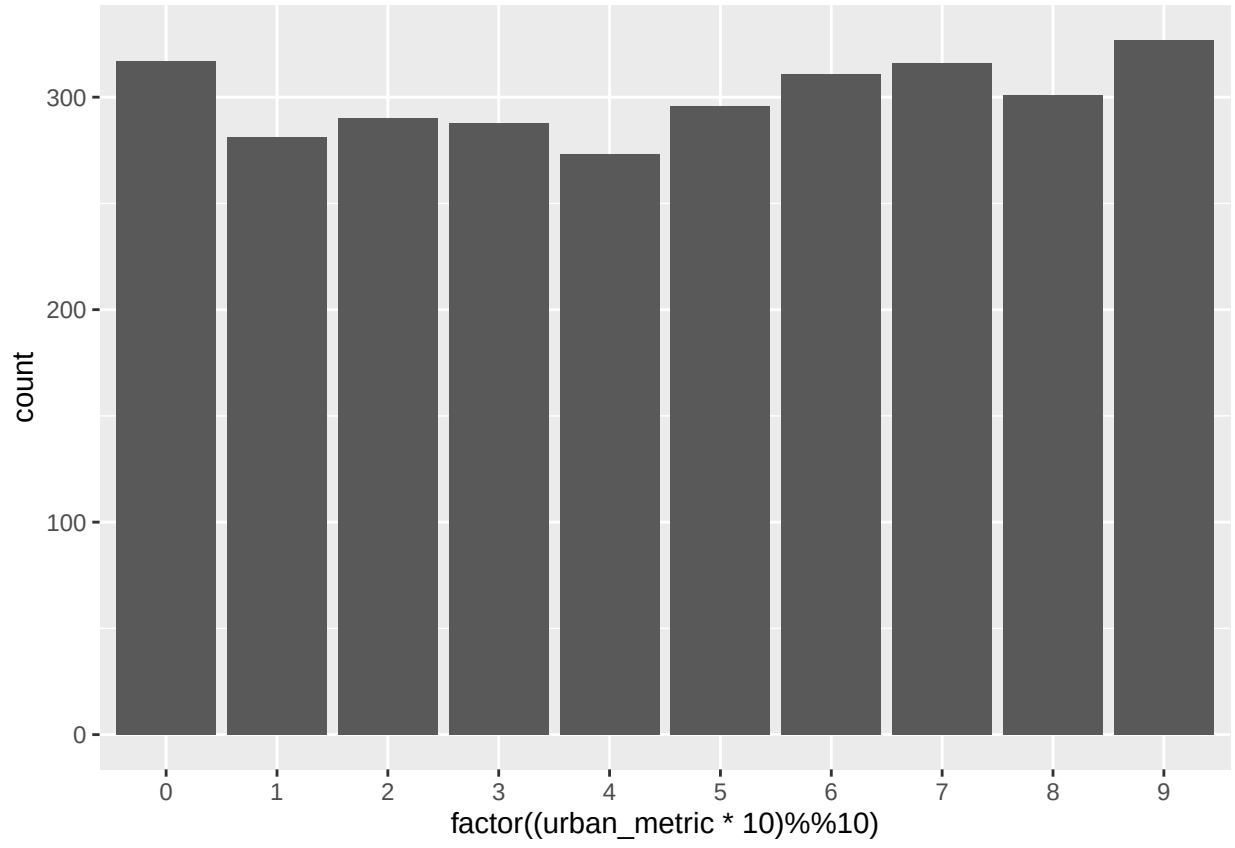
### 0.3 generate subset

```
set.seed(180901)
cars_data$euro_standard <- as.factor(cars_data$euro_standard)
cars_data$transmission_type <- as.factor(cars_data$transmission_type)
cars_data$fuel_type <- as.factor(cars_data$fuel_type)
subset_cars = cars_data[sample(nrow(cars_data),3000,replace=FALSE),]
```

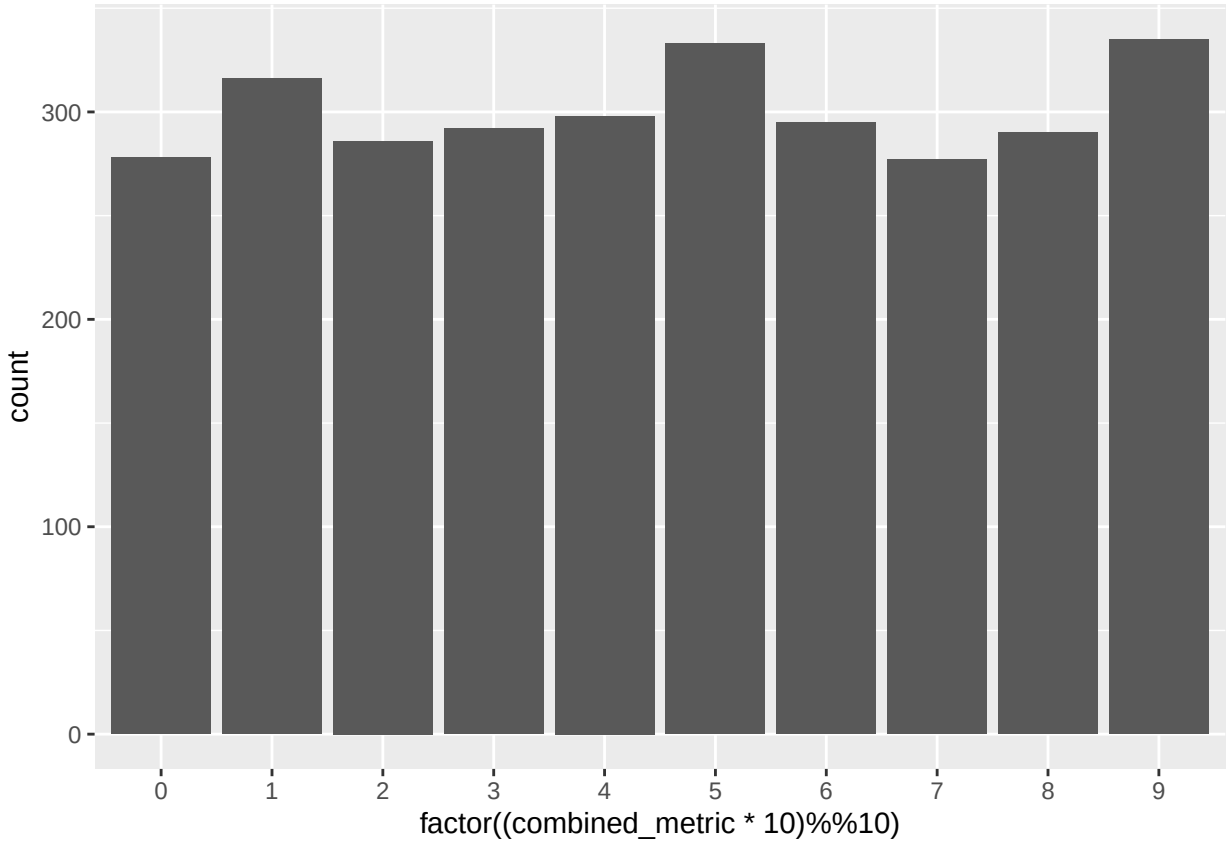
# 1 EDA

I reviewed the data set in a csv viewer to get a feel for it. Noise level has weird rounding thing going on.

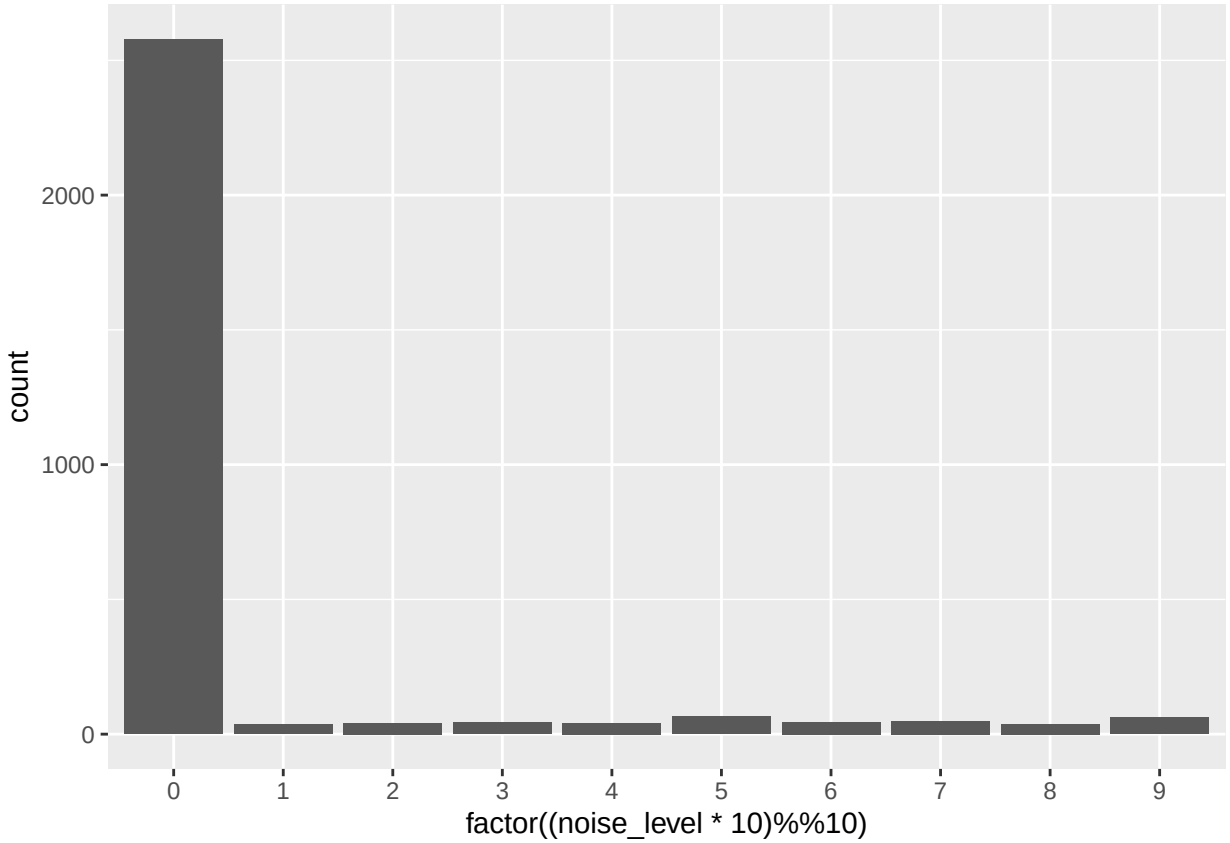
```
# what I expect:
ggplot(subset_cars,aes(factor((urban_metric*10)%%10)))+ geom_bar()
```



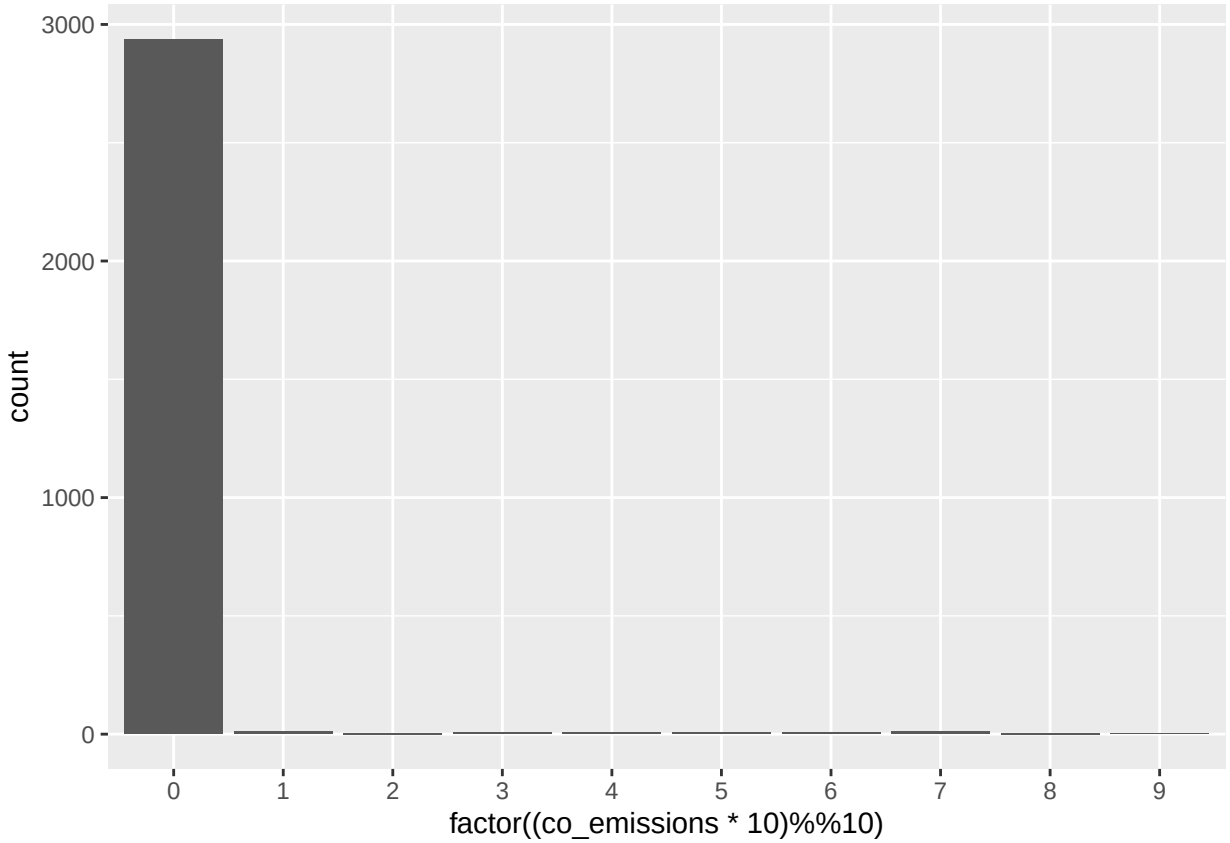

```
ggplot(subset_cars,aes(factor((combined_metric*10)%%10)))+ geom_bar()
```



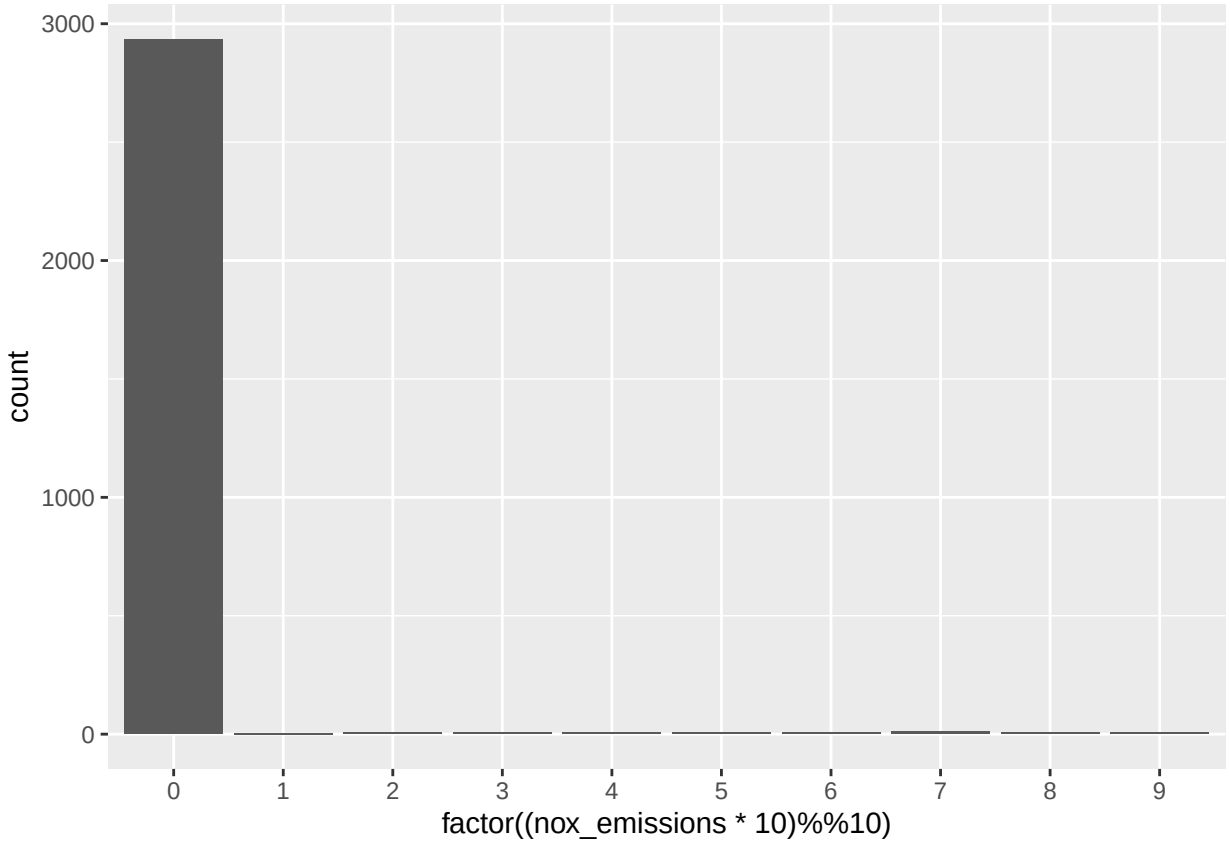

```
# weird rounding:
ggplot(subset_cars,aes(factor((noise_level*10)%%10)))+ geom_bar()
```



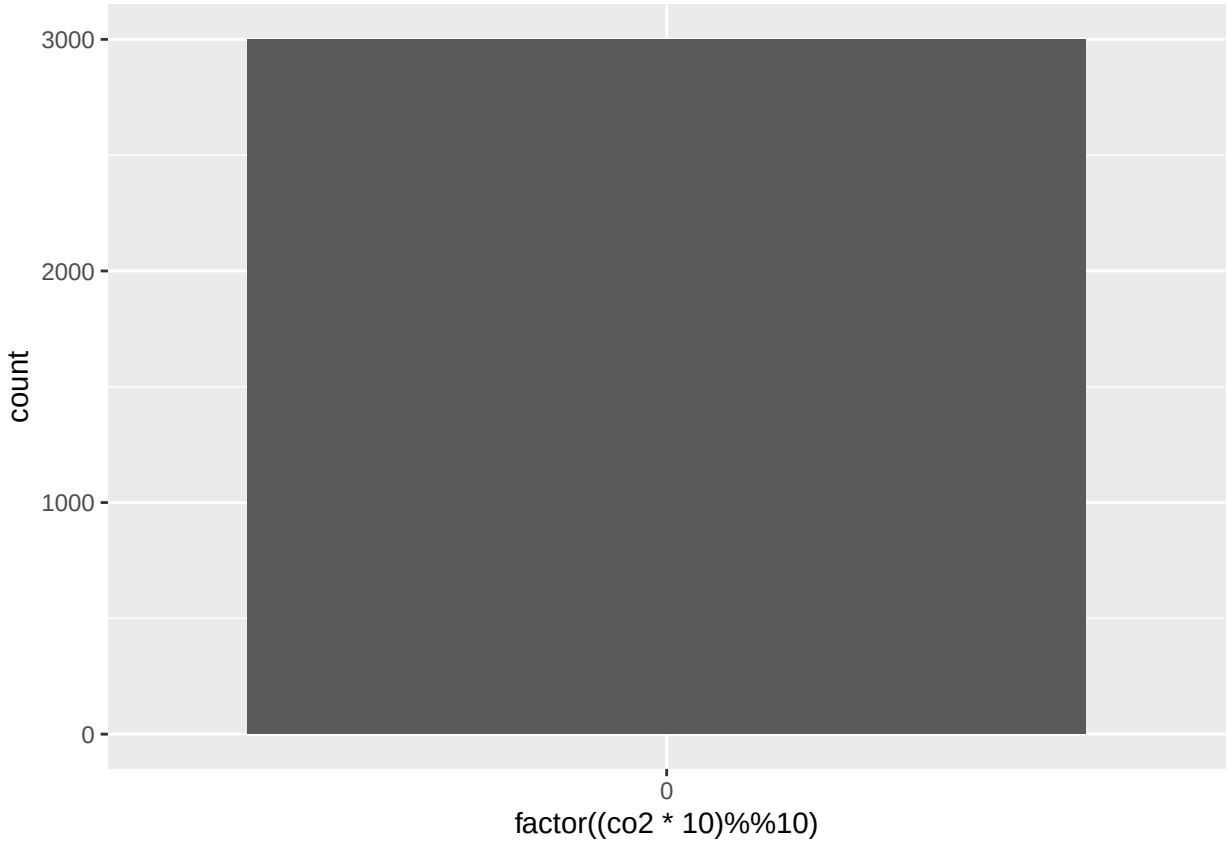

```
ggplot(subset_cars,aes(factor((co_emissions*10)%%10)))+ geom_bar()
```




```
ggplot(subset_cars,aes(factor((nox_emissions*10)%%10)))+ geom_bar()
```




```
# normal rounding:
ggplot(subset_cars,aes(factor((co2*10)%%10)))+ geom_bar()
```




It would be interesting to see if rounding can be predicted or is useful.

### 1.1 checking for duplicates

```
#View(subset_cars[duplicated(subset_cars),])
print(sum(duplicated(subset_cars)))
```

```
## [1] 66
```

```
print(sum(duplicated(subset_cars["model"])))
```

```
## [1] 1730
```

```
print(sum(duplicated(subset_cars[c("urban_metric", "noise_level", "co2")])))
```

```
## [1] 598
```

OK some duplicates. In theory it shouldn't be impossible ... but IDK if this was intended when gathering the data ... A more in depth study of duplicates could be done using clustering techniques.

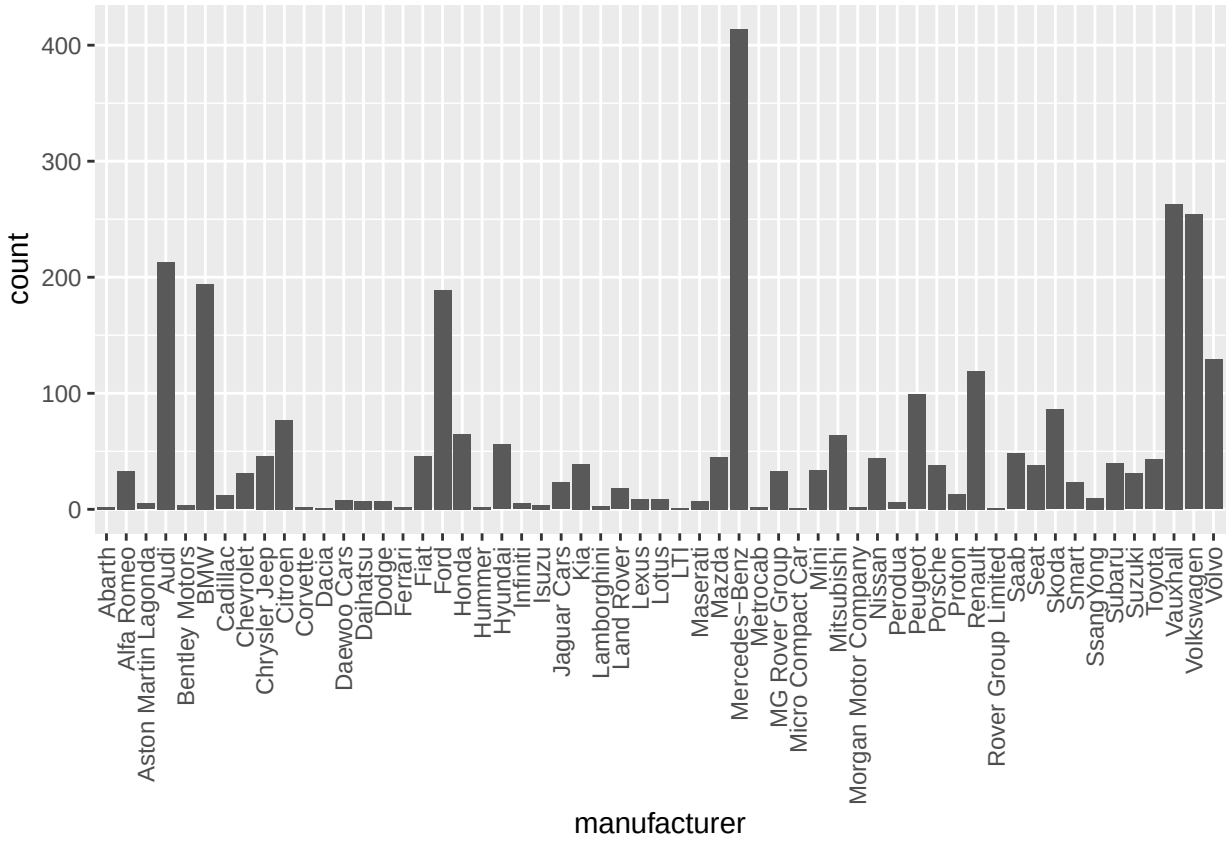
### 1.2 are there missing values?

```
sum(is.na(subset_cars))
```

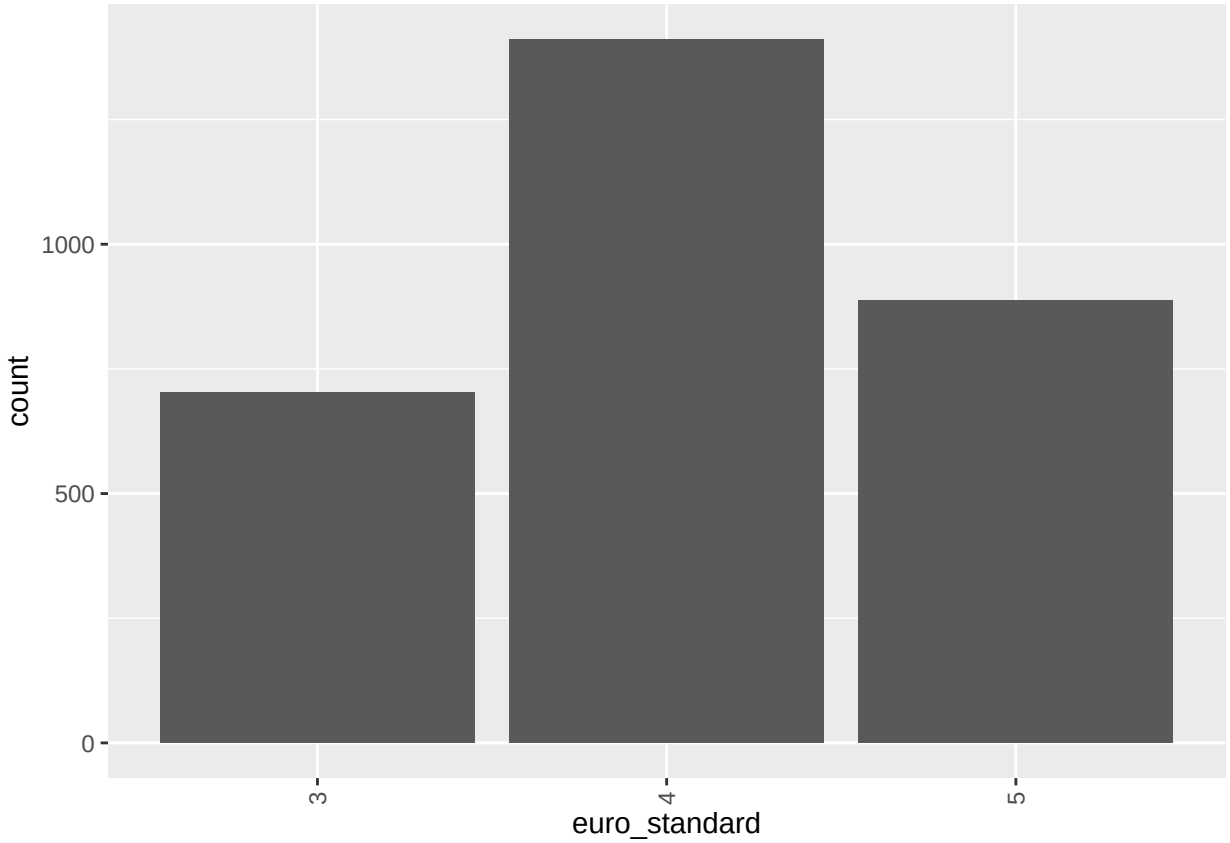
```
## [1] 0
```

### 1.3 univariate plots for factor types

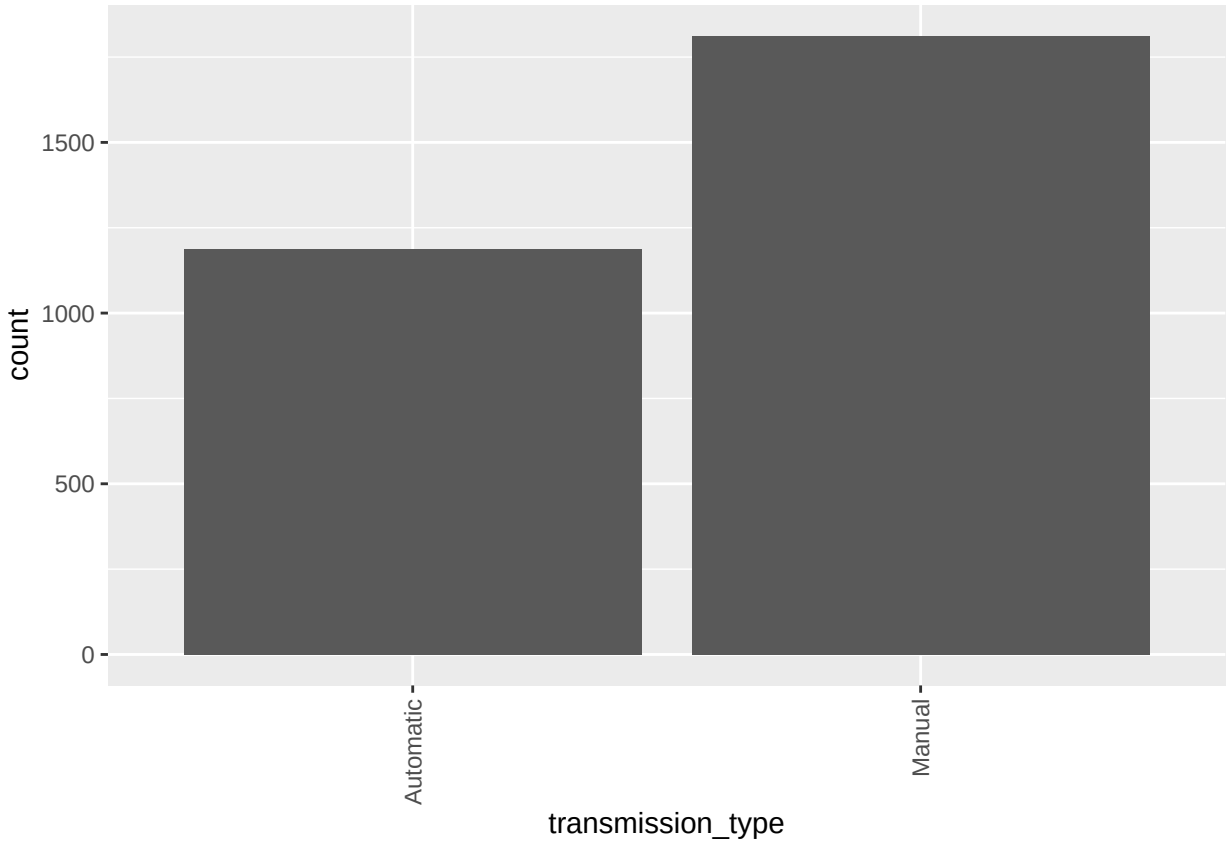
```
# (for future me) for loop doesn't work and dont plot model
ggplot(subset_cars ,aes(manufacturer)) + geom_bar(stat ="count")+
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```



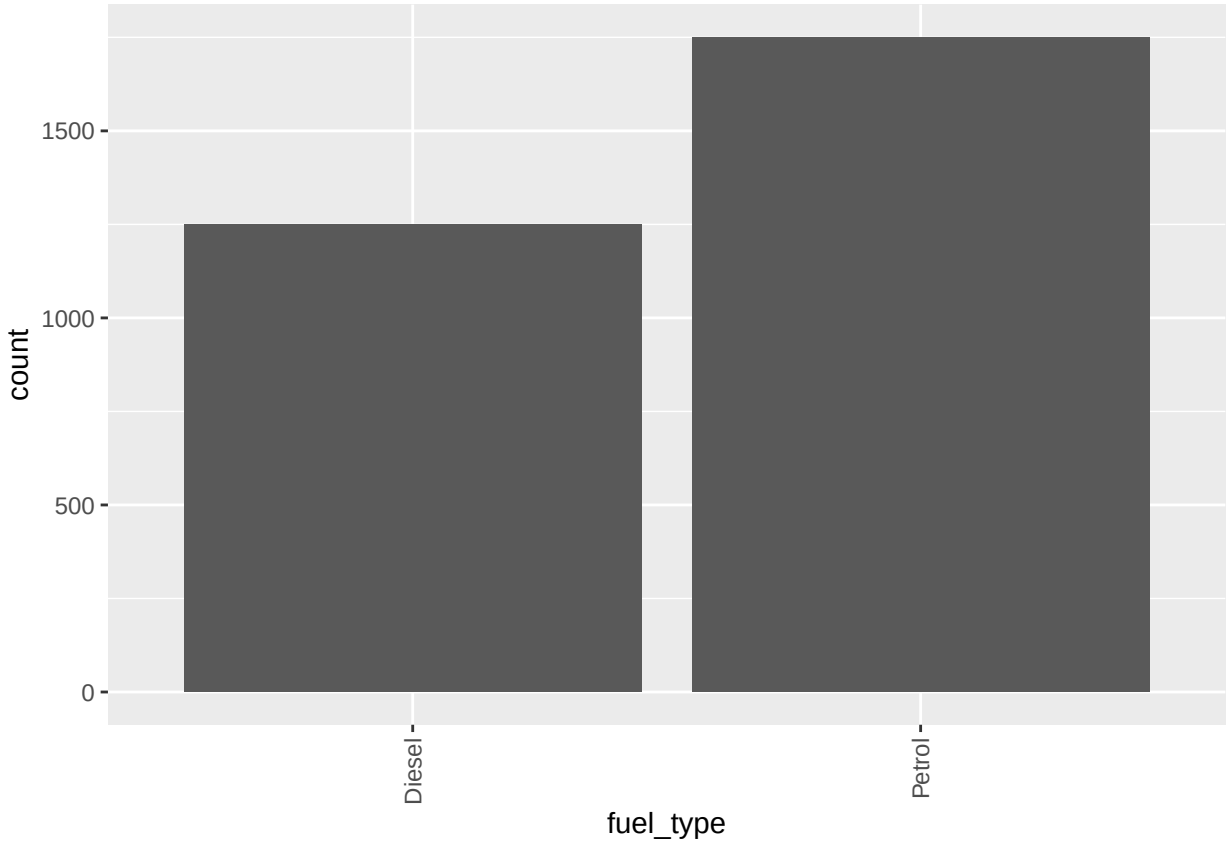

```
ggplot(subset_cars ,aes(euro_standard)) + geom_bar(stat ="count")+
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```




```
ggplot(subset_cars ,aes(transmission_type)) + geom_bar(stat ="count")+
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```




```
ggplot(subset_cars ,aes(fuel_type)) + geom_bar(stat ="count")+
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```

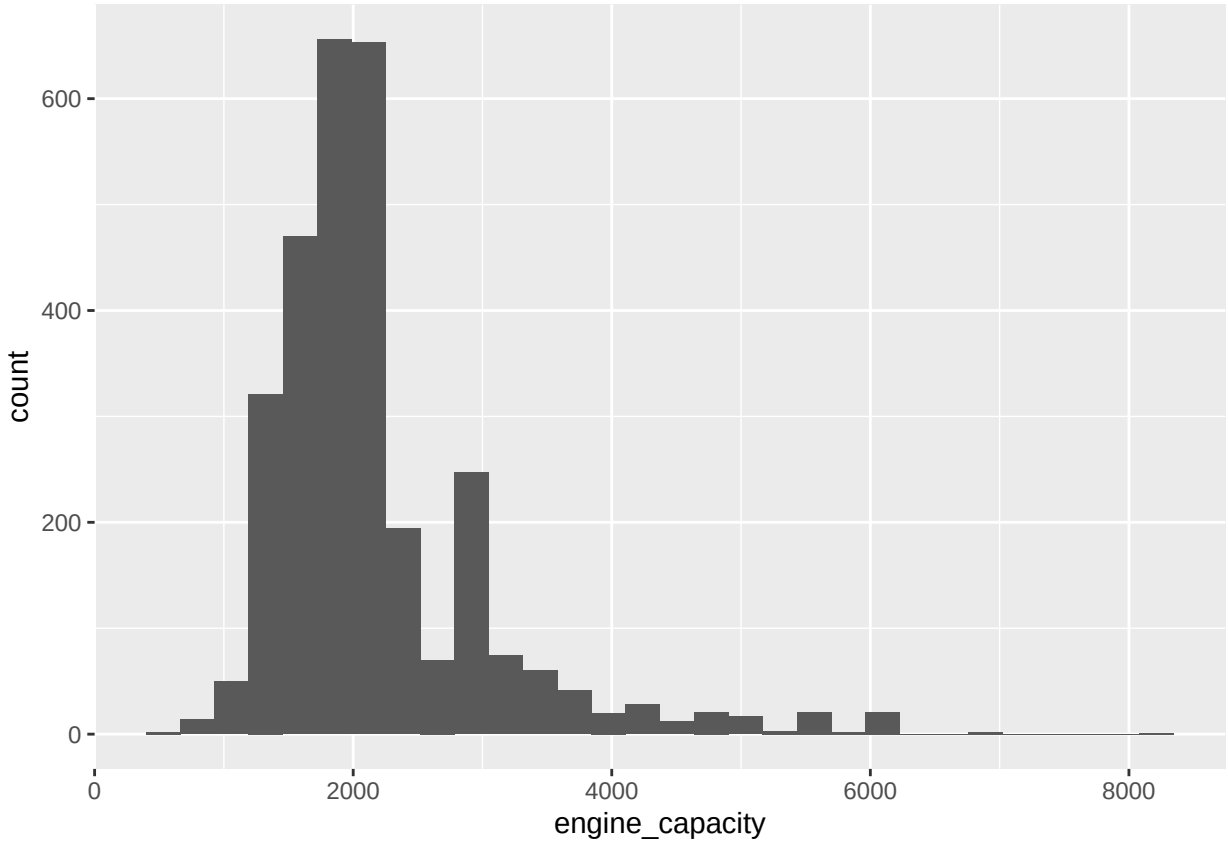



Everything looks balanced minus the manufacturer. From our humble car knowledge we think that there are major differences between cars from different manufacturers (you shouldn't compare a Lamborghini with a mini it just has other stats). This probably gives a problem with balancing (whatever we mean). We are not sure that the counts represents the amount of cars in Europe? Later we'll look into it.

### 1.4 plots for numerical types

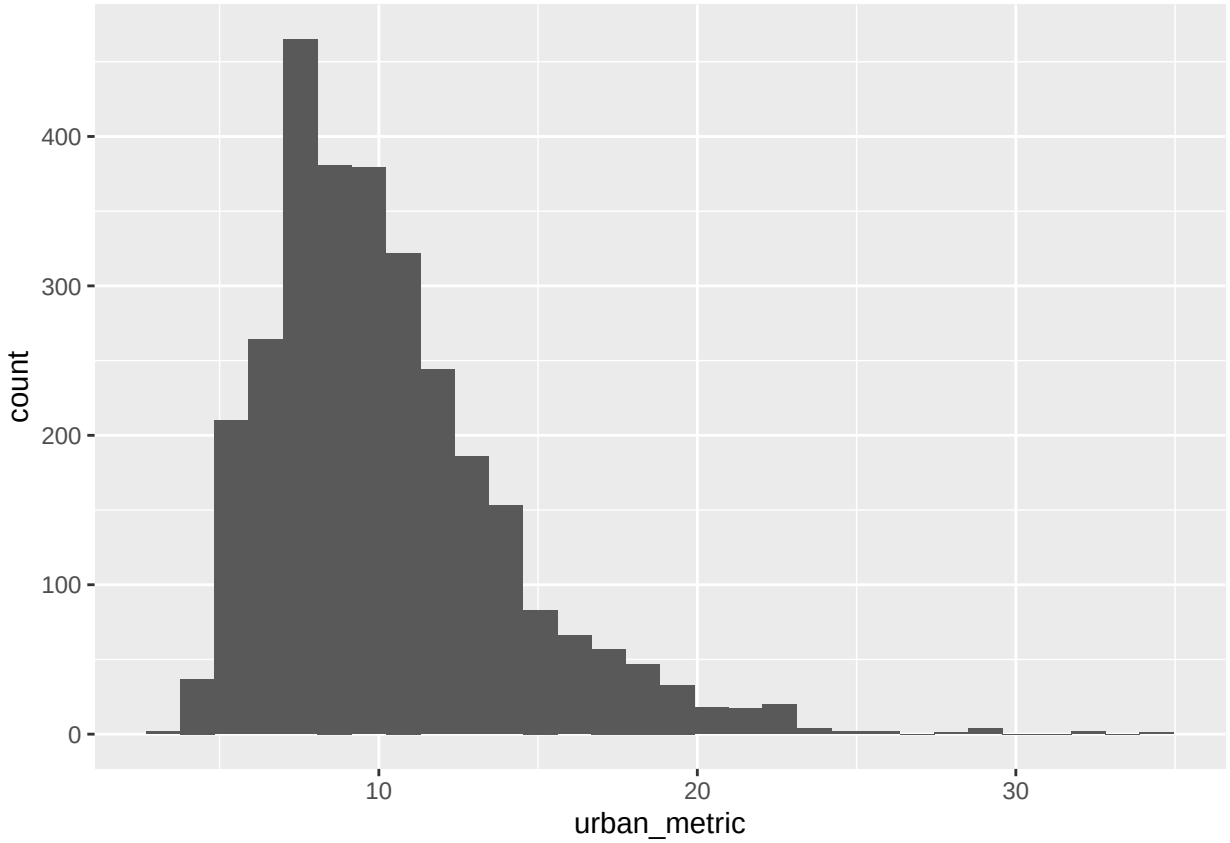
```
ggplot(subset_cars,aes(engine_capacity))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



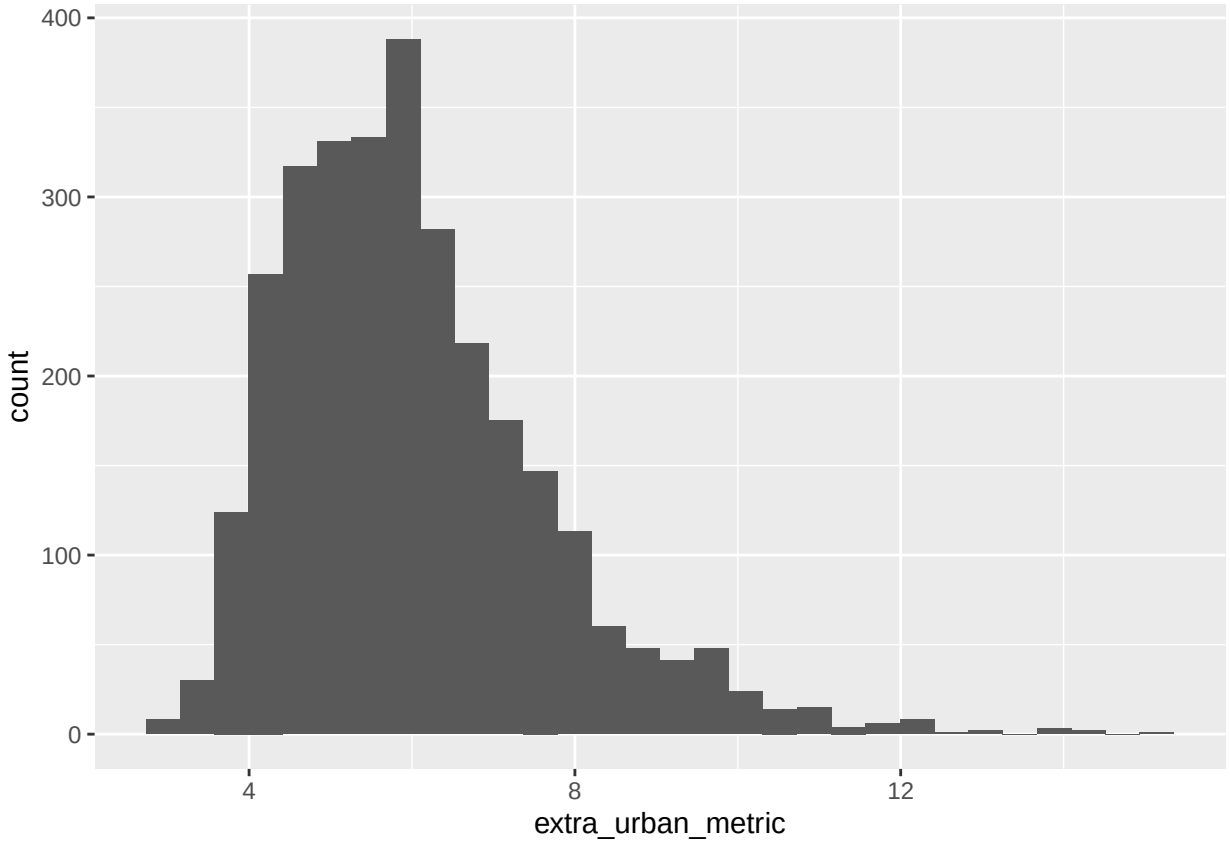

```
ggplot(subset_cars,aes(urban_metric))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



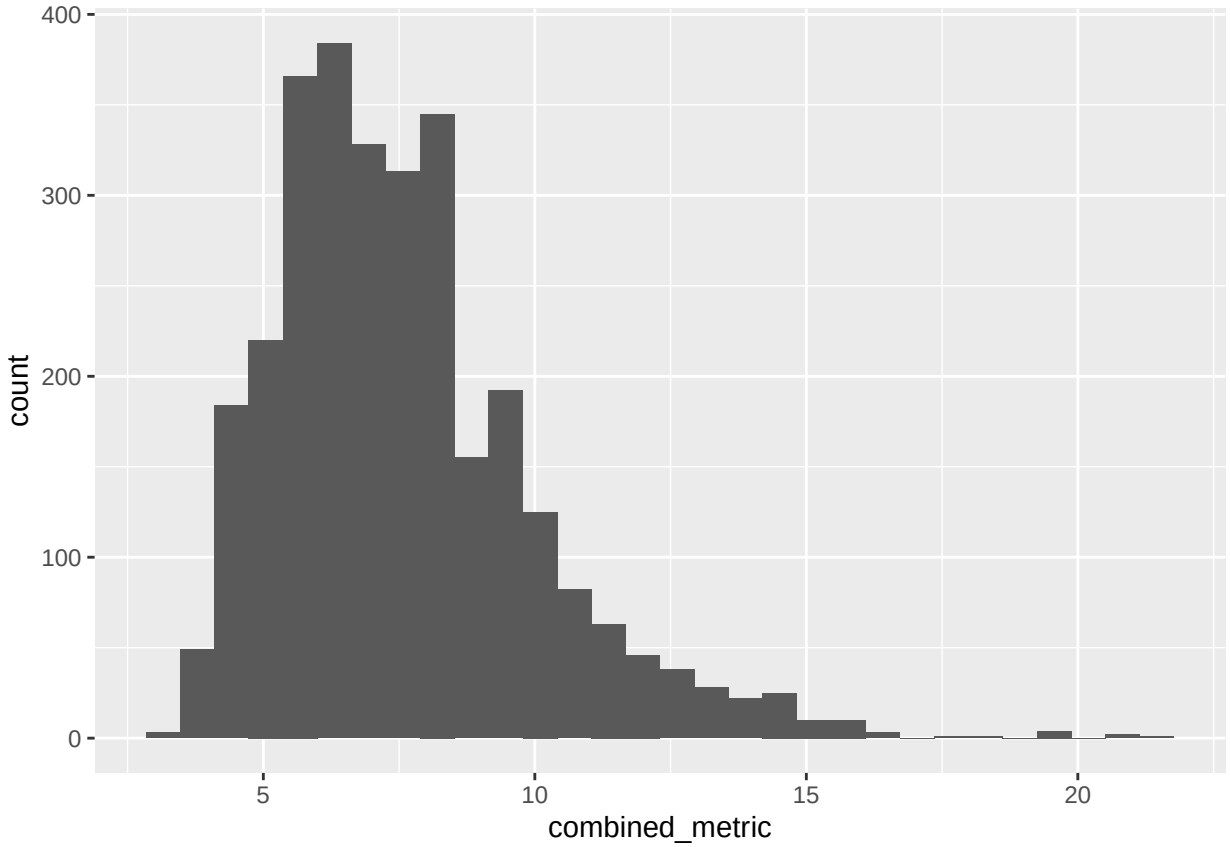

```
ggplot(subset_cars,aes(extra_urban_metric))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

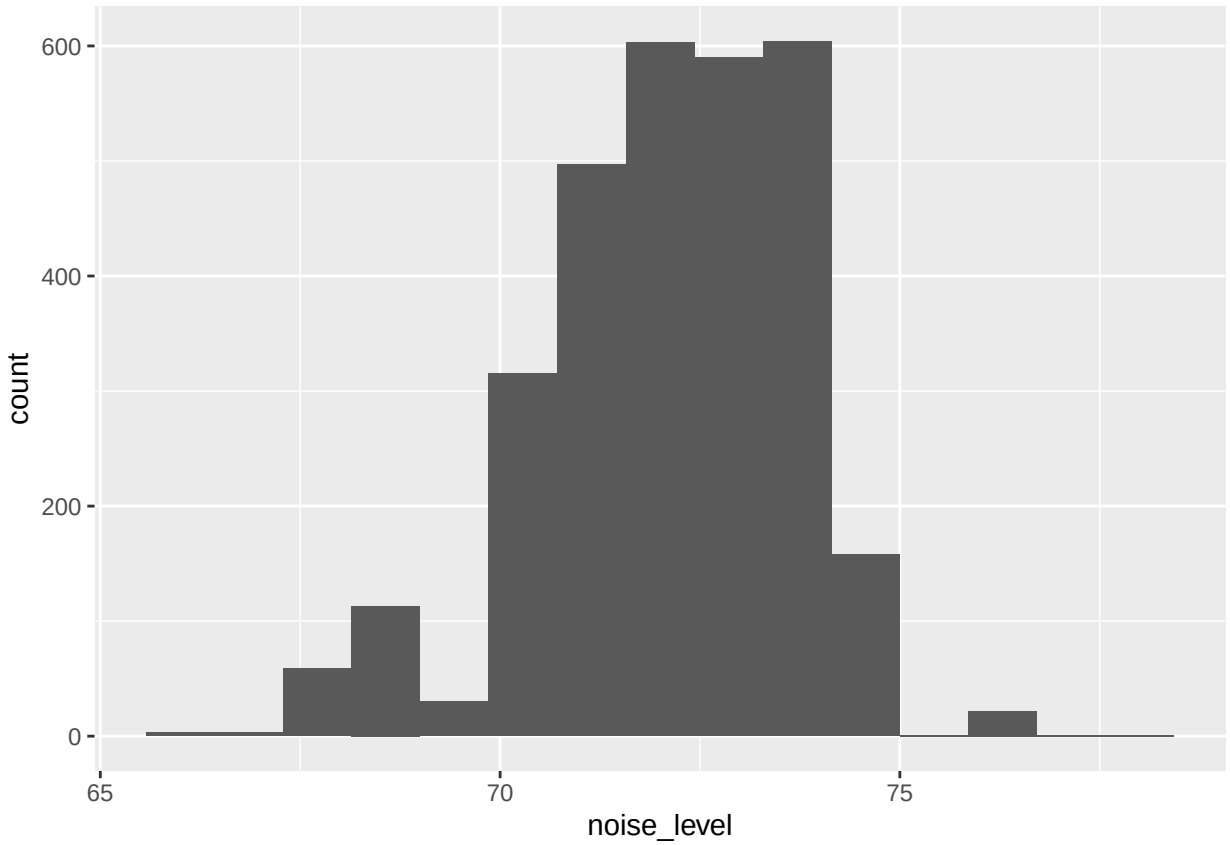



```
ggplot(subset_cars,aes(combined_metric))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

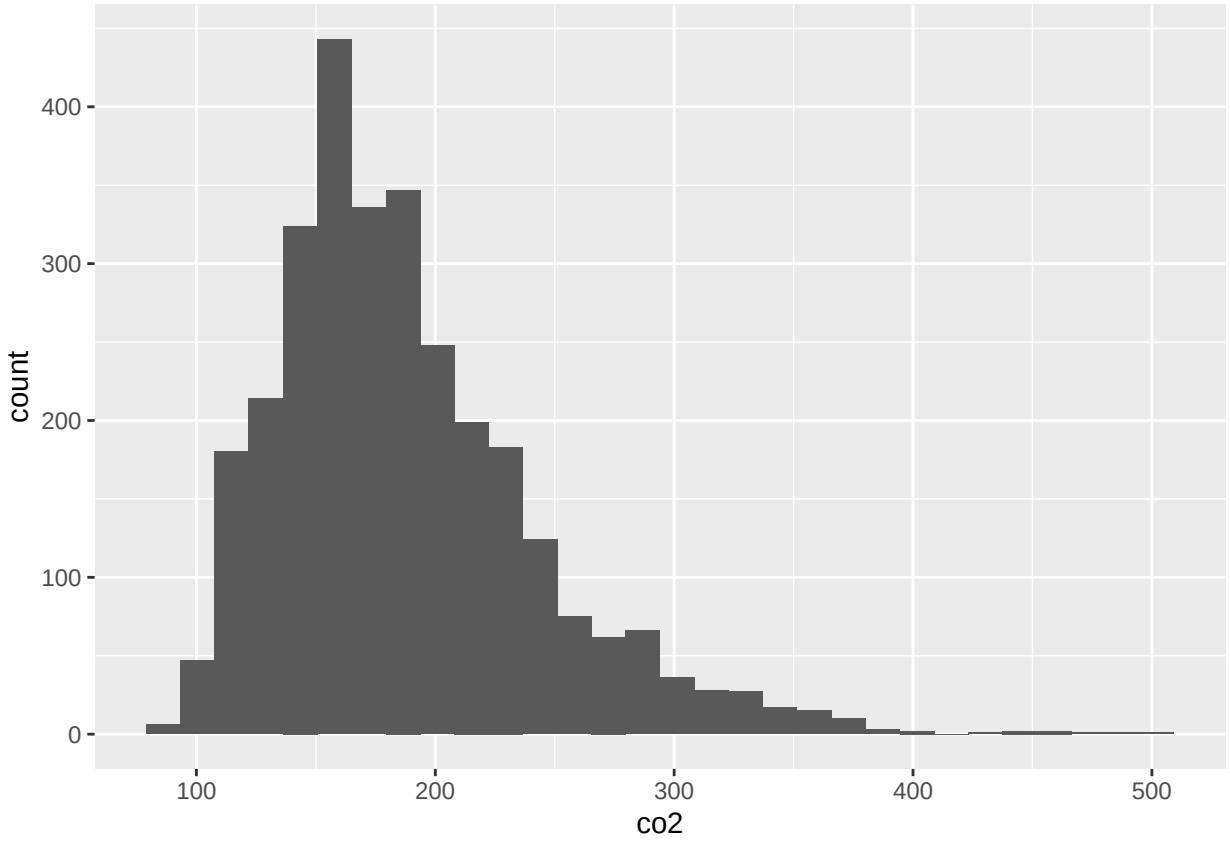



```
selectNormalNoise = (60<subset_cars$noise_level) & (subset_cars$noise_level < 80)
ggplot(subset_cars[selectNormalNoise,],aes(noise_level))+ geom_histogram(bins = 15)
```



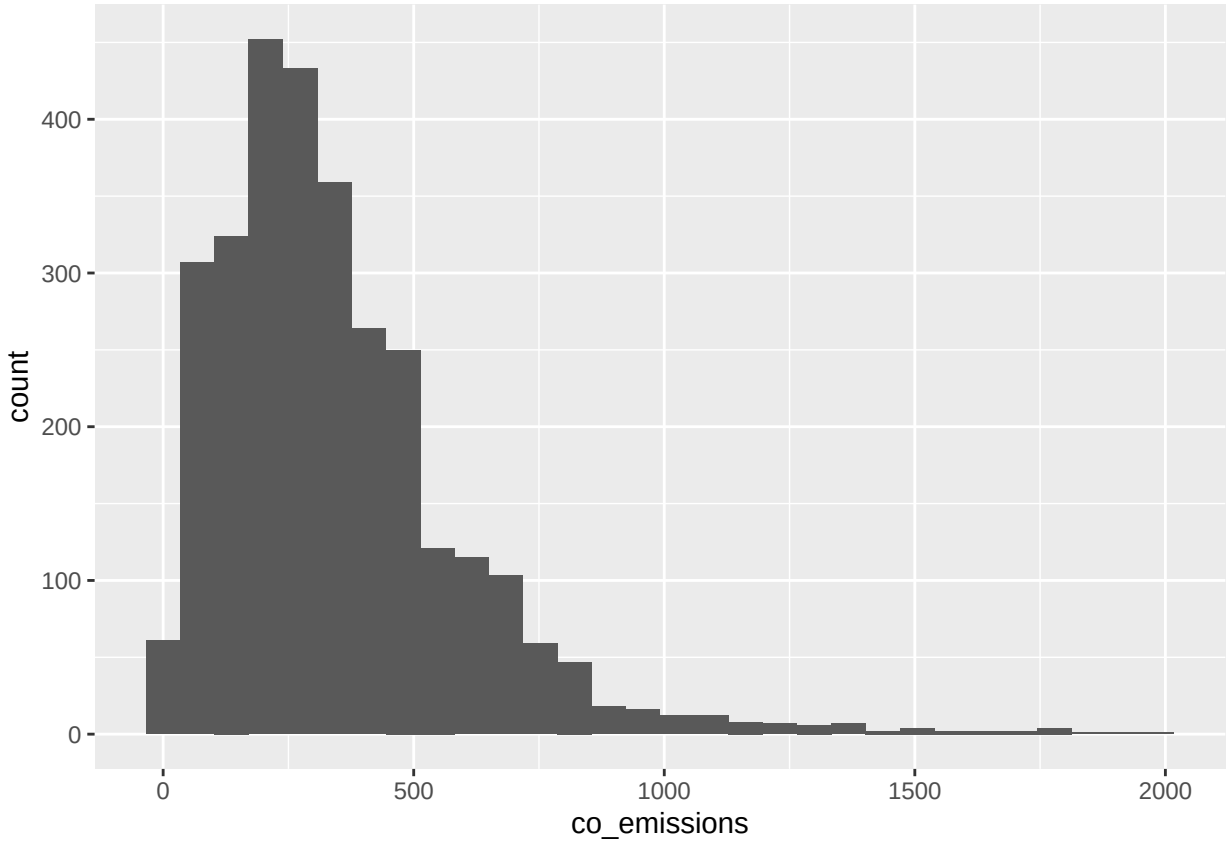

```
ggplot(subset_cars,aes(co2))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



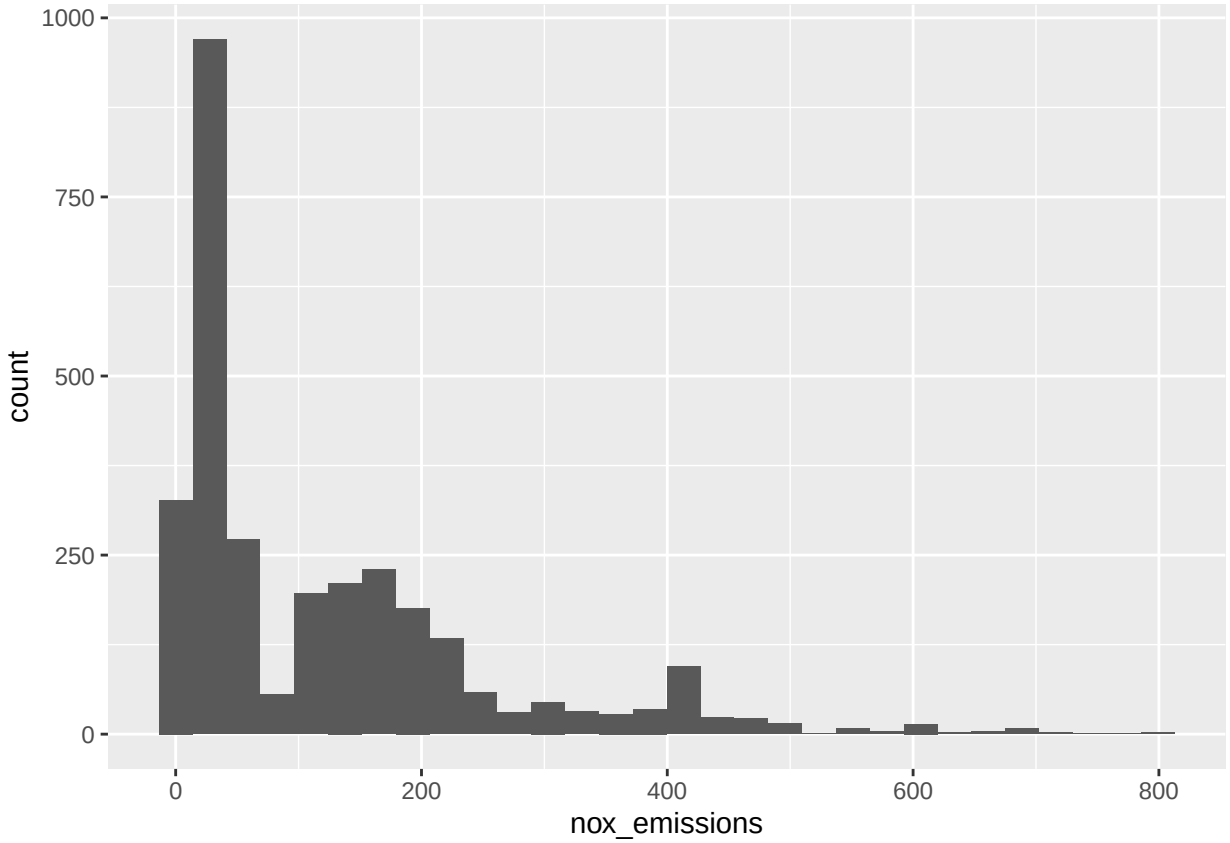

```
ggplot(subset_cars,aes(co_emissions))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
selectNormalNox = (subset_cars$nox_emissions < 1000)
ggplot(subset_cars[selectNormalNox, ],aes(nox_emissions))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



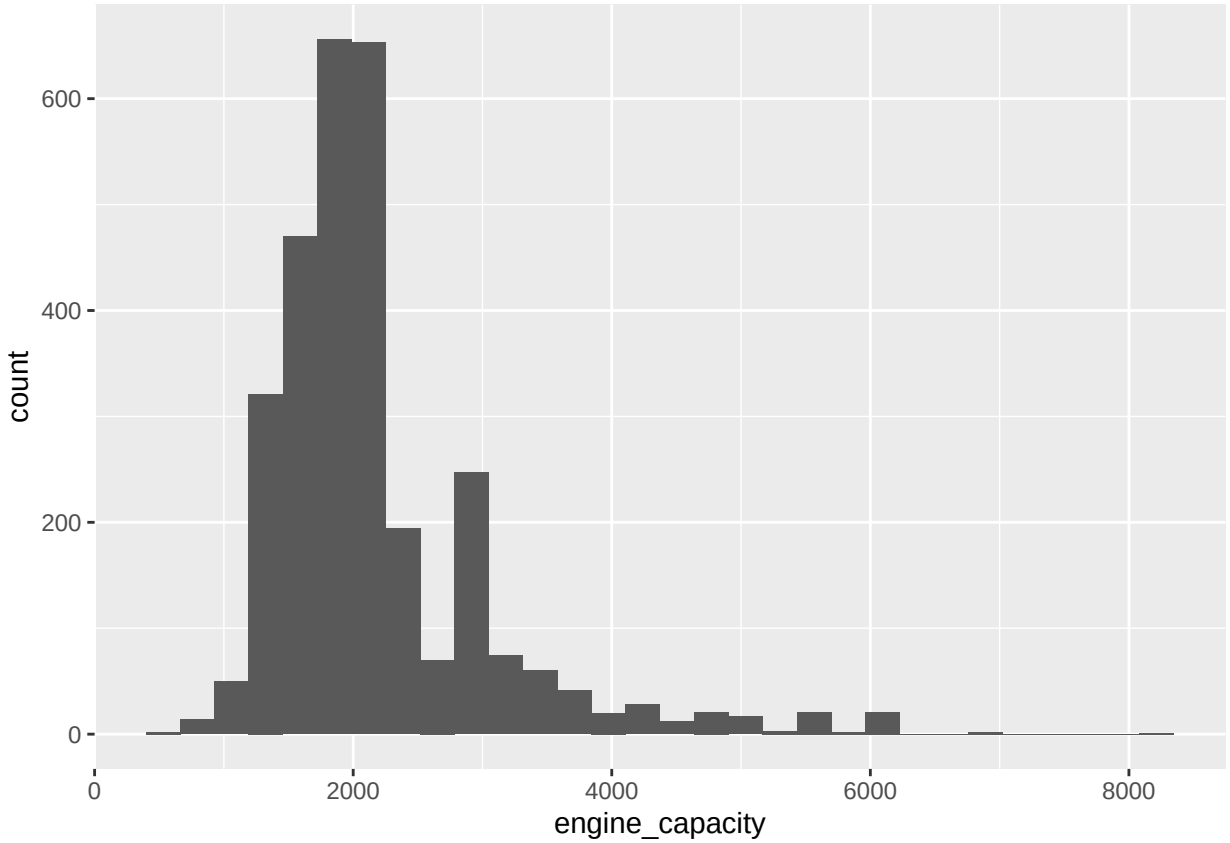

Everything looks skewed to the right minus the noise emissions (maybe this is because it is on a log scale?). I'll try transforming the noise. Then some outliers (points that ruined my plots so we filterd them out).

```
selectBadRows <- !selectNormalNoise | !selectNormalNox
selectNormalRows <- !selectBadRows
#why doesn't this work? OK used cars_data instead of subset ...
subset.bad <- subset_cars[selectBadRows, ]
subset.trans <- subset_cars[selectNormalRows, ]
subset.trans$noise_level <- 1.26^(subset.trans$noise_level-70)
```

```
ggplot(subset.trans,aes(engine_capacity))+ geom_histogram()
```

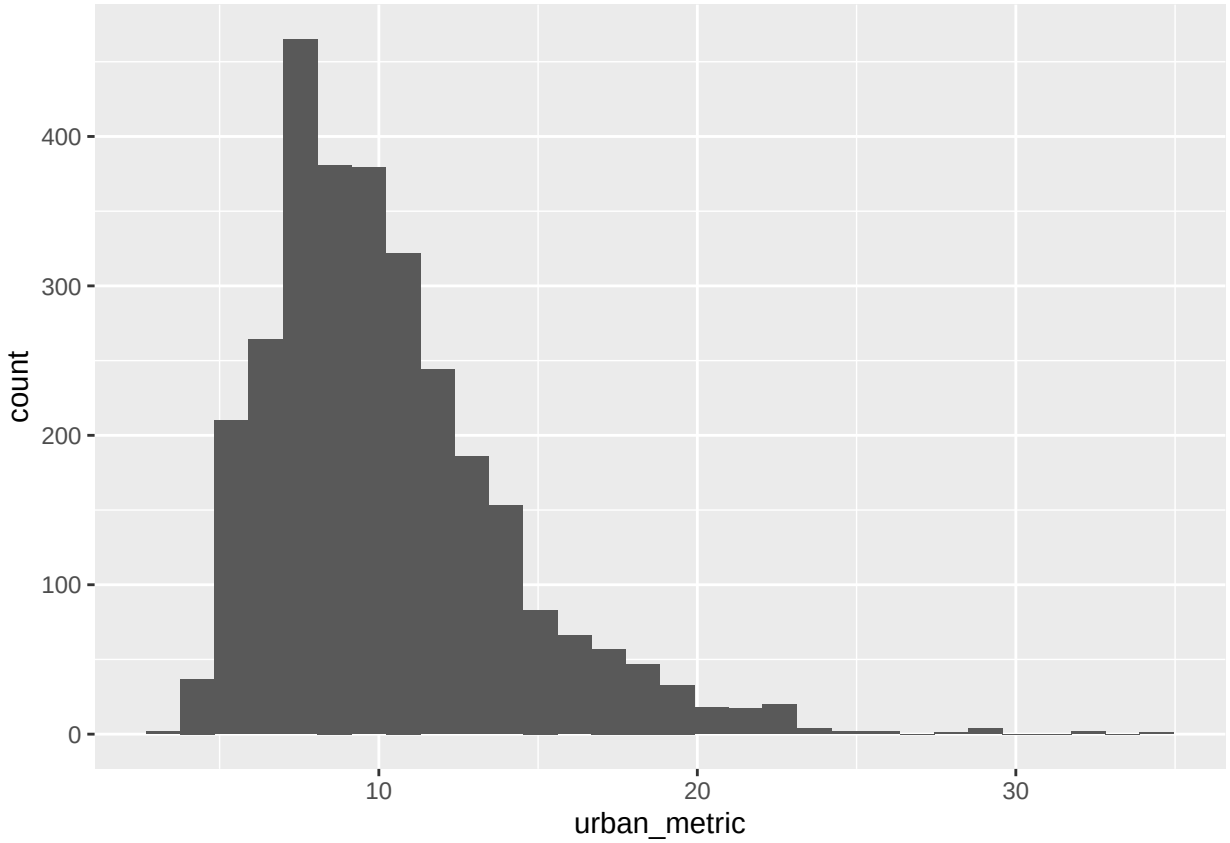
```
##
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



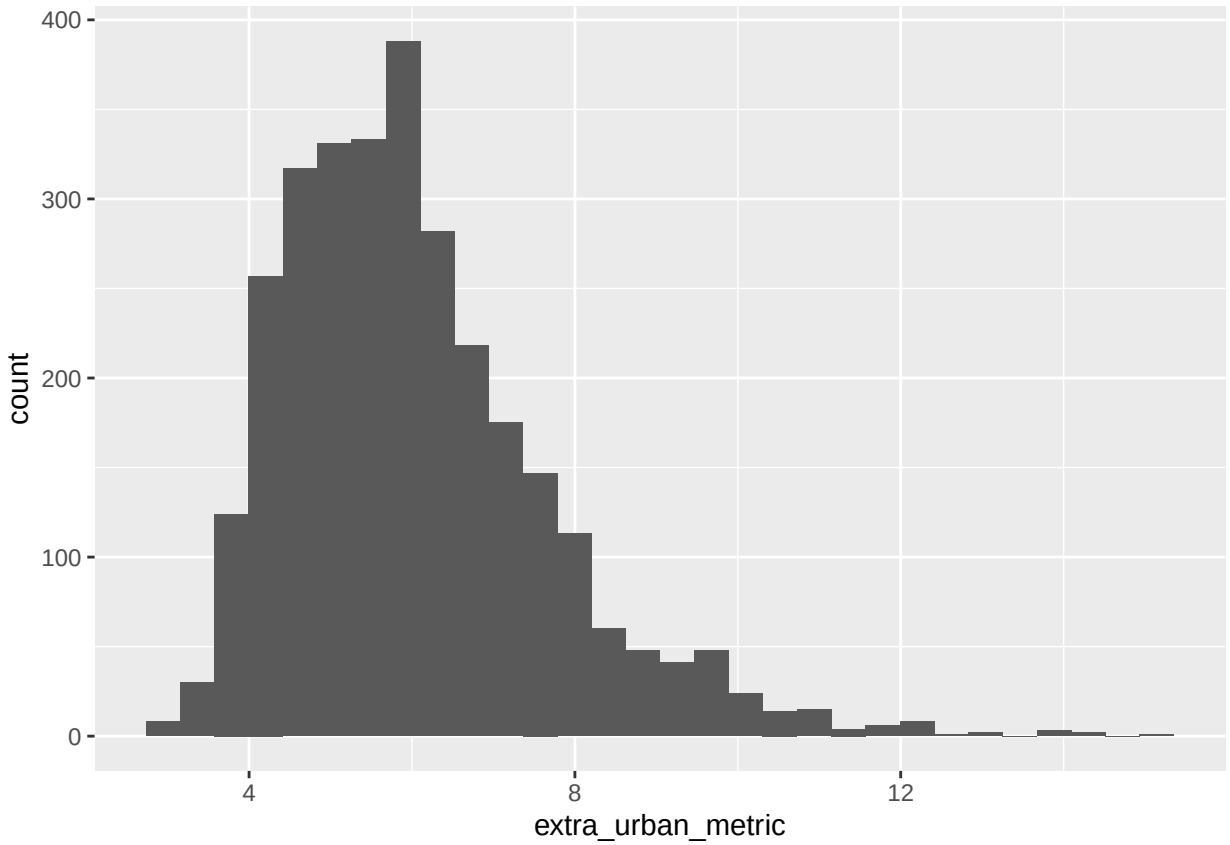

```
ggplot(subset.trans,aes(urban_metric))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



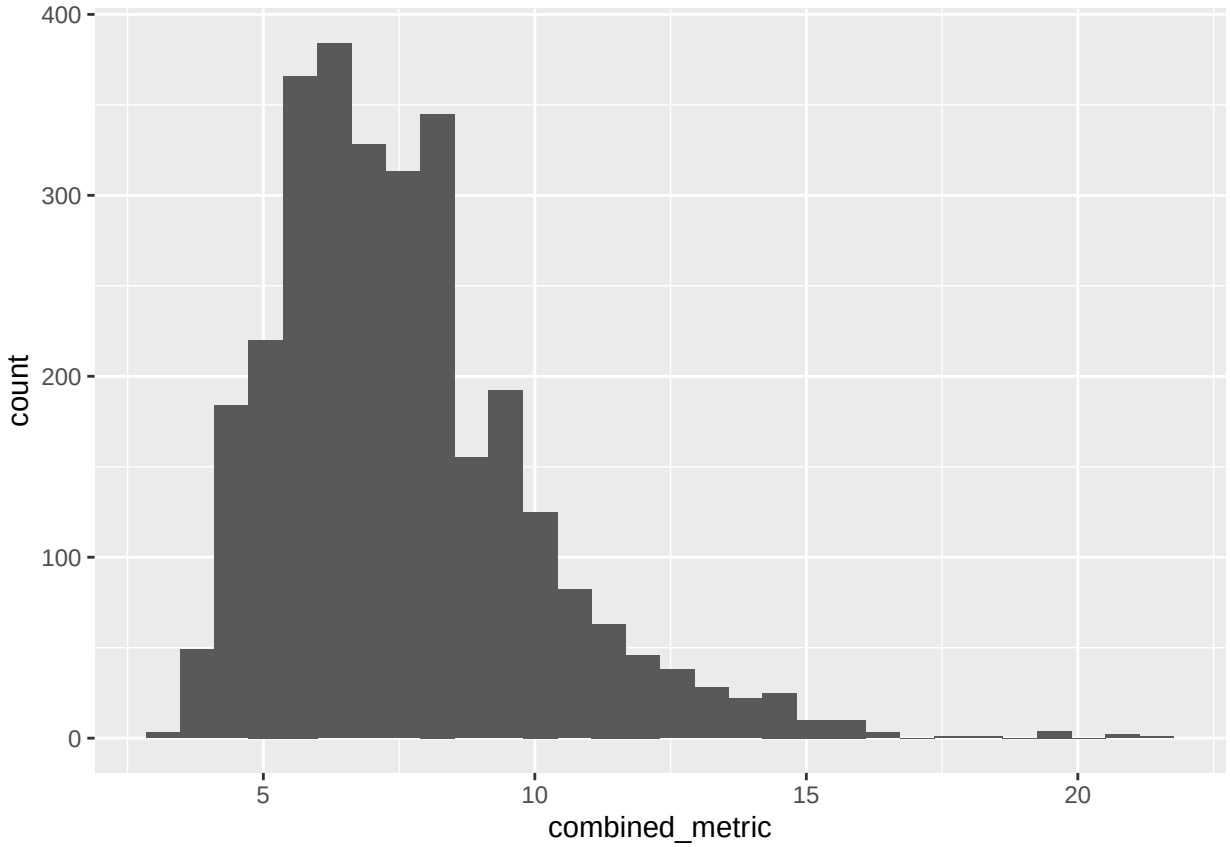

```
ggplot(subset.trans,aes(extra_urban_metric))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

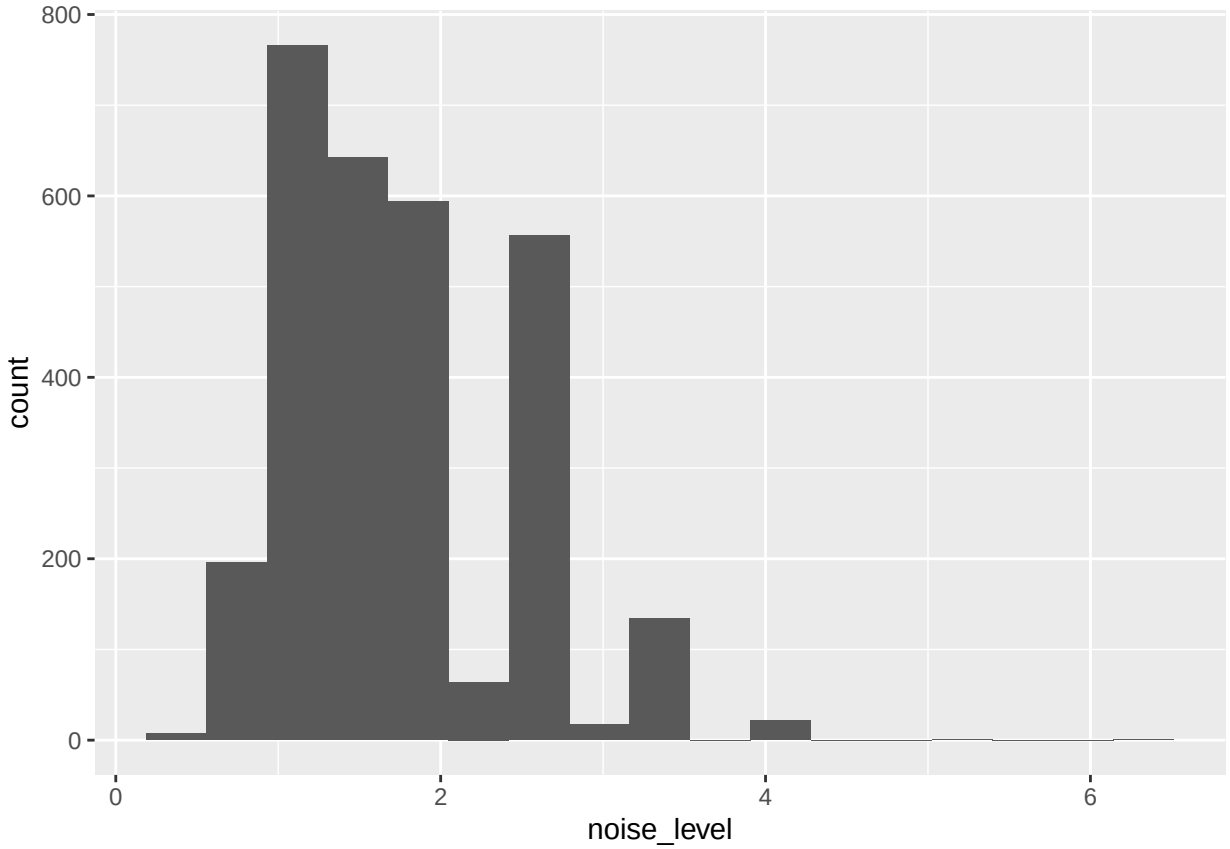



```
ggplot(subset.trans,aes(combined_metric))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

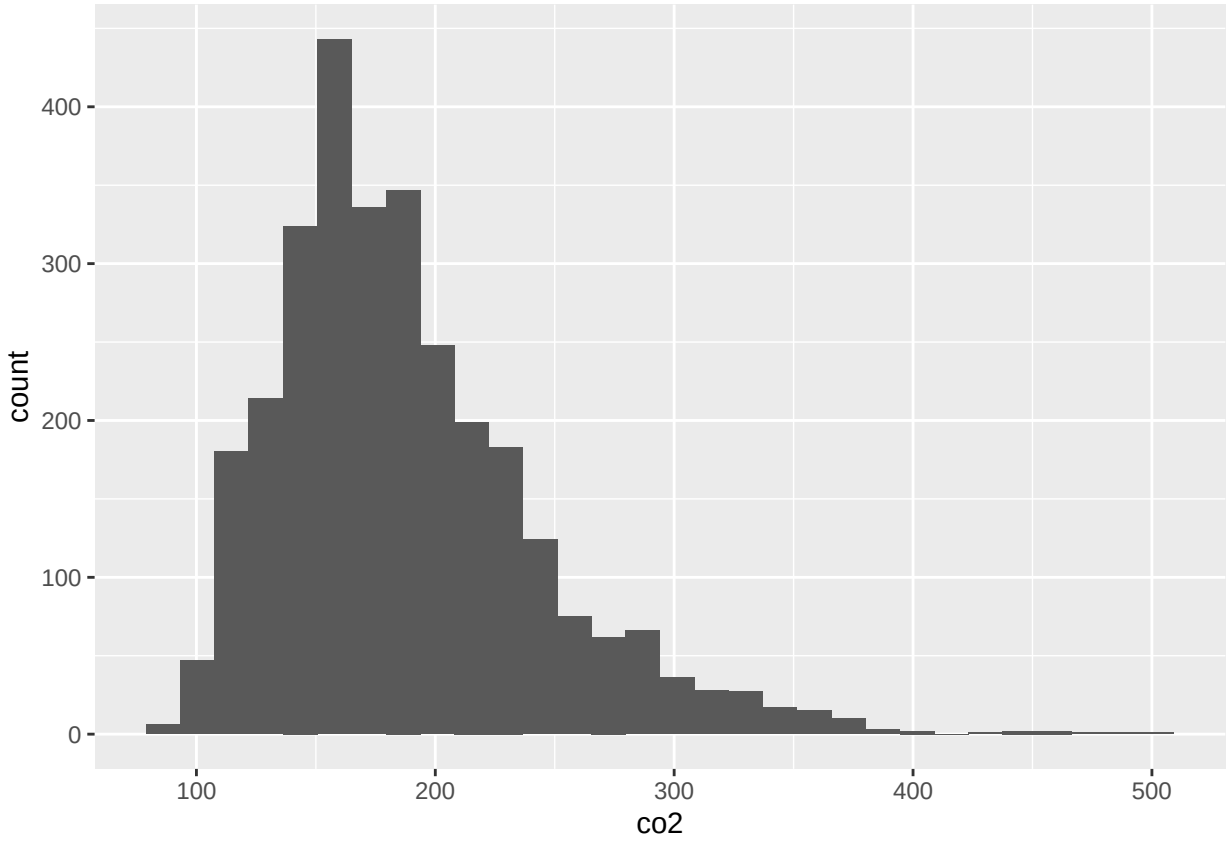



```
ggplot(subset.trans,aes(noise_level))+ geom_histogram(bins = 17)
```



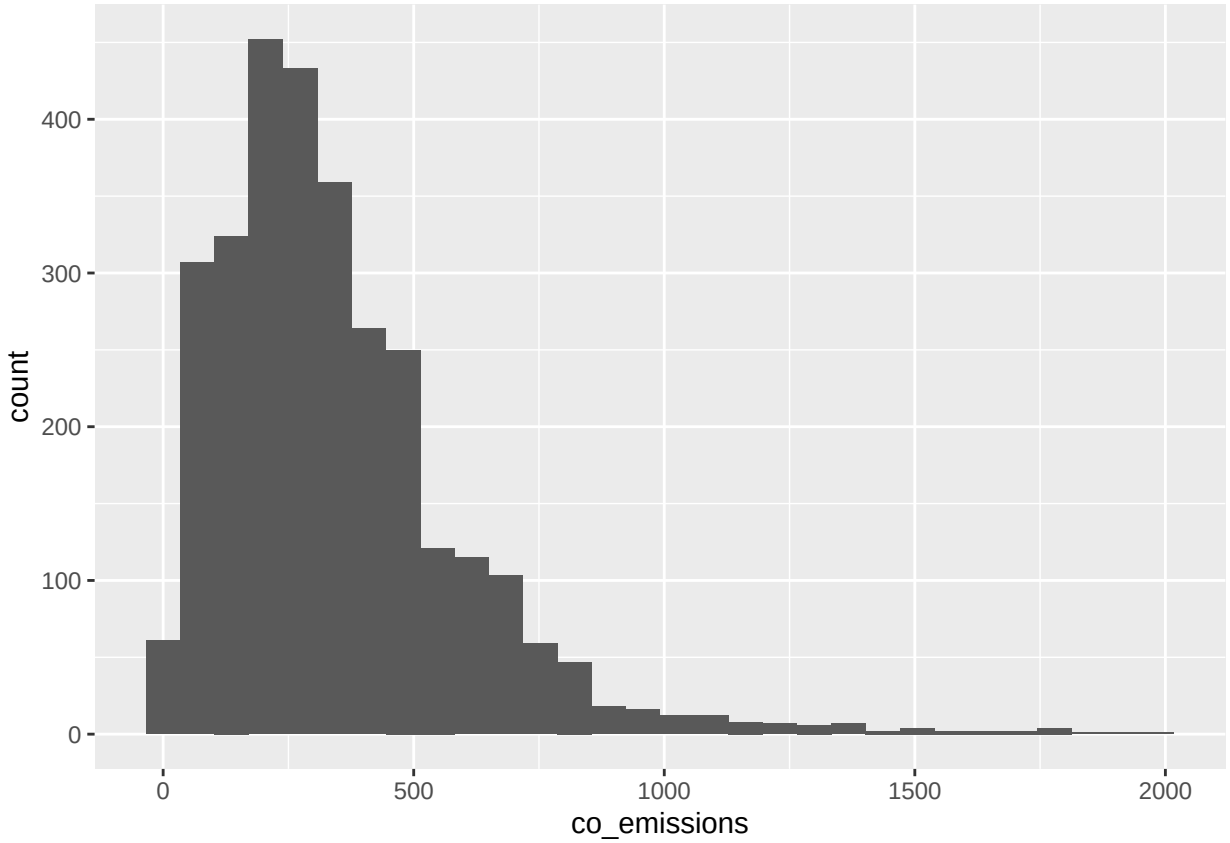

```
ggplot(subset.trans,aes(co2))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



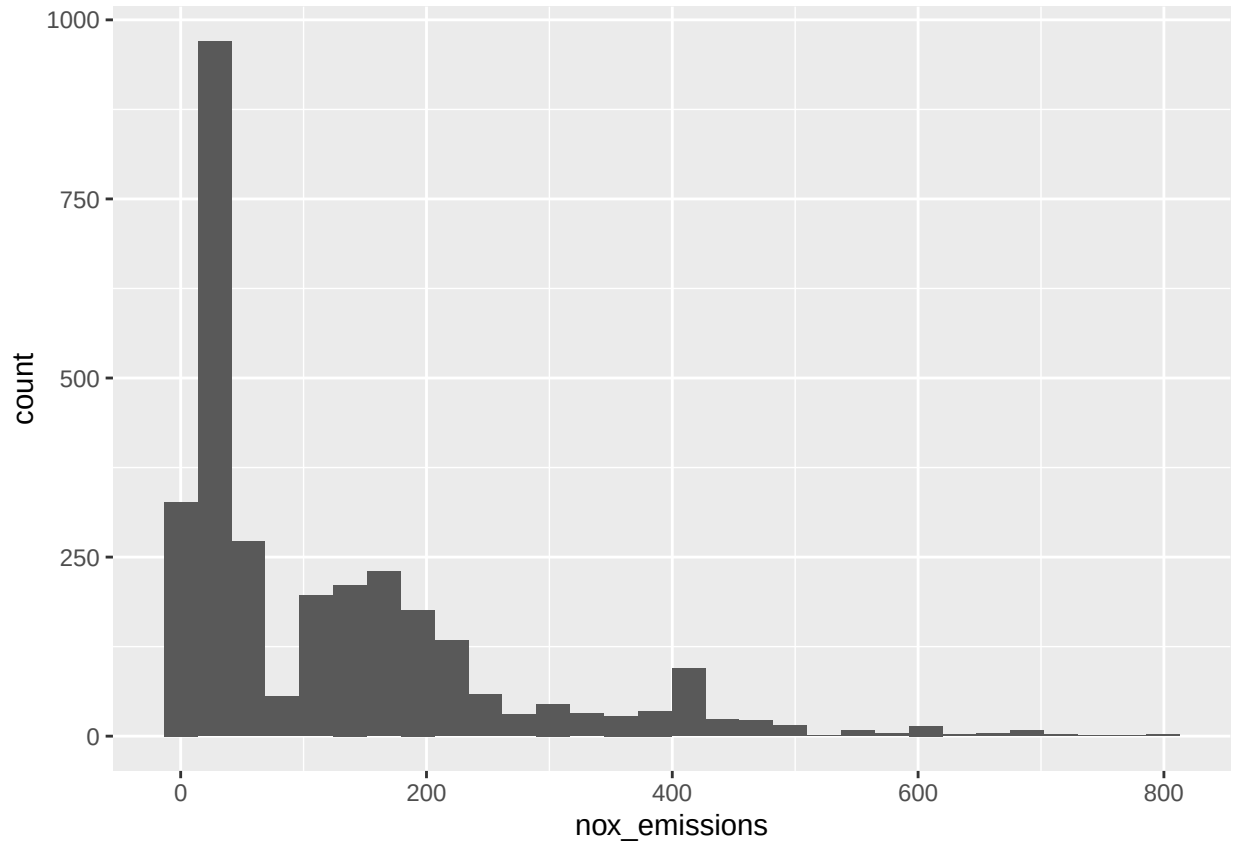

```
ggplot(subset.trans,aes(co_emissions))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(subset.trans,aes(nox_emissions))+ geom_histogram()
```

```
##
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

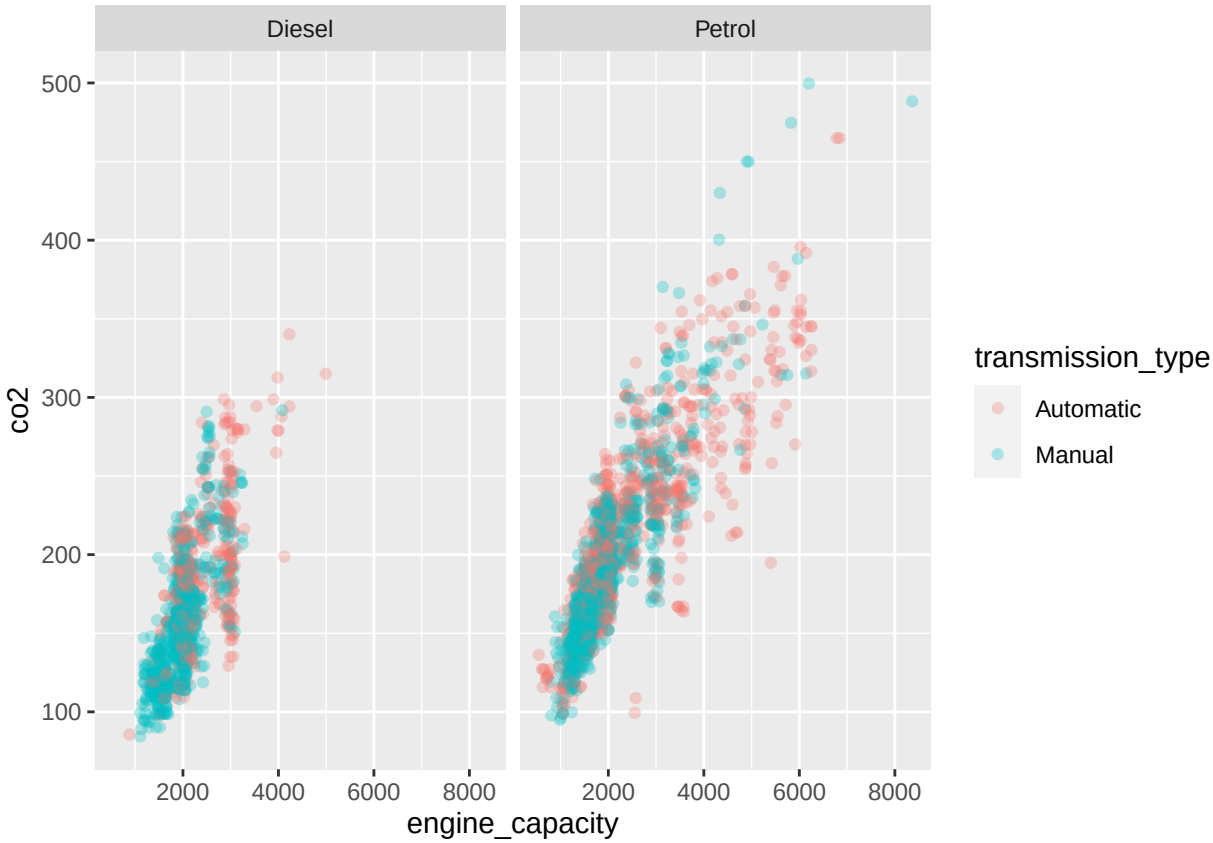



```
subset.trans[order(subset.trans$noise_level,decreasing = TRUE), c("manufacturer","noise_level" )][1:20,
```

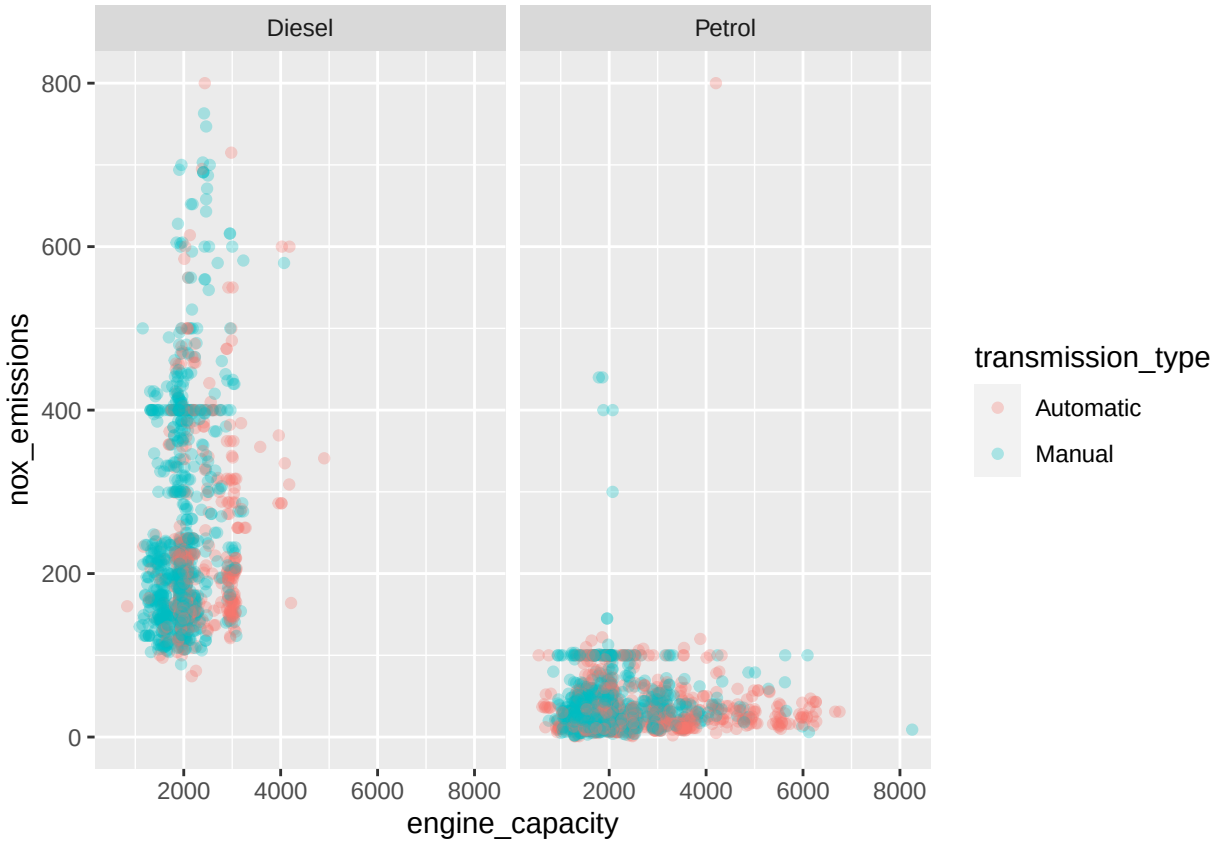
```
##        manufacturer noise_level
## 29213     Chevrolet    6.352788
## 14299       Porsche    5.041895
## 8863     Mitsubishi    4.001504
## 14112    Mitsubishi    4.001504
## 20108    Volkswagen    4.001504
## 21067    Mitsubishi    4.001504
## 36289 Mercedes-Benz    4.001504
## 12507           BMW    4.001504
## 26962    Mitsubishi    4.001504
## 17468 Chrysler Jeep    4.001504
## 18031       Porsche    4.001504
## 35842         Isuzu    4.001504
## 33068    Land Rover    4.001504
## 33650      Vauxhall    4.001504
## 16751       Porsche    4.001504
## 26537 Chrysler Jeep    4.001504
## 2930          Isuzu    4.001504
## 34674    Land Rover    4.001504
## 8409        Porsche    4.001504
## 22900    Mitsubishi    4.001504
```

### 1.5 plots for comparing variables

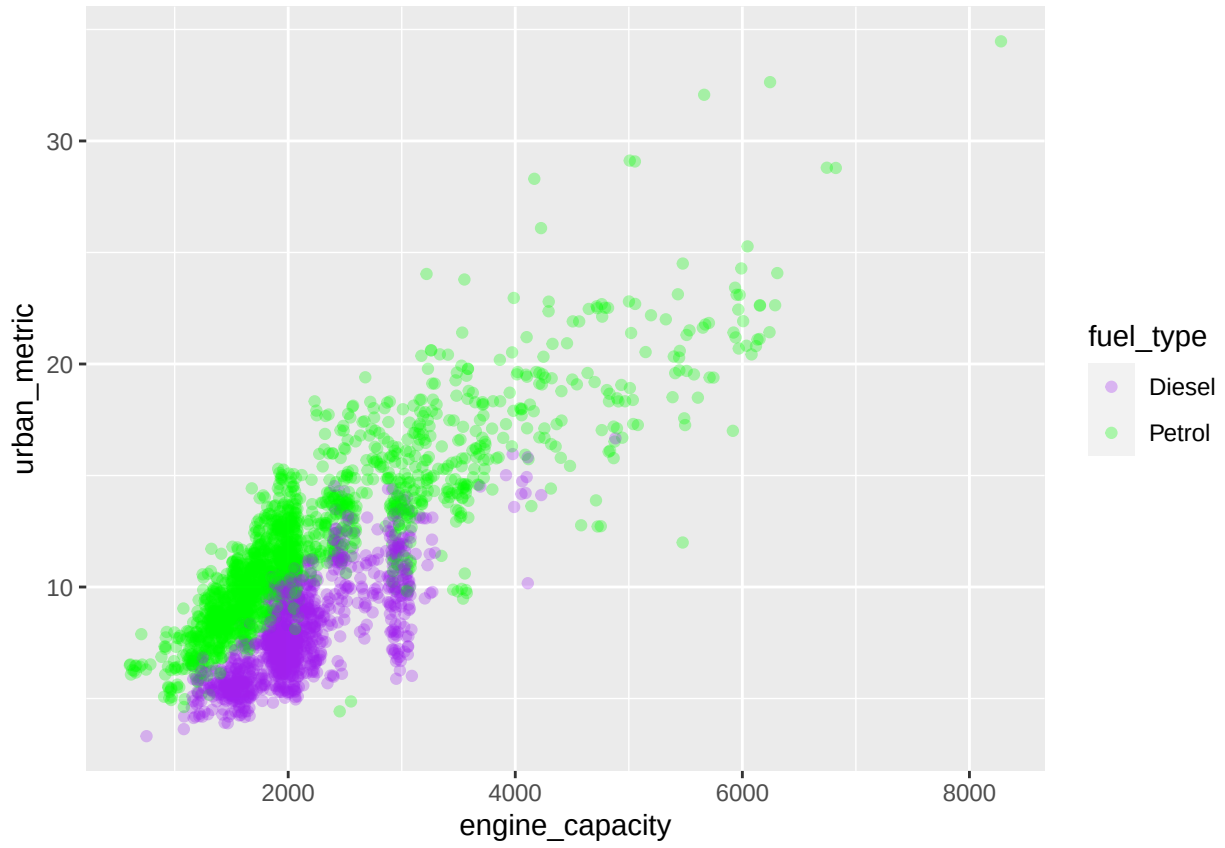
```
ggplot(subset.trans, aes(x = engine_capacity, y = co2, color = transmission_type)) +
  geom_jitter(alpha = 0.3, width = 100) +
  facet_grid(~fuel_type)
```



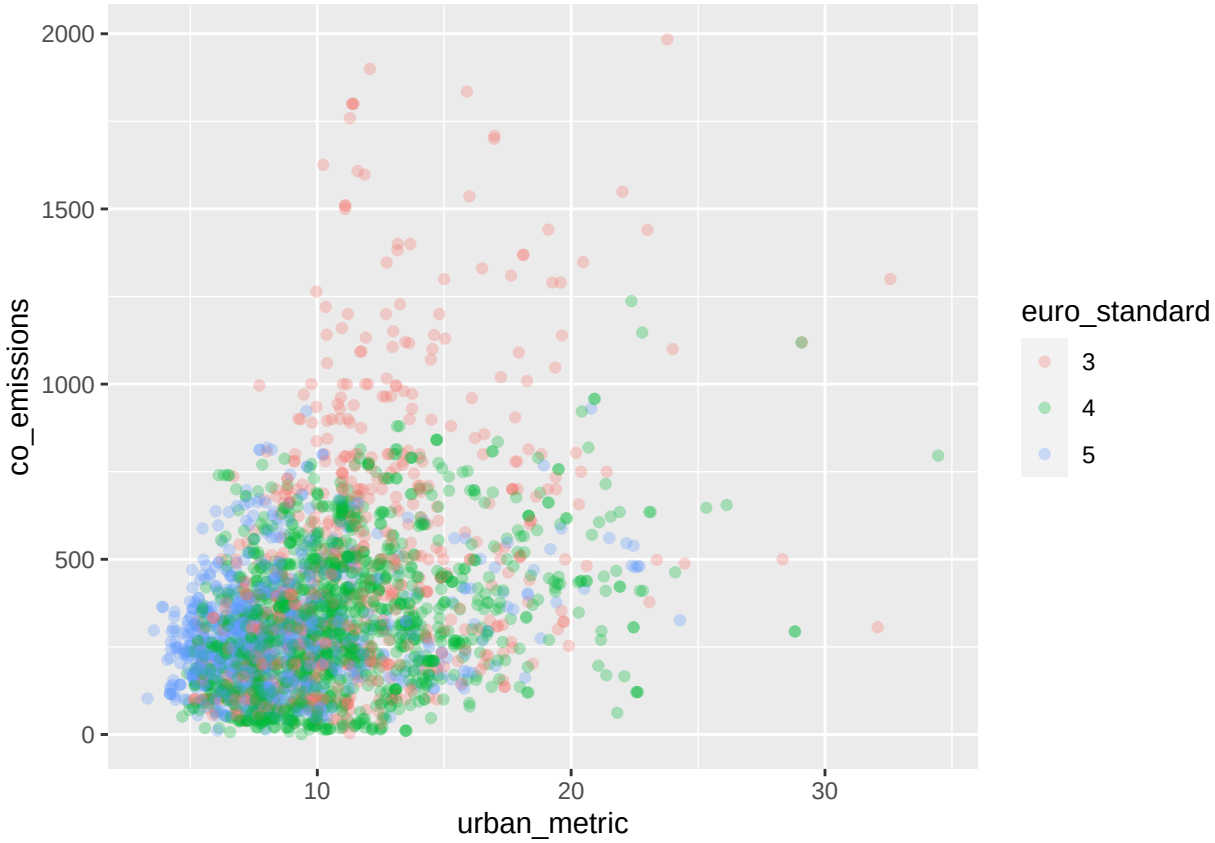

```
ggplot(subset.trans, aes(x = engine_capacity, y = nox_emissions, color = transmission_type)) +
  geom_jitter(alpha = 0.3, width = 100) +
  facet_grid(~fuel_type)
```



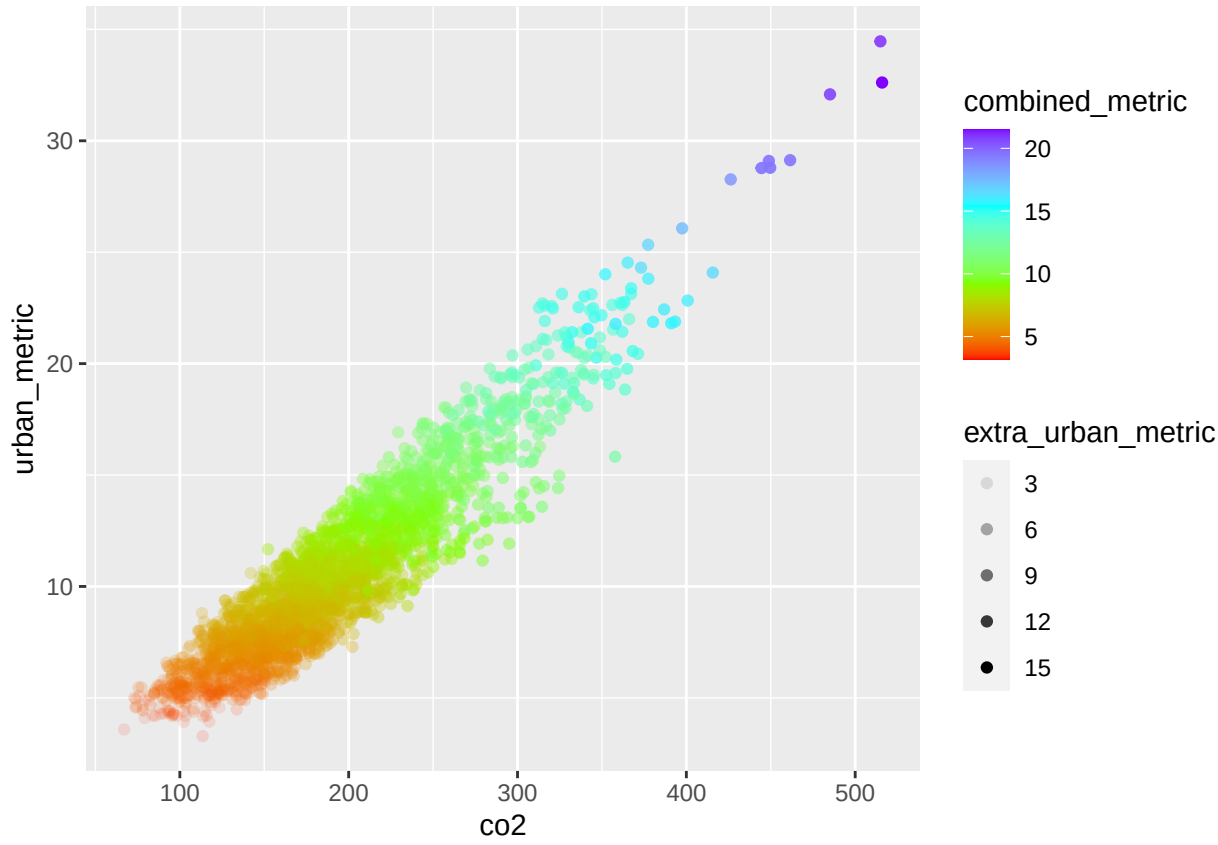

```
ggplot(subset.trans, aes(x = engine_capacity, y = urban_metric, color = fuel_type)) +
  geom_jitter(alpha = 0.3, width = 100)+
  scale_color_manual(values=c("purple", "green"))
```



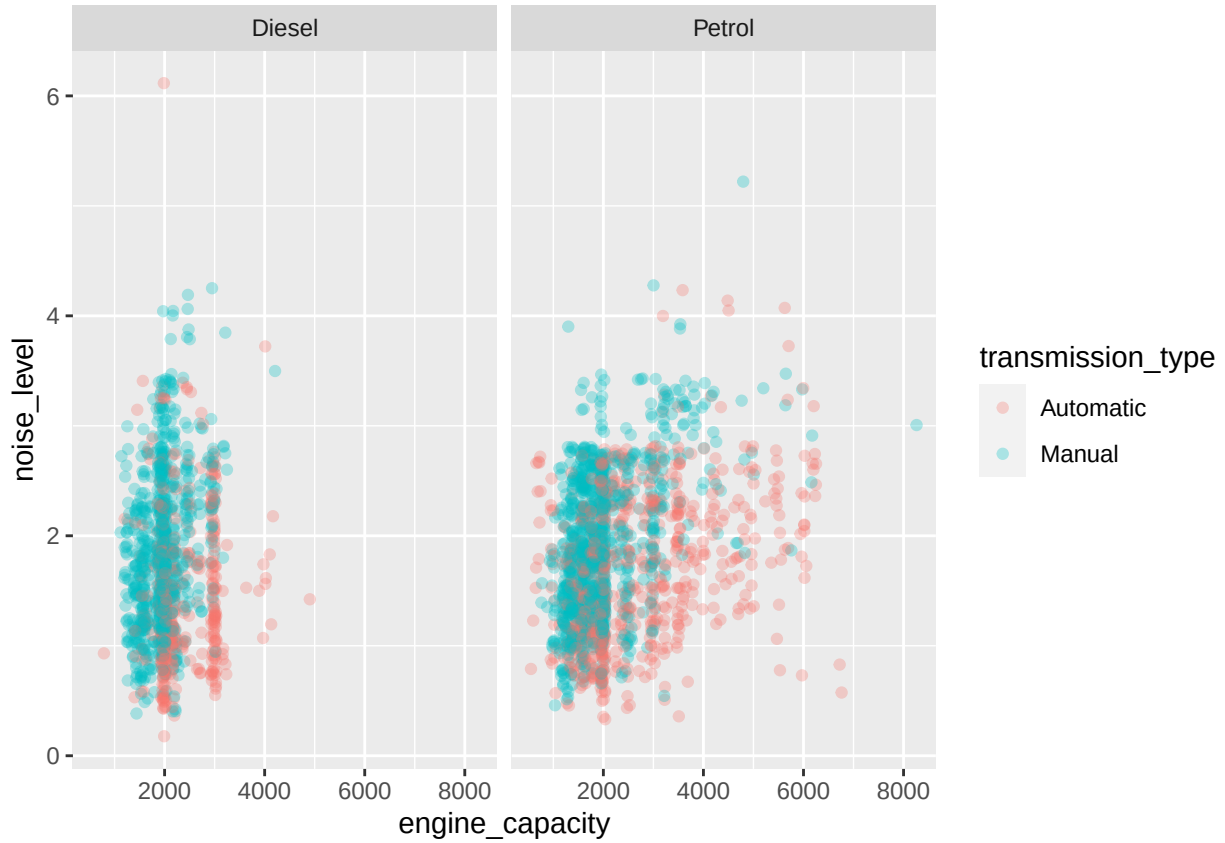

```
ggplot(subset.trans, aes(x = urban_metric, y = co_emissions, color = euro_standard)) +
  geom_jitter(alpha = 0.3 )
```



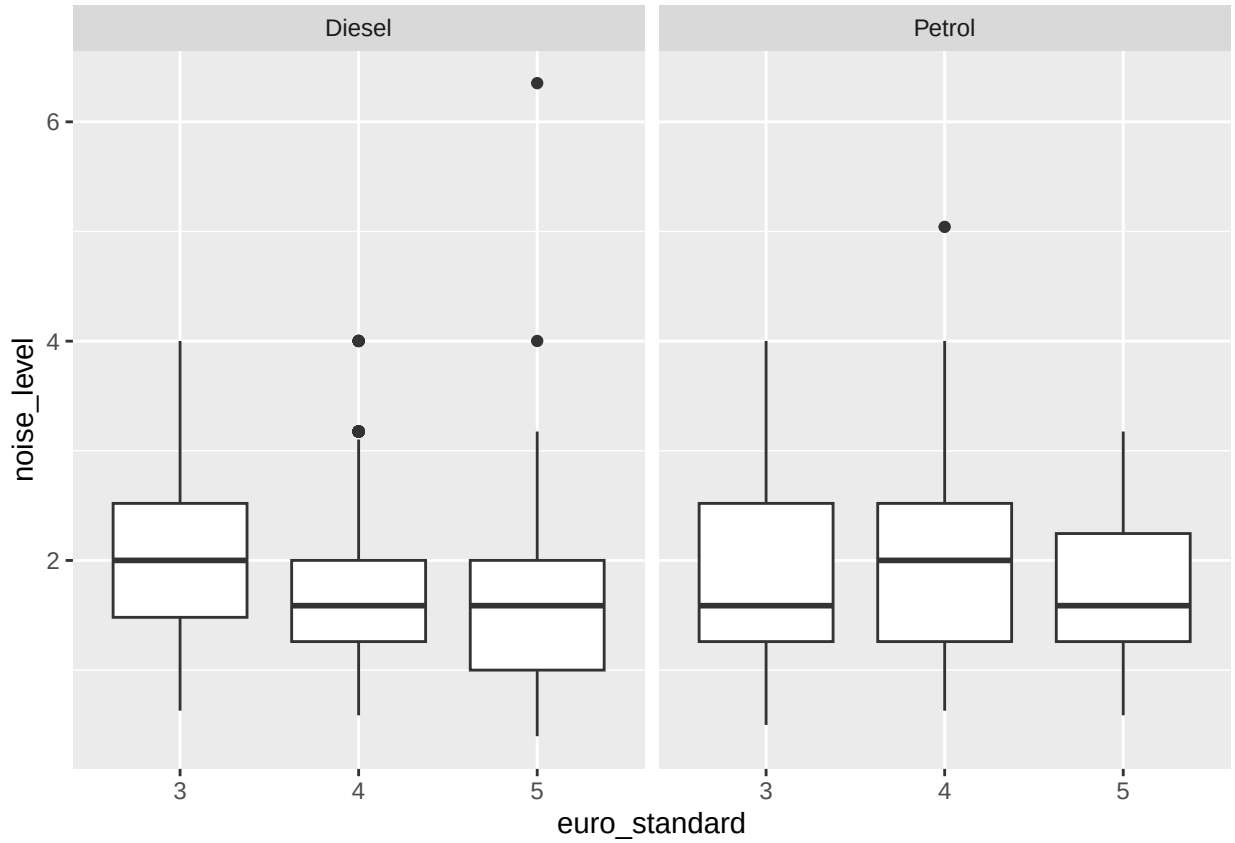

```
ggplot(subset.trans, aes(x = co2, y = urban_metric, color = combined_metric, alpha = extra_urban_metric)
  geom_jitter( width=30)+
  scale_colour_gradientn(colours=rainbow(4))
```




```
# a non linear jitter would be more appropriate ...
ggplot(subset.trans, aes(x = engine_capacity, y = noise_level, color = transmission_type)) +
  geom_jitter(alpha = 0.3, width=50, height = 0.3 ) +
  facet_grid(~fuel_type)
```




```
ggplot(subset.trans, aes(x = euro_standard, y = noise_level)) +
  geom_boxplot() +
  facet_grid(~fuel_type)
```




These plot show that there are a lot of (near) linear relationships between variables and that the discrete variables are important. It makes it easy to find interesting rows.

### 1.6 intersting rows

```
subset.trans[subset.trans$engine_capacity> 8000,]
```

```
##       manufacturer model description euro_standard transmission_type
## 36079        Dodge SRT10   Cabriolet             4            Manual
##       engine_capacity fuel_type urban_metric extra_urban_metric combined_metric
## 36079            8285    Petrol         34.5               13.2            21.1
##       noise_level co2 co_emissions nox_emissions
## 36079    3.175797 488          796             9
```

```
subset.trans[subset.trans$nox_emissions> 200 & subset.trans$fuel_type == "Petrol",]
```

```
##       manufacturer                  model             description euro_standard
## 22186       Proton                   Wira        1800 5 Door DOHC             3
## 27077       Proton                  Putra               1800 DOHC             3
## 20835      Renault Laguna II Sport Tourer 2.0 16V From VIN KG0K05             3
## 28466       Proton                   Wira        1800 4 Door DOHC             3
## 21588        Honda   CR-V (up to 2001 YM)                2.0i ES             3
## 13484    Chevrolet                 Blazer                4.3L V6             3
##       transmission_type engine_capacity fuel_type urban_metric
## 22186            Manual            1834    Petrol         14.0
## 27077            Manual            1834    Petrol         14.0
## 20835            Manual            1998    Petrol         10.9
```

```
## 28466            Manual            1834    Petrol         14.0
## 21588            Manual            1973    Petrol         11.9
## 13484         Automatic            4300    Petrol         19.1
##       extra_urban_metric combined_metric noise_level co2 co_emissions
## 22186                6.9             9.5    2.143990 227          710
## 27077                6.9             9.5    1.823742 227          710
## 20835                6.4             8.0    1.071794 190          600
## 28466                6.9             9.5    2.351641 227          700
## 21588                8.4             9.7    2.351641 229         1000
## 13484               11.1            14.0    2.520474 335          700
##       nox_emissions
## 22186           440
## 27077           440
## 20835           300
## 28466           400
## 21588           400
## 13484           800
```

```
subset.trans[subset.trans$co2> 440,]
```

```
##         manufacturer                          model    description euro_standard
## 35407    Lamborghini      Gallardo Model Year 2005            140             3
## 36784 Bentley Motors                    Brooklands         CoupÃ©             4
## 16794    Lamborghini      Gallardo Model Year 2004            140             4
## 36079          Dodge                         SRT10      Cabriolet             4
## 30694    Lamborghini   Murcielgo Model Year 2002/3            147             3
## 17575 Bentley Motors Arnage (from 2007 model year)              T             4
## 12481        Ferrari                    All Models 612 Scaglietti             3
##       transmission_type engine_capacity fuel_type urban_metric
## 35407            Manual            4961    Petrol         29.1
## 36784         Automatic            6761    Petrol         28.8
## 16794            Manual            4961    Petrol         29.1
## 36079            Manual            8285    Petrol         34.5
## 30694            Manual            6192    Petrol         32.6
## 17575         Automatic            6761    Petrol         28.8
## 12481            Manual            5748    Petrol         32.1
##       extra_urban_metric combined_metric noise_level co2 co_emissions
## 35407               13.9            19.5   2.8292248 450         1119
## 36784               14.1            19.5   0.7936508 465          294
## 16794               13.9            19.5   2.5204738 450         1119
## 36079               13.2            21.1   3.1757969 488          796
## 30694               15.1            21.5   2.5204738 500         1300
## 17575               14.1            19.5   0.6298816 465          294
## 12481               14.0            20.7   1.9546751 475          306
##       nox_emissions
## 35407            79
## 36784            31
## 16794            79
## 36079             9
## 30694           100
## 17575            31
## 12481            32
```

# 2 PCA

## 2.1 splitting into training and validation

```
subset.transNoText <- subset.trans[ , -which(names(subset.trans) %in% c("manufacturer"
,"model","description" ))]
subset.transNoText <- na.omit(subset.transNoText)
```

```
# it looks complicated because I had a bug and now I dont want to touch it ...
set.seed(180901)
train_ind <- sample.int(n=nrow(subset.trans),size=1500,replace=F)
subset.transNoText$inTrain = FALSE
subset.transNoText[train_ind,]$inTrain = TRUE
nrow(na.omit(subset.transNoText))
```

```
## [1] 3000
```

```
Xtrain = subset.transNoText[subset.transNoText$inTrain,]
Xtrain <- na.omit(Xtrain)
Xval = subset.transNoText[!(subset.transNoText$inTrain),]
Xval <- na.omit(Xval[!is.na(Xval),])
nrow(Xtrain)
```

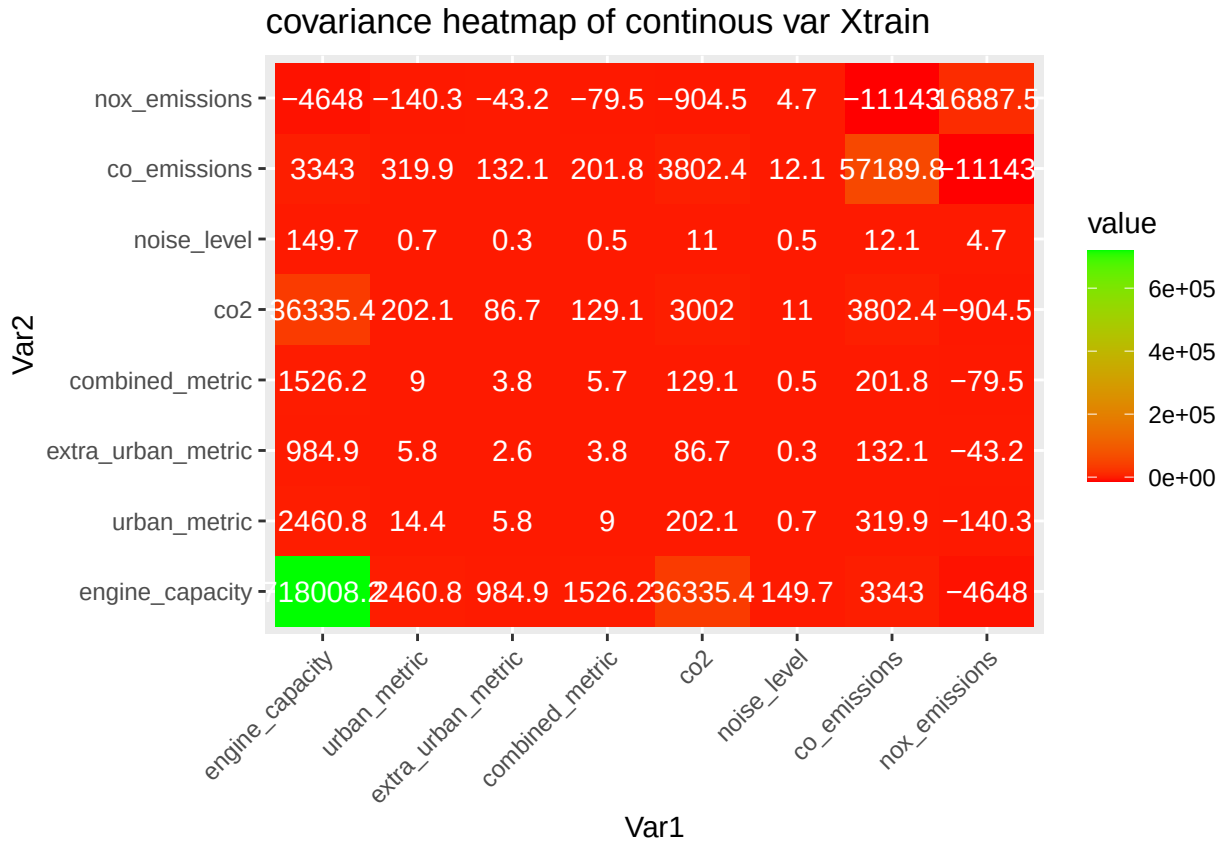
```
## [1] 1500
```

```
nrow(Xval)
```

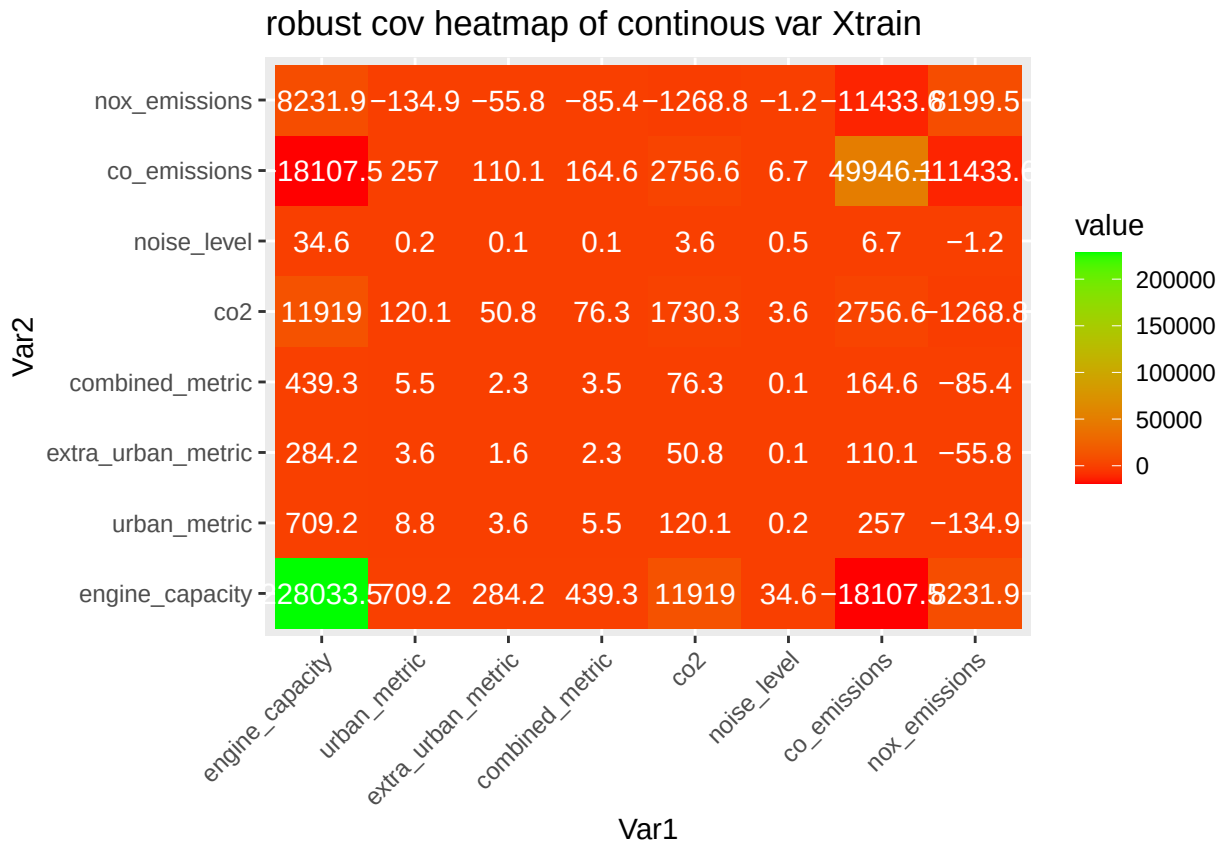
```
## [1] 1500
```

## 2.2 PCA on training

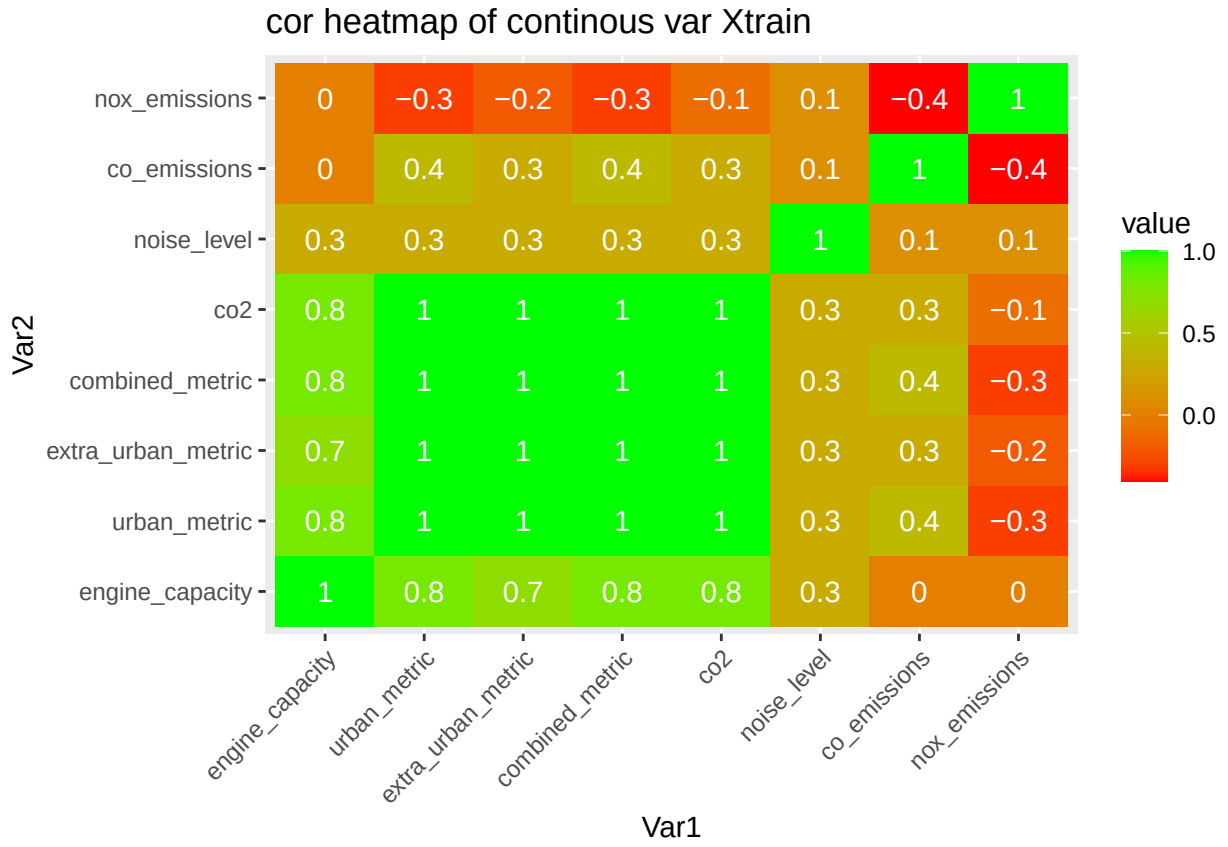
```
continousCols <- c("engine_capacity", "urban_metric", "extra_urban_metric", "combined_metric","co2", "n

ggplot(melt(round(cov(Xtrain[,continousCols]),1)),
       aes(x = Var1, y= Var2, fill= value))+
  geom_tile()+
  geom_text(aes(Var2, Var1, label = value),color = "white", size = 4)+
  theme(axis.text.x=element_text(angle=45, hjust=1,vjust=1)) +
  scale_fill_gradient(low = "red", high = "green")+
  ggtitle("covariance heatmap of continous var Xtrain")
```



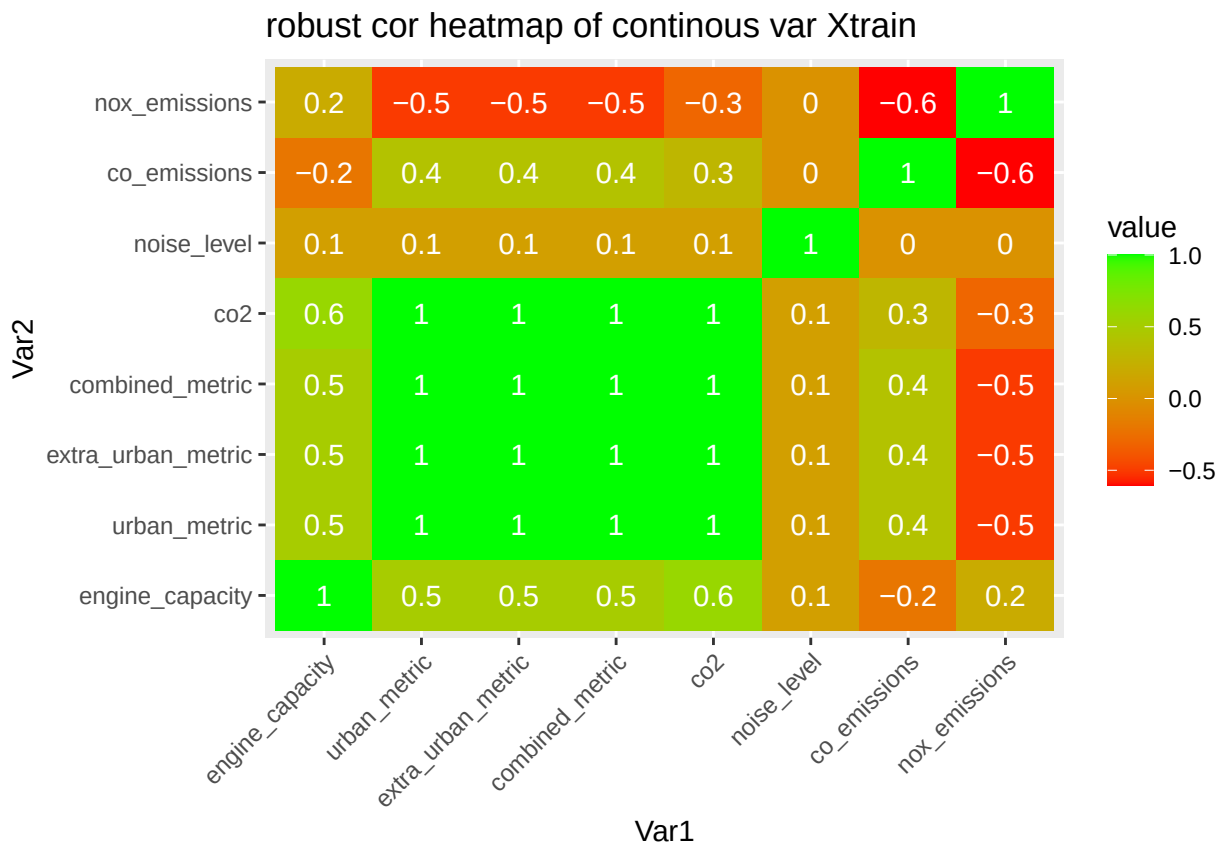

```
ggplot(melt(round(covMcd(Xtrain[,continousCols], cor = TRUE)$cov,1)),
       aes(x = Var1, y= Var2, fill= value))+
  geom_tile()+
  geom_text(aes(Var2, Var1, label = value),color = "white", size = 4)+
  theme(axis.text.x=element_text(angle=45, hjust=1,vjust=1)) +
  scale_fill_gradient(low = "red", high = "green")+
  ggtitle("robust cov heatmap of continous var Xtrain")
```




```
ggplot(melt(round(cor(Xtrain[,continousCols]),1)),
       aes(x = Var1, y= Var2, fill= value))+
  geom_tile()+
  geom_text(aes(Var2, Var1, label = value),color = "white", size = 4)+
  theme(axis.text.x=element_text(angle=45, hjust=1,vjust=1)) +
  scale_fill_gradient(low = "red", high = "green")+
  ggtitle("cor heatmap of continous var Xtrain")
```




```
ggplot(melt(round(covMcd(Xtrain[,continousCols], cor = TRUE)$cor,1)),
       aes(x = Var1, y= Var2, fill= value))+
  geom_tile()+
  geom_text(aes(Var2, Var1, label = value),color = "white", size = 4)+
  theme(axis.text.x=element_text(angle=45, hjust=1,vjust=1)) +
  scale_fill_gradient(low = "red", high = "green")+
  ggtitle("robust cor heatmap of continous var Xtrain")
```




From the variance heatmap it is clearly seen that pca on the covariance matrix would be dominated by engine capacity therefore we have to scale the data (or use the correlation) before doing PCA.

```
{
Xtrain.pca.unscaled = PcaClassic(Xtrain[,continousCols])
print(summary(Xtrain.pca.unscaled))
Xtrain.pca.unscaled$loadings
}
```

```
##
## Call:
## PcaClassic(x = Xtrain[, continousCols])
## Importance of components:
##                             PC1      PC2       PC3      PC4    PC5    PC6
## Standard deviation     848.4774 245.4904 118.31342 30.53100 0.6454 0.6363
## Proportion of Variance   0.9054   0.0758   0.01761  0.00117 0.0000 0.0000
## Cumulative Proportion    0.9054   0.9812   0.99883  1.00000 1.0000 1.0000
##                             PC7     PC8
## Standard deviation     0.2878 0.03883
## Proportion of Variance 0.0000 0.00000
## Cumulative Proportion  1.0000 1.00000
```

```
##                                PC1           PC2           PC3           PC4
## engine_capacity     0.9986694073 -0.0101106064  0.0042065004 -0.050392404
## urban_metric        0.0034317584  0.0055059331 -0.0030232307  0.061913749
## extra_urban_metric  0.0013737526  0.0022218144 -0.0002319066  0.030664163
## combined_metric     0.0021285844  0.0034416727 -0.0012861889  0.042173959
```

```
## co2                 0.0506547414  0.0617224764  0.0198587622  0.993287541
## noise_level         0.0002084503  0.0001608142  0.0005968358  0.002877939
## co_emissions        0.0054444488  0.9665795388  0.2478388202 -0.065344323
## nox_emissions      -0.0067551092 -0.2485256205  0.9685828049 -0.003171447
##                             PC5           PC6           PC7           PC8
## engine_capacity    -0.0002630839 -3.801125e-05  0.0001755488  2.208969e-06
## urban_metric        0.9472610253 -1.024526e-01  0.0406785881 -2.943819e-01
## extra_urban_metric -0.1952011655  1.140301e-02  0.8371527590 -5.098966e-01
## combined_metric     0.2221221081 -2.745481e-02  0.5429284997  8.082945e-01
## co2                -0.0627940313  4.310173e-03 -0.0514607291 -2.224476e-04
## noise_level         0.1062563426  9.942837e-01  0.0098076698 -2.165651e-03
## co_emissions       -0.0005725187 -5.349057e-05 -0.0002475313 -4.392907e-06
## nox_emissions       0.0045745230 -1.040703e-03  0.0021599938  3.941431e-05
```

Most of the variance gets captured by the first PC and the loadings aren't interesting because of the poor scaling.

```
Xtrain.pca.scaled = PcaClassic(Xtrain[,continousCols],scale =TRUE,crit.pca.distances=0.99)
summary(Xtrain.pca.scaled)
```

```
##
## Call:
## PcaClassic(x = Xtrain[, continousCols], scale = TRUE, crit.pca.distances = 0.99)
## Importance of components:
##                            PC1    PC2    PC3     PC4     PC5    PC6     PC7
## Standard deviation     2.200 1.1491 0.9390 0.80133 0.52045 0.1980 0.07619
## Proportion of Variance 0.605 0.1651 0.1102 0.08027 0.03386 0.0049 0.00073
## Cumulative Proportion  0.605 0.7700 0.8802 0.96048 0.99434 0.9992 0.99997
##                            PC8
## Standard deviation     0.01632
## Proportion of Variance 0.00003
## Cumulative Proportion  1.00000
```

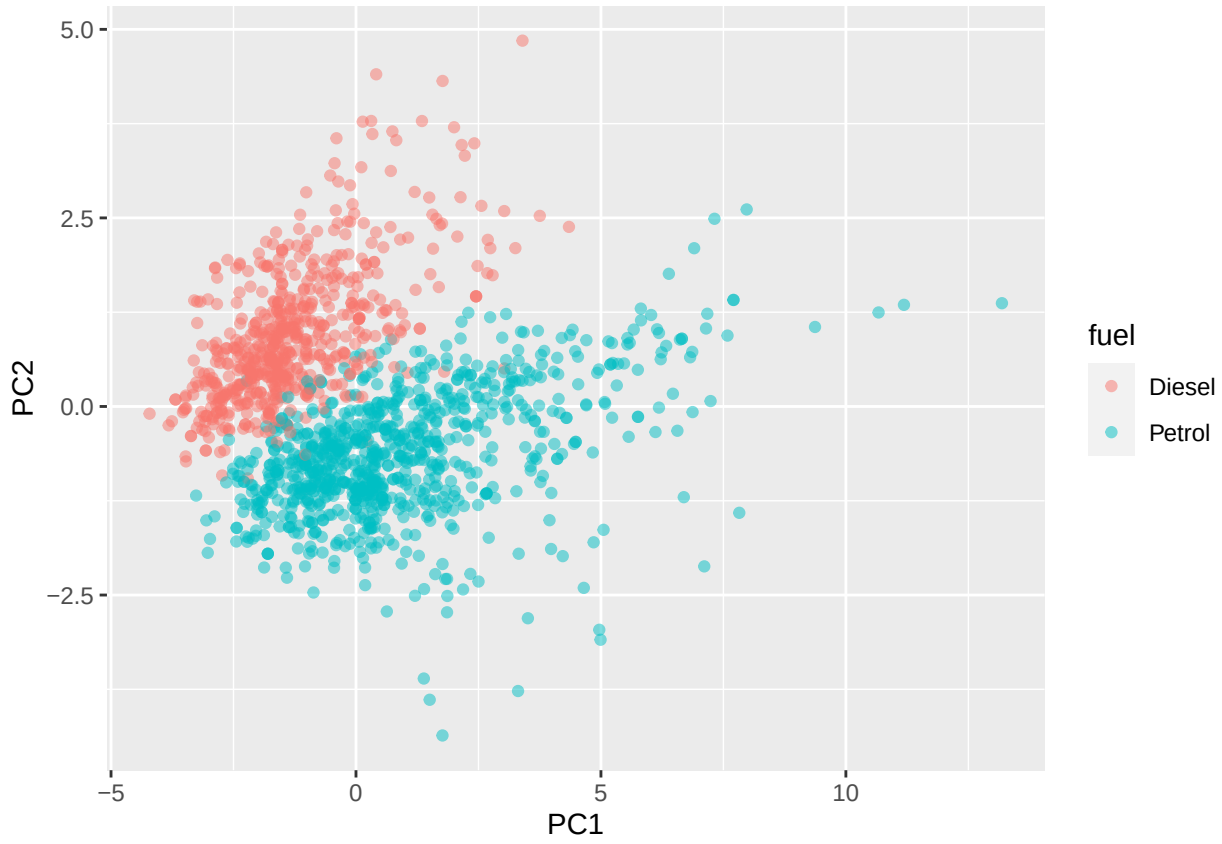
```
Xtrain.pca.scaled$loadings
```

```
##                            PC1         PC2         PC3         PC4         PC5
## engine_capacity     0.3645715  0.29326537 -0.19390237 -0.18300493  0.83298929
## urban_metric        0.4479022 -0.02425818 -0.07575889 -0.01116939 -0.11590667
## extra_urban_metric  0.4421129  0.01302090 -0.06484149  0.08742907 -0.30825381
## combined_metric     0.4506531 -0.01006524 -0.07098758  0.02985197 -0.20020543
## co2                 0.4457684  0.09606274 -0.07957677  0.10147217 -0.18866030
## noise_level         0.1596613  0.28138372  0.90745525 -0.26750009 -0.01695428
## co_emissions        0.1715616 -0.60932945  0.34212847  0.60002775  0.34856752
## nox_emissions      -0.1195182  0.67338613  0.02365217  0.71830583 -0.01457356
##                             PC6          PC7           PC8
## engine_capacity    -0.122086007 -0.034878340  0.0012082266
## urban_metric        0.697165918 -0.267225145 -0.4712186421
## extra_urban_metric -0.693183870 -0.314964762 -0.3432073961
## combined_metric     0.111520878 -0.279754493  0.8124643782
## co2                 0.003166202  0.860101300 -0.0075139800
## noise_level        -0.002592673 -0.002373134 -0.0005973052
## co_emissions       -0.020644828  0.016532182 -0.0004438165
## nox_emissions       0.075305707 -0.099275648  0.0021989360
```

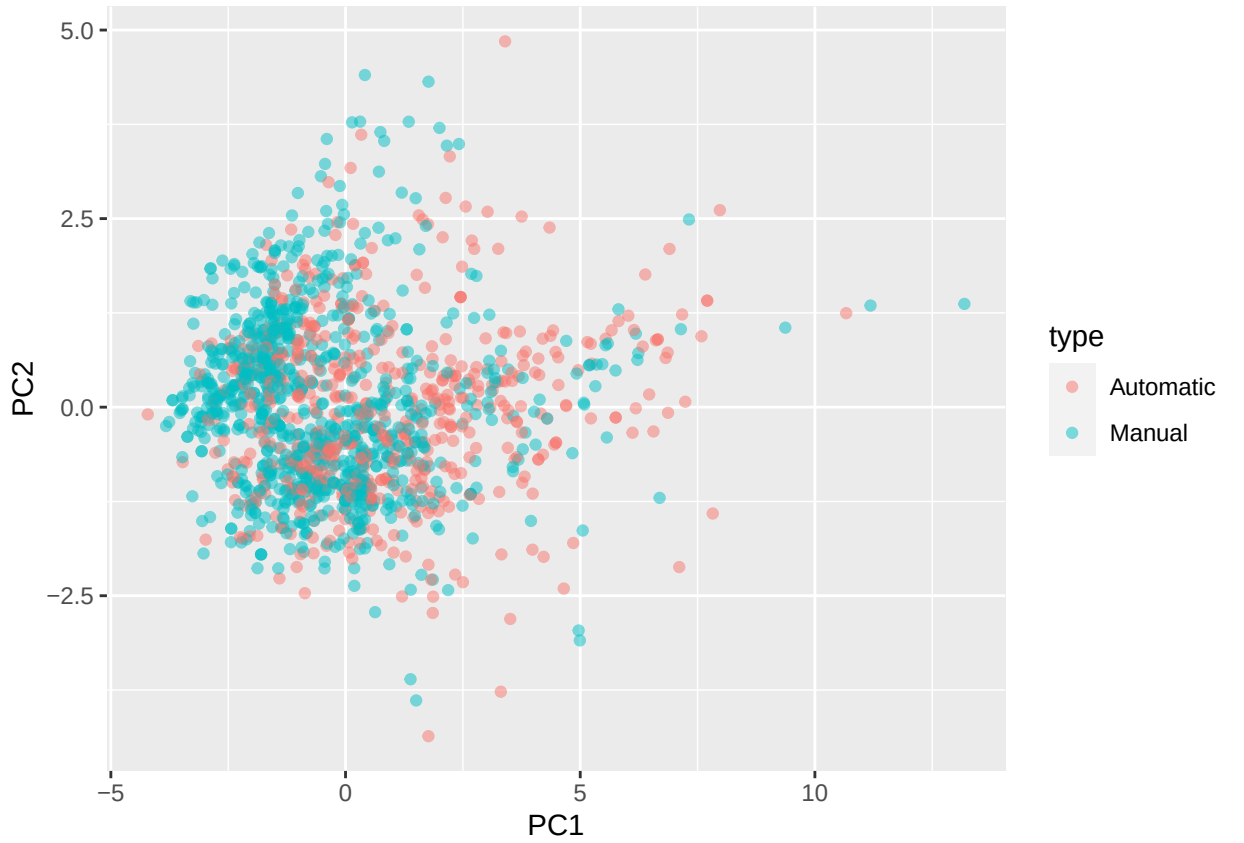
In would save 3 PC, the first 2 for sure because they capture a lot variances and the third because it is basically the noise level (look loadings) which is a variable that sticks out.

### 2.3 biplots PCA

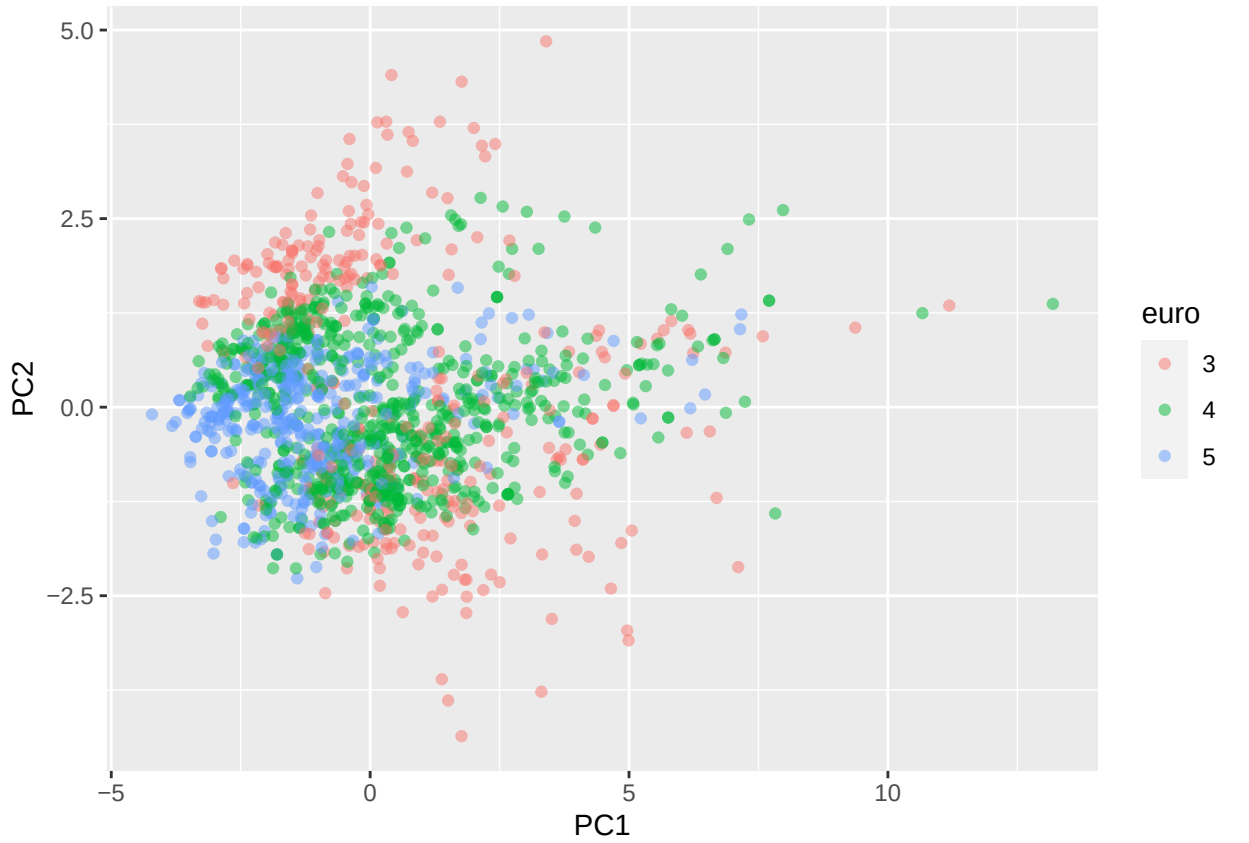
```
# plotting scores
score_plot <- data.frame(Xtrain.pca.scaled$scores)
score_plot$fuel <-Xtrain[,"fuel_type"]
score_plot$type <-Xtrain[,"transmission_type"]
score_plot$euro <-Xtrain[,"euro_standard"]

ggplot(score_plot ,aes(PC1,PC2, color = fuel)) + geom_point(alpha = 0.5)
```



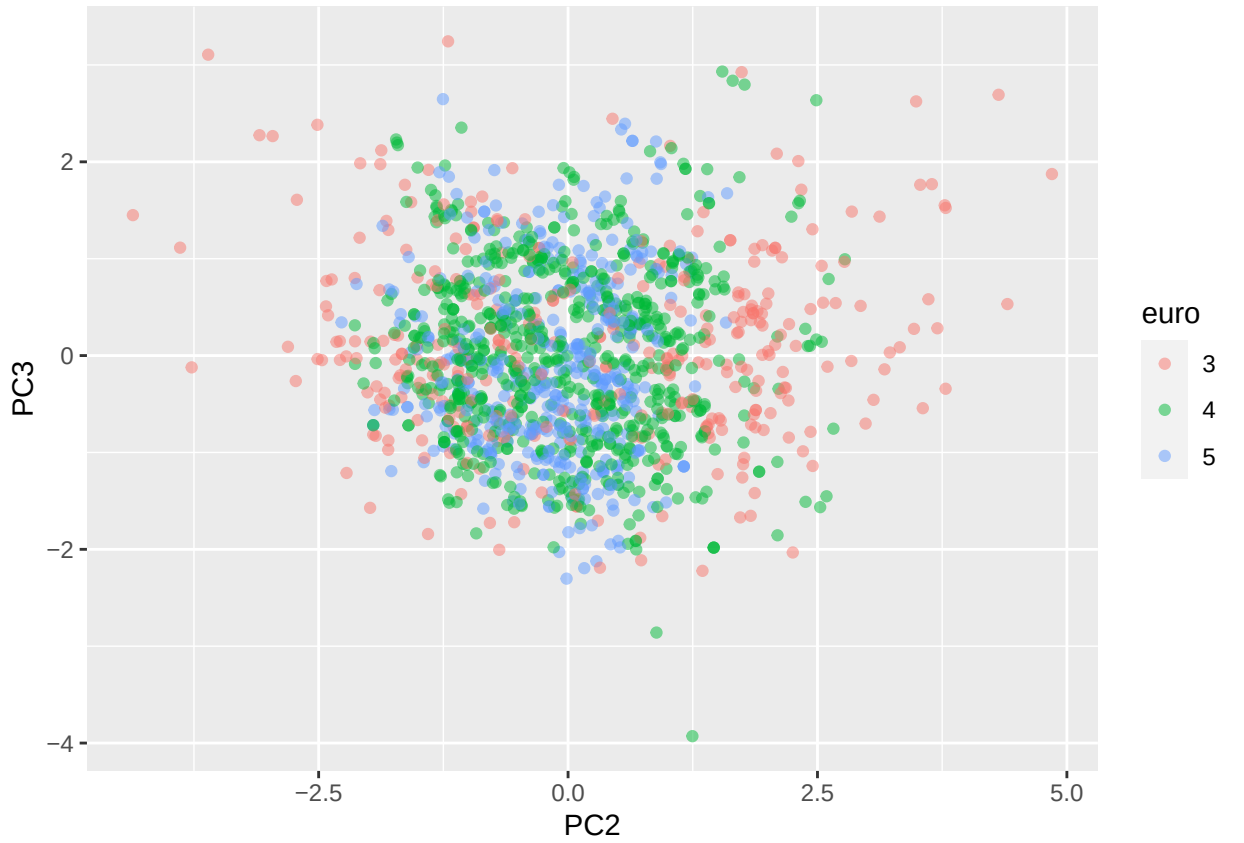

```
ggplot(score_plot ,aes(PC1,PC2, color = type)) + geom_point(alpha = 0.5)
```



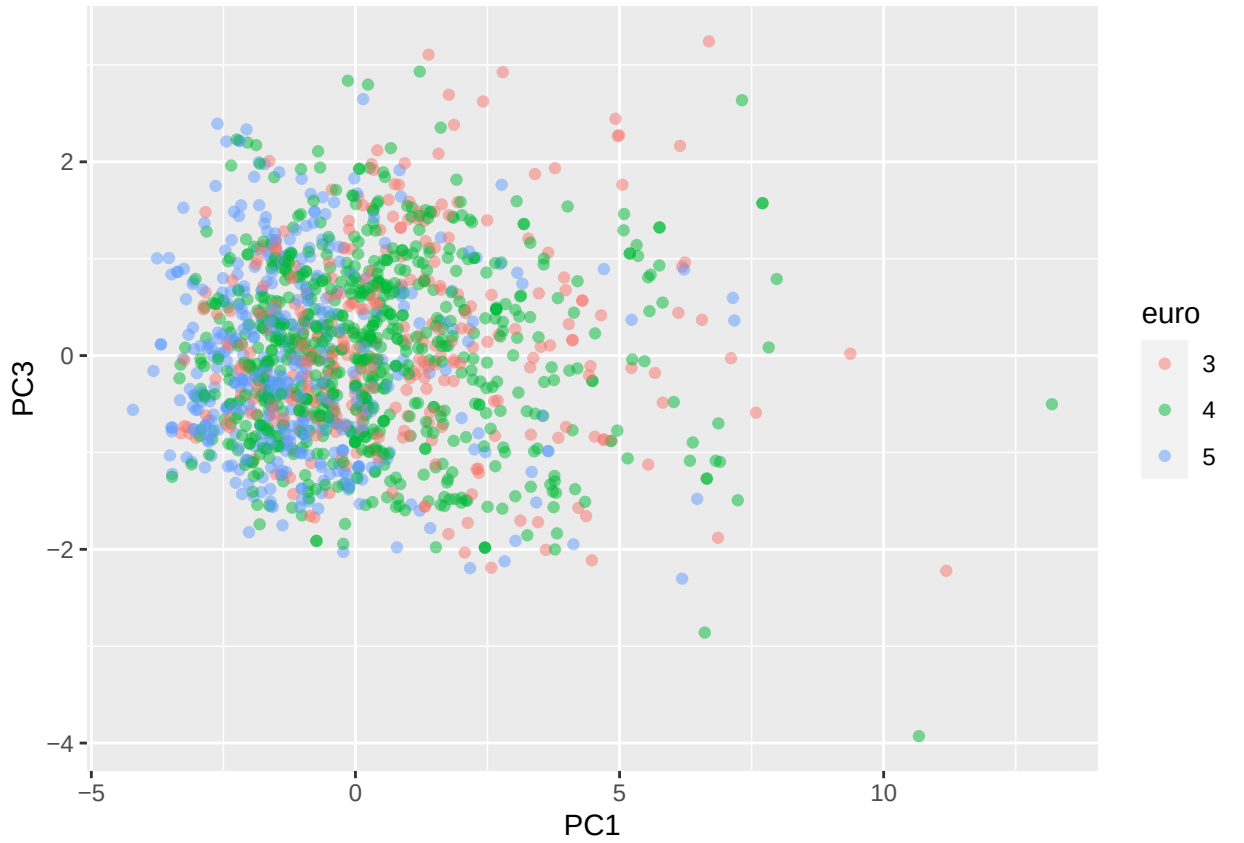

```
ggplot(score_plot ,aes(PC1,PC2, color = euro)) + geom_point(alpha = 0.5)
```




```
ggplot(score_plot ,aes(PC2,PC3, color = euro)) + geom_point(alpha = 0.5)
```




```
ggplot(score_plot ,aes(PC1,PC3, color = euro)) + geom_point(alpha = 0.5)
```




We could only recognize the fuel type out of the principal components.

### 2.4 compare to robust PCA

```
Xtrain.robpca <-PcaHubert(Xtrain[,continousCols], k=3,scale= mad,crit.pca.distances=0.99)
summary(Xtrain.robpca)
```

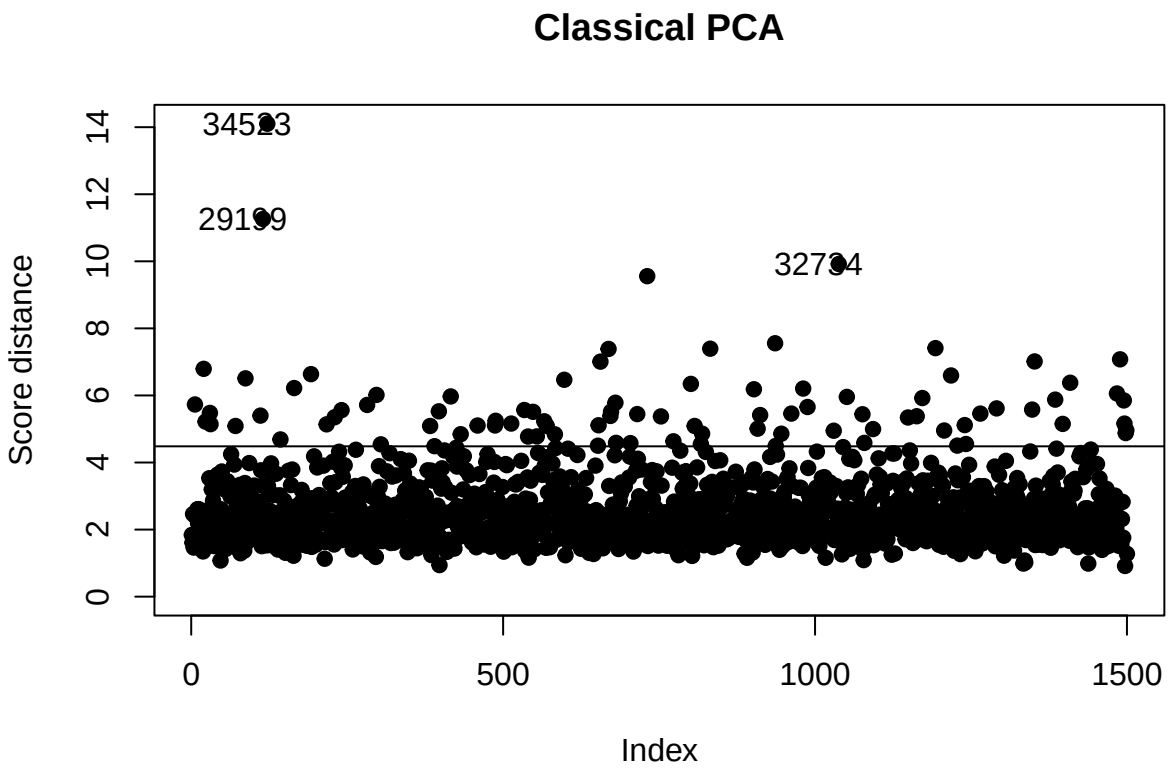
```
##
## Call:
## PcaHubert(x = Xtrain[, continousCols], k = 3, scale = mad, crit.pca.distances = 0.99)
## Importance of components:
##                            PC1    PC2    PC3
## Standard deviation     2.3202 1.4146 1.0879
## Proportion of Variance 0.5553 0.2064 0.1221
## Cumulative Proportion  0.5553 0.7617 0.8838
```

```
Xtrain.robpca$loadings
```

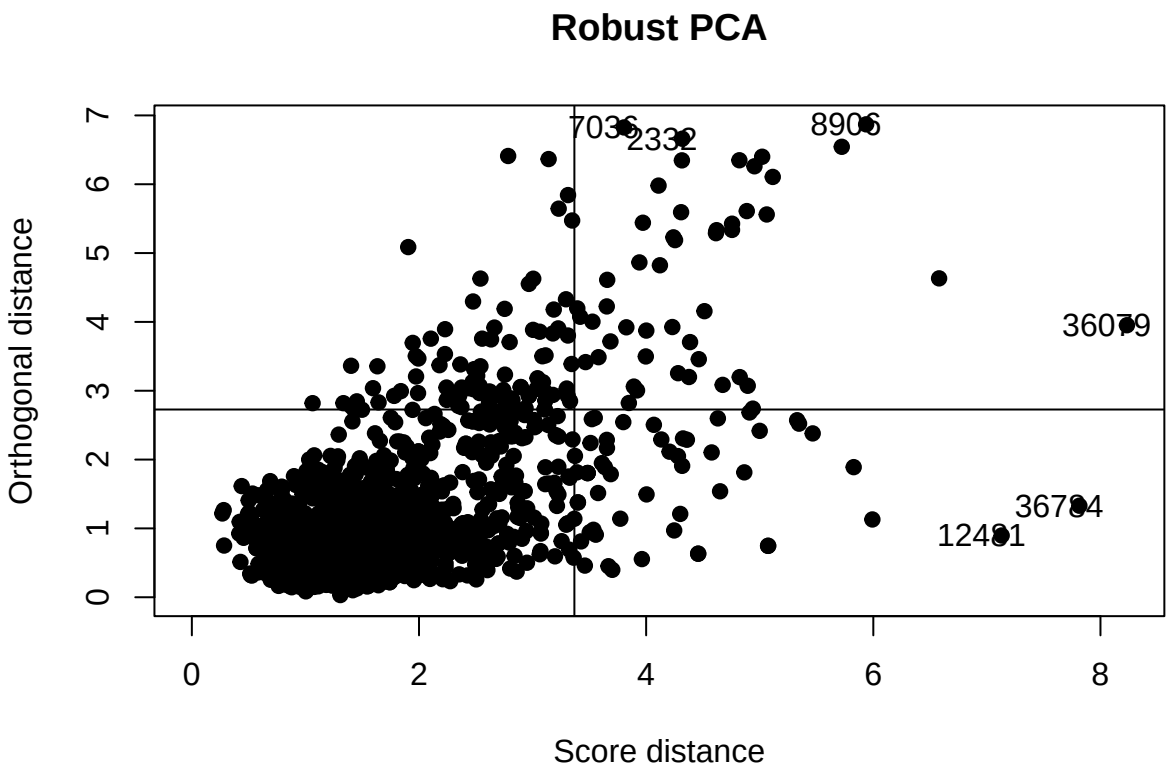
```
##                             PC1         PC2         PC3
## engine_capacity    -0.3195506 -0.51086403  0.06520871
## urban_metric       -0.4595804 -0.07100224  0.07352439
## extra_urban_metric -0.3924848 -0.05472938  0.09417052
## combined_metric    -0.4163030 -0.06024124  0.07944701
## co2                -0.4234058 -0.16231766  0.09312102
## noise_level        -0.1338704 -0.14187366 -0.97662779
## co_emissions       -0.2267574  0.53815975 -0.10973282
## nox_emissions       0.3327282 -0.62550878  0.02544445
```

Looks very similar to not robust PCA maybe the loading of nox emissions is a little bit of.

```
plot(Xtrain.pca.scaled,pch=19)
```




```
plot(Xtrain.robpca,pch=19)
```




```
bad_leverage_points <- c("7036","2332","8906","36079")
good_leverage_points <- c("12481","36784")
print(subset.trans[bad_leverage_points,])
```

```
##       manufacturer              model                 description
## 7036     SsangYong             Rexton               270 SX Manual
## 2332       Renault             Espace 1.9 dCi (JK0GB6) (w/sunroof)
## 8906    Land Rover Discovery Series II    2.5 TD5 (2002 Model Year)
## 36079        Dodge              SRT10                   Cabriolet
##       euro_standard transmission_type engine_capacity fuel_type urban_metric
## 7036              3            Manual            2696    Diesel         10.9
## 2332              3            Manual            1870    Diesel          8.9
## 8906              3            Manual            2495    Diesel         11.5
## 36079             4            Manual            8285    Petrol         34.5
##       extra_urban_metric combined_metric noise_level co2 co_emissions
## 7036                 7.7             8.9    3.032345 237          780
## 2332                 6.0             7.0    1.587600 187          337
## 8906                 8.2             9.4    4.001504 262          300
## 36079               13.2            21.1    3.175797 488          796
##       nox_emissions
## 7036            580
## 2332            694
## 8906            700
## 36079             9
```

```
print(subset.trans[good_leverage_points,])
```

```
##         manufacturer      model    description euro_standard transmission_type
## 12481        Ferrari All Models 612 Scaglietti             3            Manual
## 36784 Bentley Motors Brooklands         CoupÃ©             4         Automatic
##       engine_capacity fuel_type urban_metric extra_urban_metric combined_metric
## 12481            5748    Petrol         32.1               14.0            20.7
## 36784            6761    Petrol         28.8               14.1            19.5
##       noise_level co2 co_emissions nox_emissions
## 12481   1.9546751 475          306            32
## 36784   0.7936508 465          294            31
```

There are a lot of bad leverage points so we choose to continue with the robust PCA.

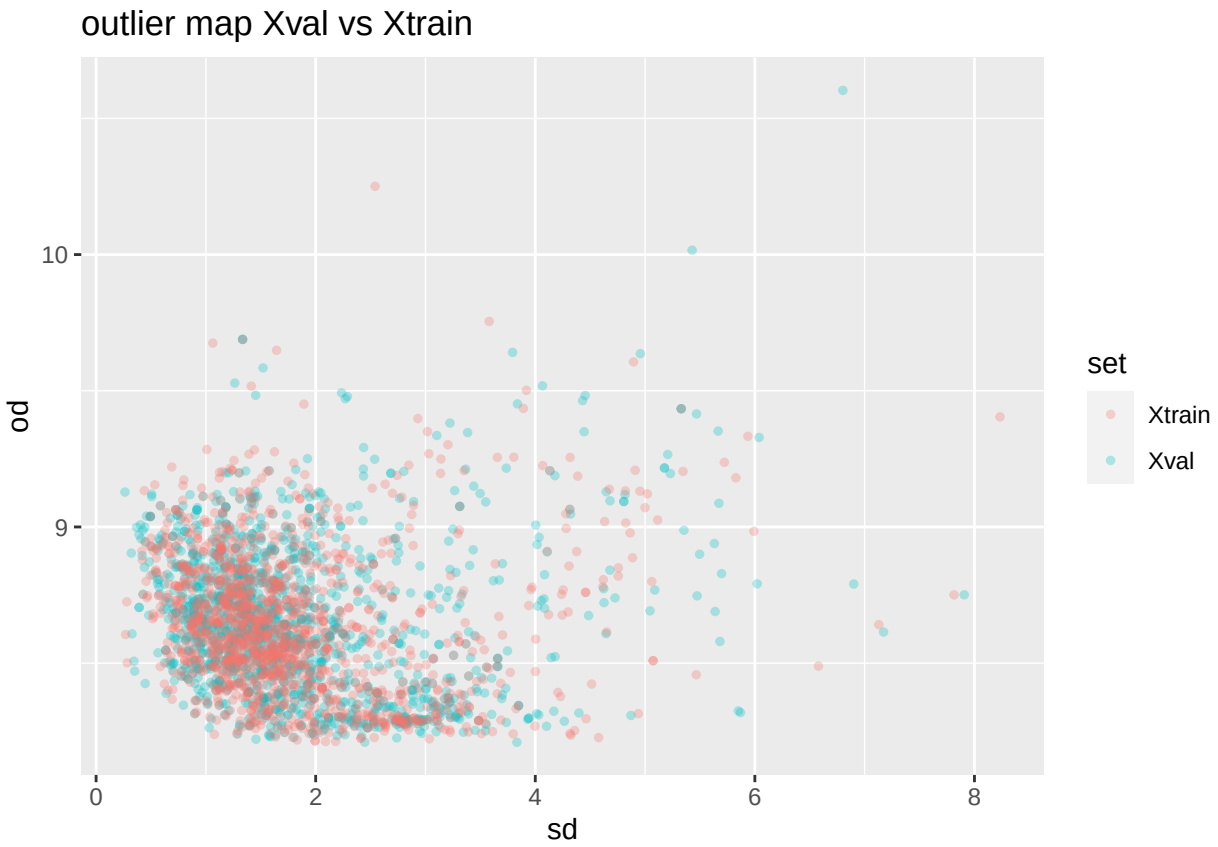
### 2.5 scores of validation set

```
Xval.robpca.scaled <- scale(Xval[,continousCols],center = Xtrain.robpca$center, scale= Xtrain.robpca$sca
Xval.robpca.scores <- Xval.robpca.scaled %*% Xtrain.robpca$loadings
Xval.robpca.pred <- t(t(Xval.robpca.scores%*%t(Xtrain.robpca$loadings)) + Xtrain.robpca$center/Xtrain.r

Xtrain.robpca.scaled <- scale(Xtrain[,continousCols],center = Xtrain.robpca$center, scale= Xtrain.robpca
Xtrain.robpca.scores <- Xtrain.robpca.scaled %*% Xtrain.robpca$loadings
Xtrain.robpca.pred <- t(t(Xtrain.robpca.scores%*%t(Xtrain.robpca$loadings)) + Xtrain.robpca$center/Xtra
```

### 2.6 outlier map validation set

To make an outlier map we need the orthogonal and score distances.

```
euclnorm <- function(y) sqrt(sum(y^2))
Xval.robpca.od <- apply(Xval.robpca.scaled - Xval.robpca.pred, 1, euclnorm)
Xval.robpca.sd <- sqrt(mahalanobis(Xval.robpca.scores, center = 0, diag(Xtrain.robpca$eigenvalues)))

Xtrain.robpca.od <- apply(Xtrain.robpca.scaled - Xtrain.robpca.pred, 1, euclnorm)
Xtrain.robpca.sd <- sqrt(mahalanobis(Xtrain.robpca.scores, center = 0, diag(Xtrain.robpca$eigenvalues)))

outlier_map <- data.frame(
  c(Xval.robpca.od,Xtrain.robpca.od),
  c(Xval.robpca.sd,Xtrain.robpca.sd),
  c(rep("Xval", 1500), rep("Xtrain",1500)))
names(outlier_map) <- c("od","sd","set")
ggplot(outlier_map,aes(sd,od, color = set)) +
  geom_point(alpha = 0.3, size = 1)+
  ggtitle("outlier map Xval vs Xtrain")
```



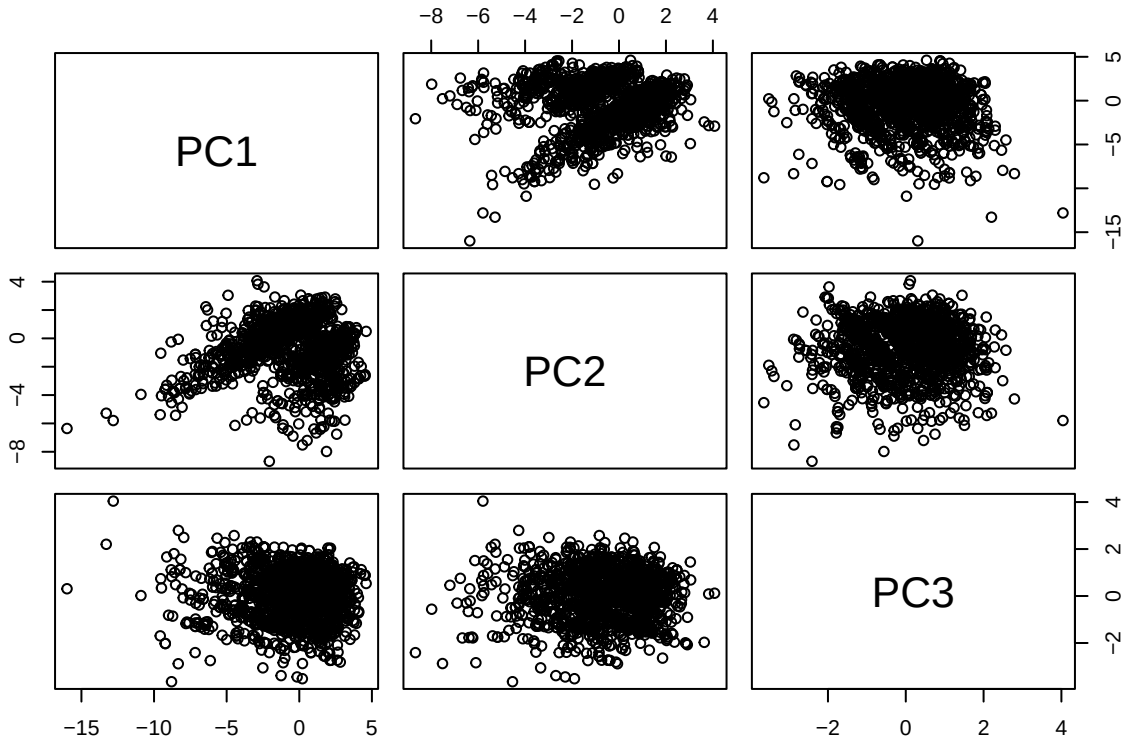

This outlier map isn't the same as from the robust PCA. As you can see the distribution of the Xval is barely different from the Xtrain on the outlier map this validates the robust PCA.

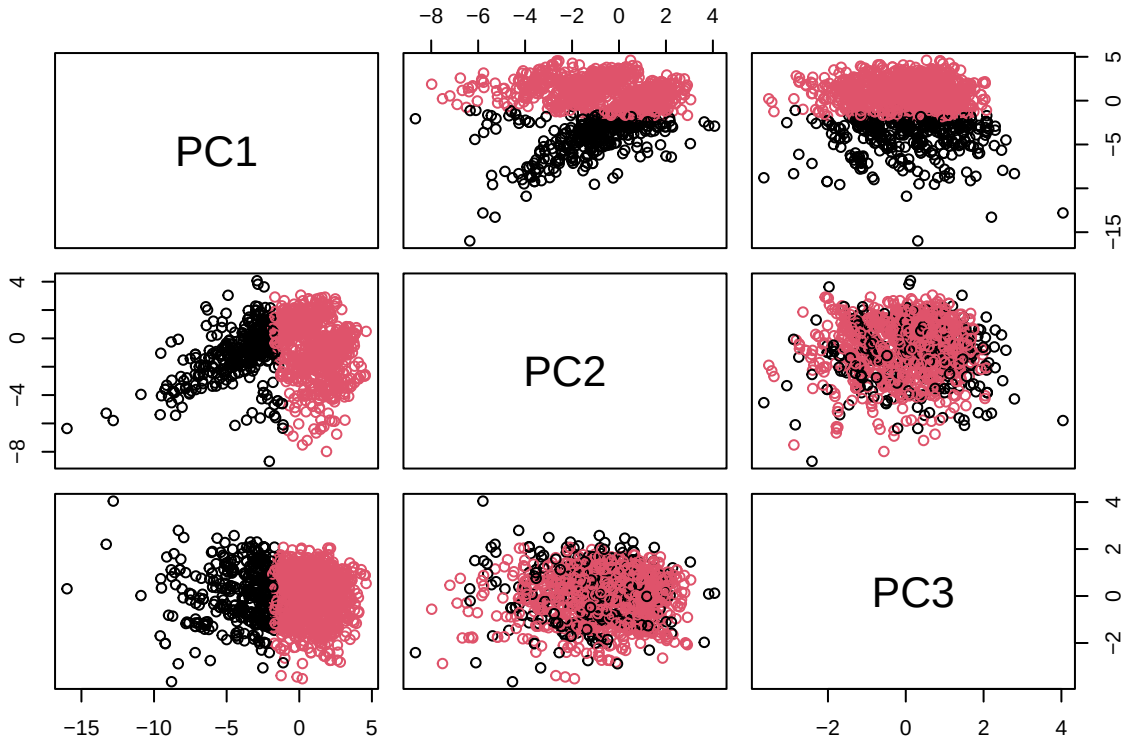
# 3 Clustering

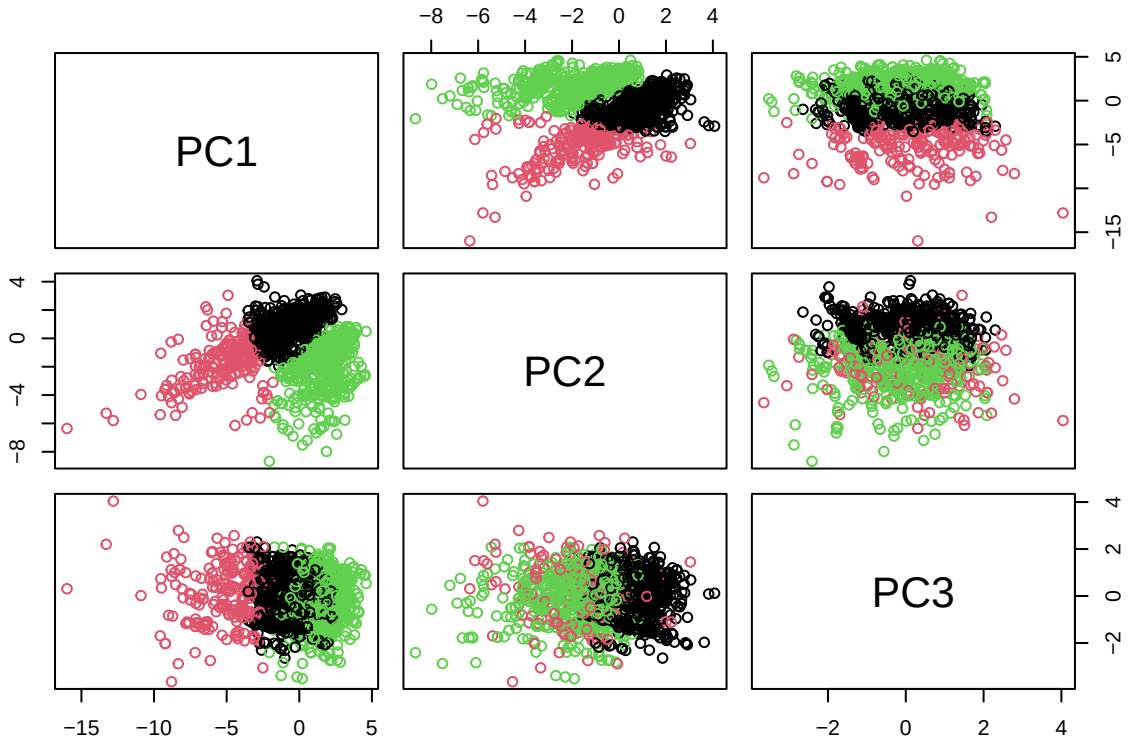
## 3.1 partitioning cluster analysis

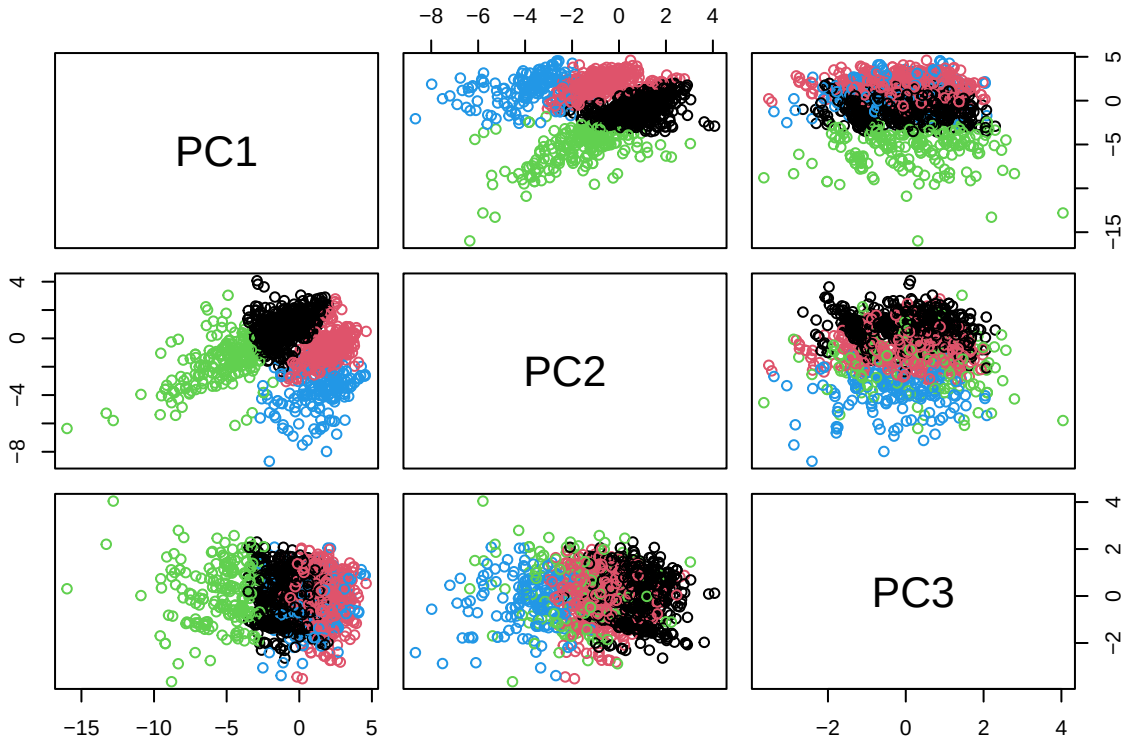
### 3.1.1 kmeans

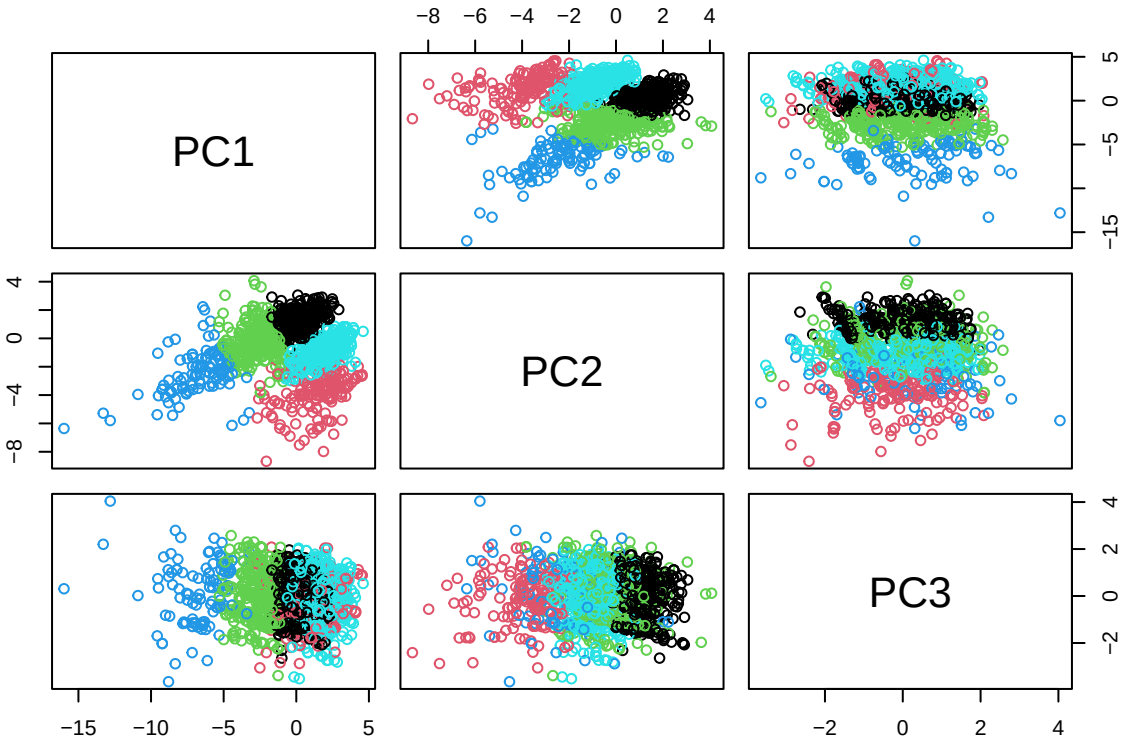
```
for (amount_centers in 1:6) {
  Xtrain.kmeans <-kmeans(Xtrain.robpca.scaled, amount_centers)
  pairs(Xtrain.robpca.scores, col=Xtrain.kmeans$cluster)
}
```

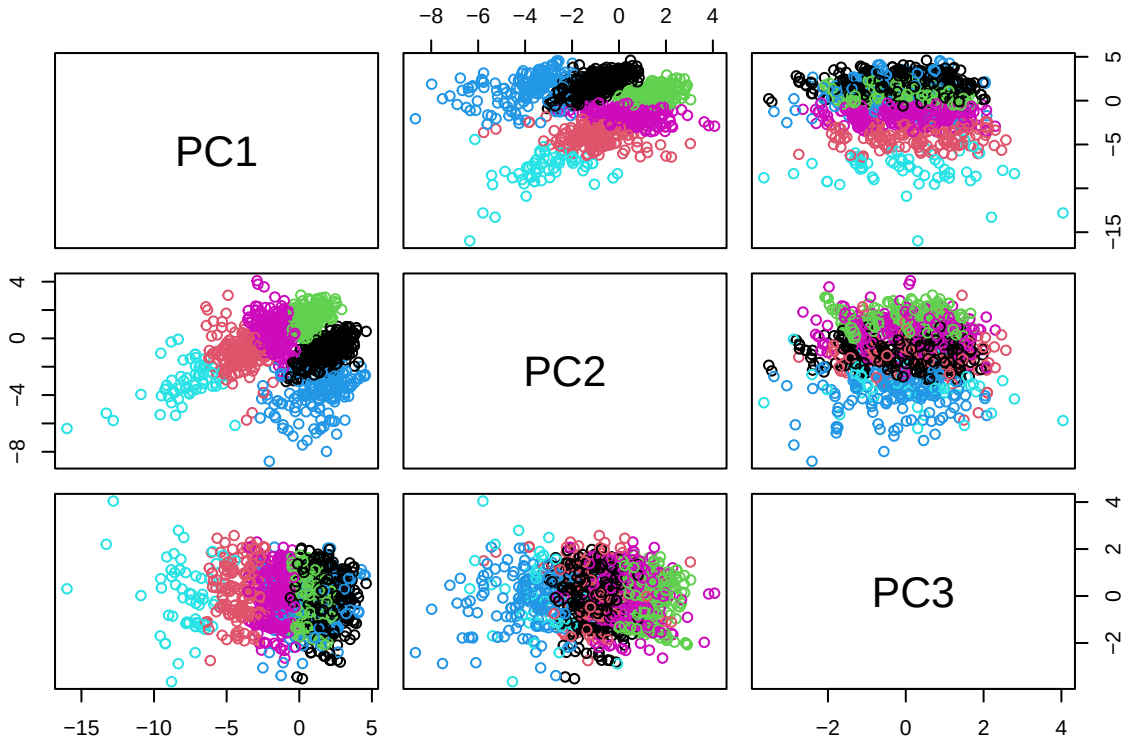

PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2


PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2


PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2

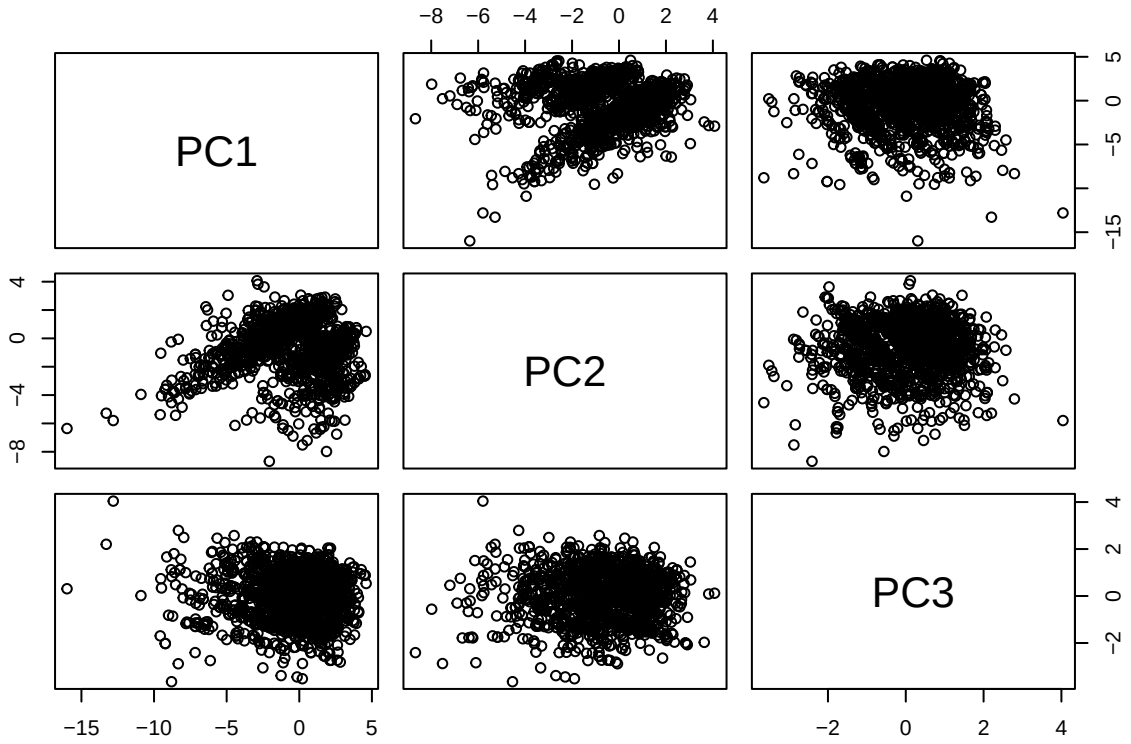

PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2

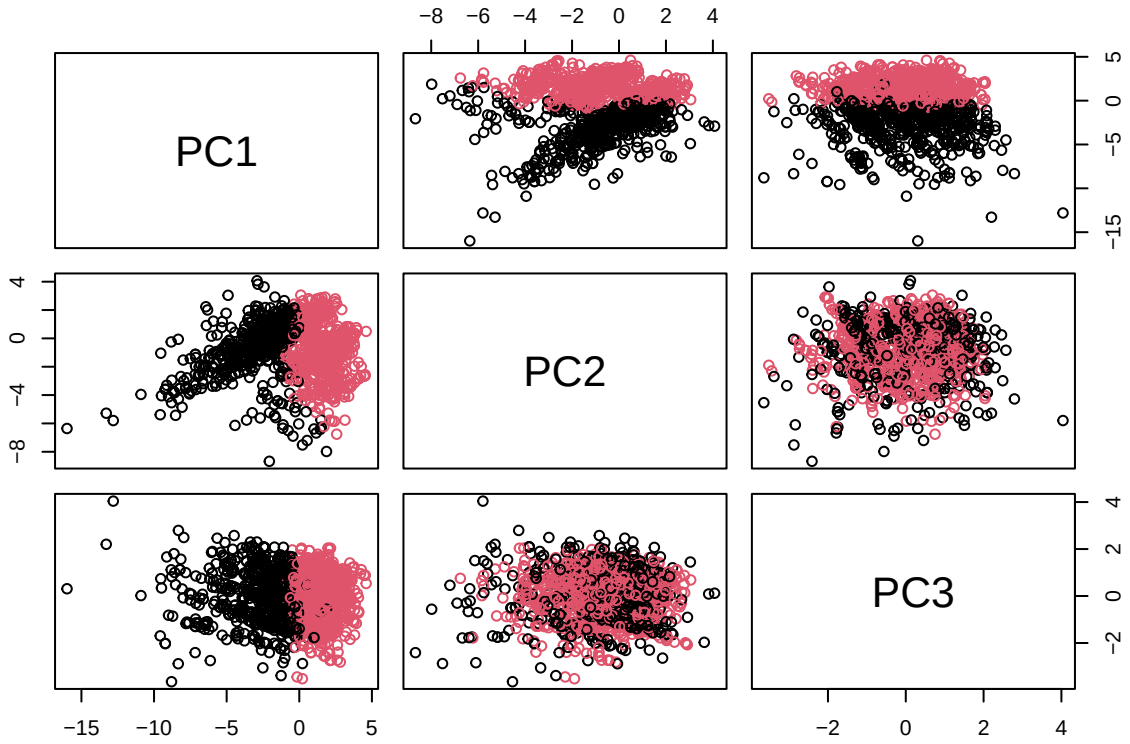

PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2

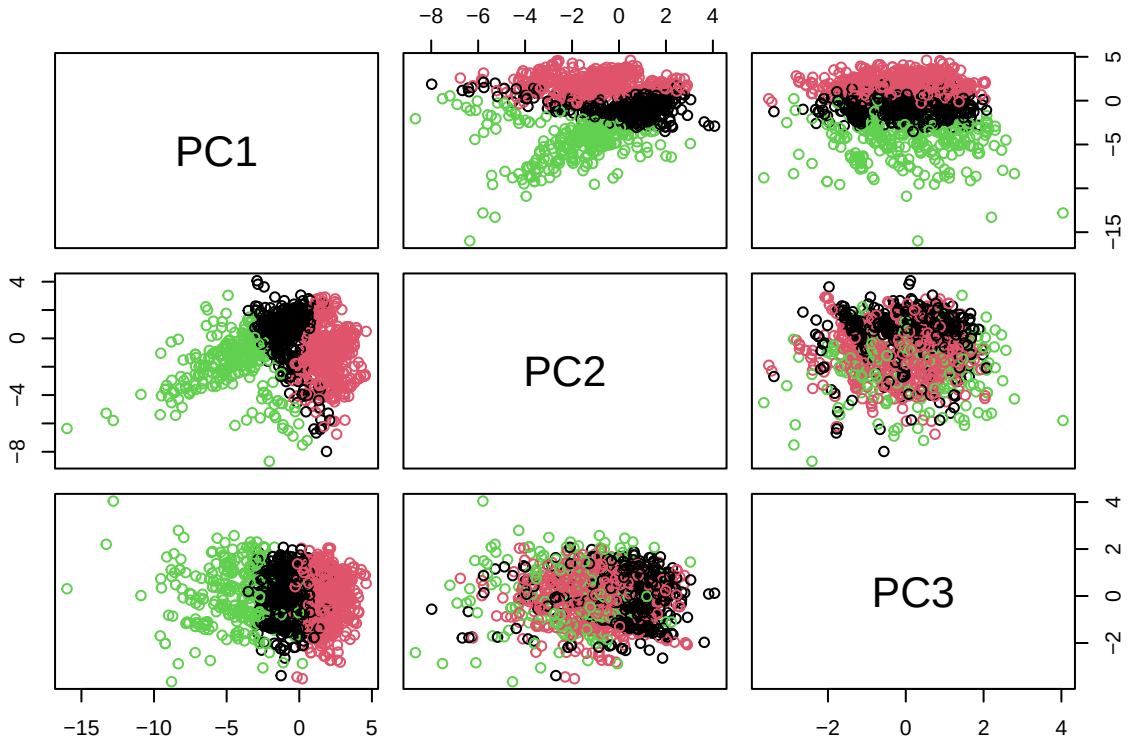


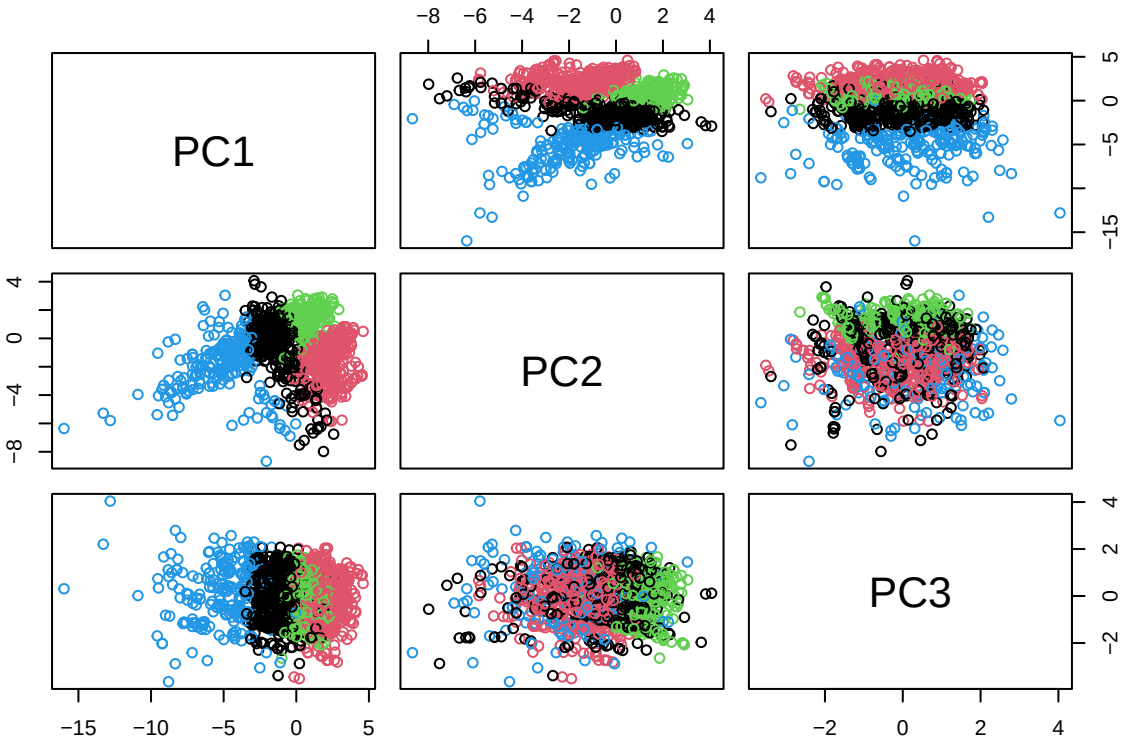

### 3.1.2 pam

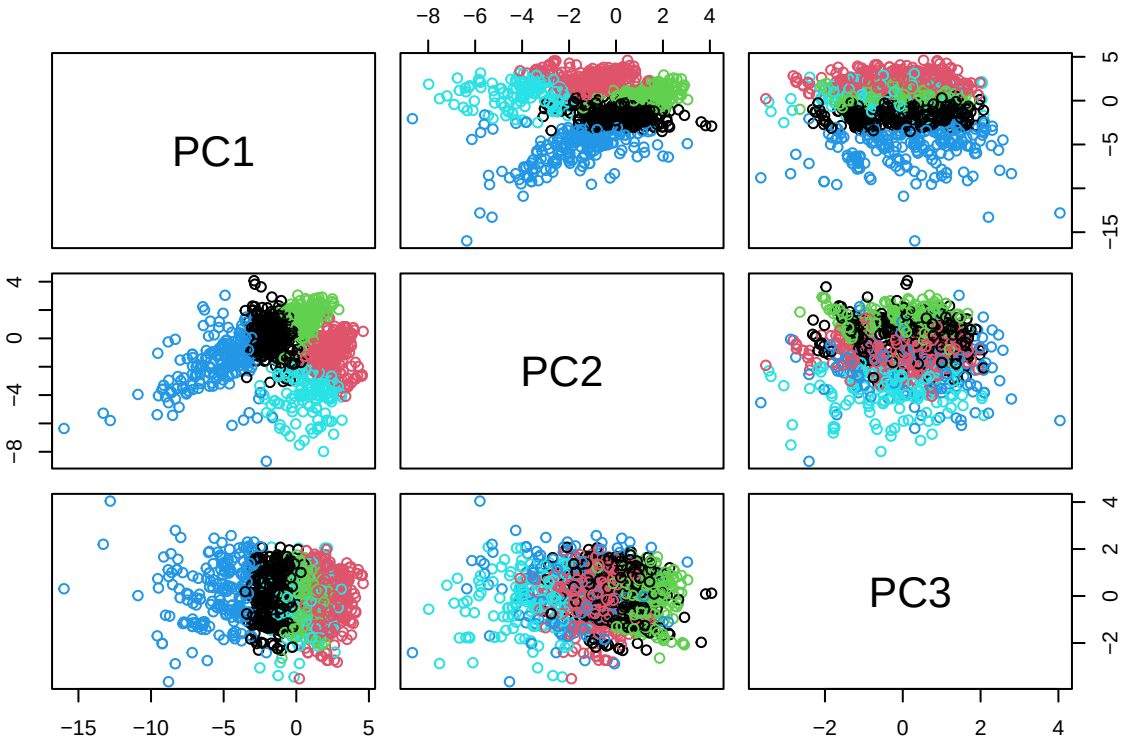
```
for (amount_centers in 1:6) {
  Xtrain.pam <-pam(Xtrain.robpca.scaled, k = amount_centers, metric="manhattan")
  pairs(Xtrain.robpca.scores, col=Xtrain.pam$clustering)
}
```

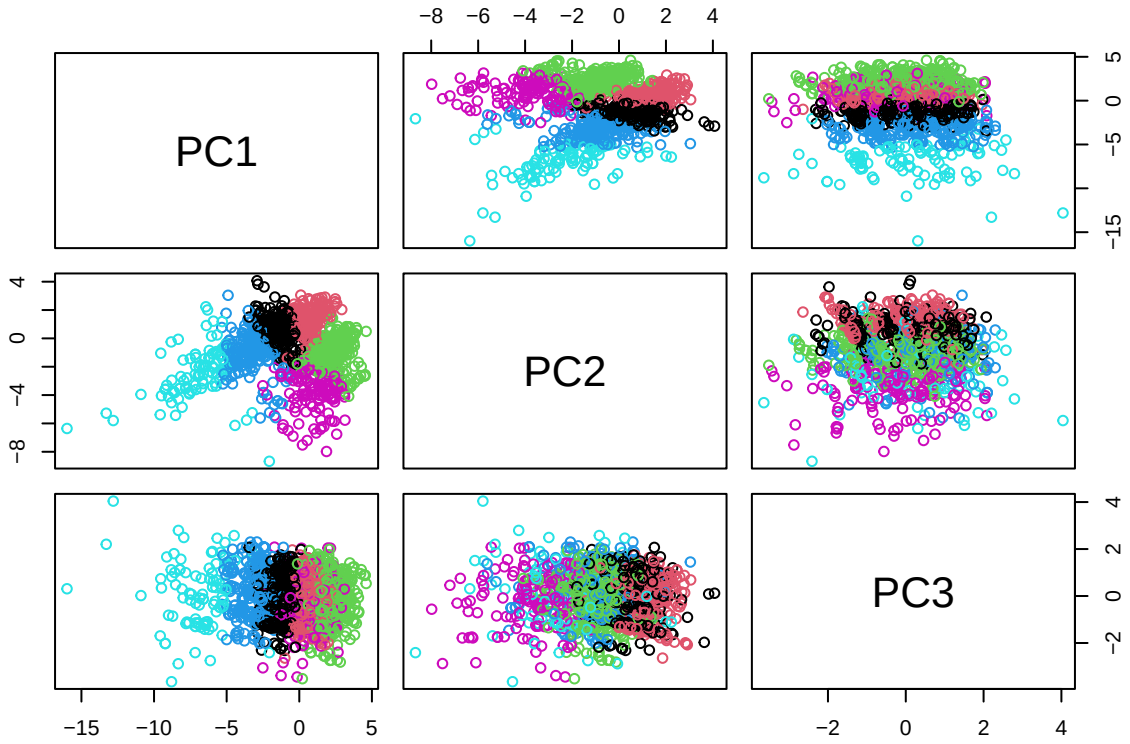

PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
−15 −5 0 5
−8 −4 0 4
−15 −10 −5 0 5
−2 0 2 4
−2 0 2 4


PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2


PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2


PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2


PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2




#### 3.1.3 conclusion

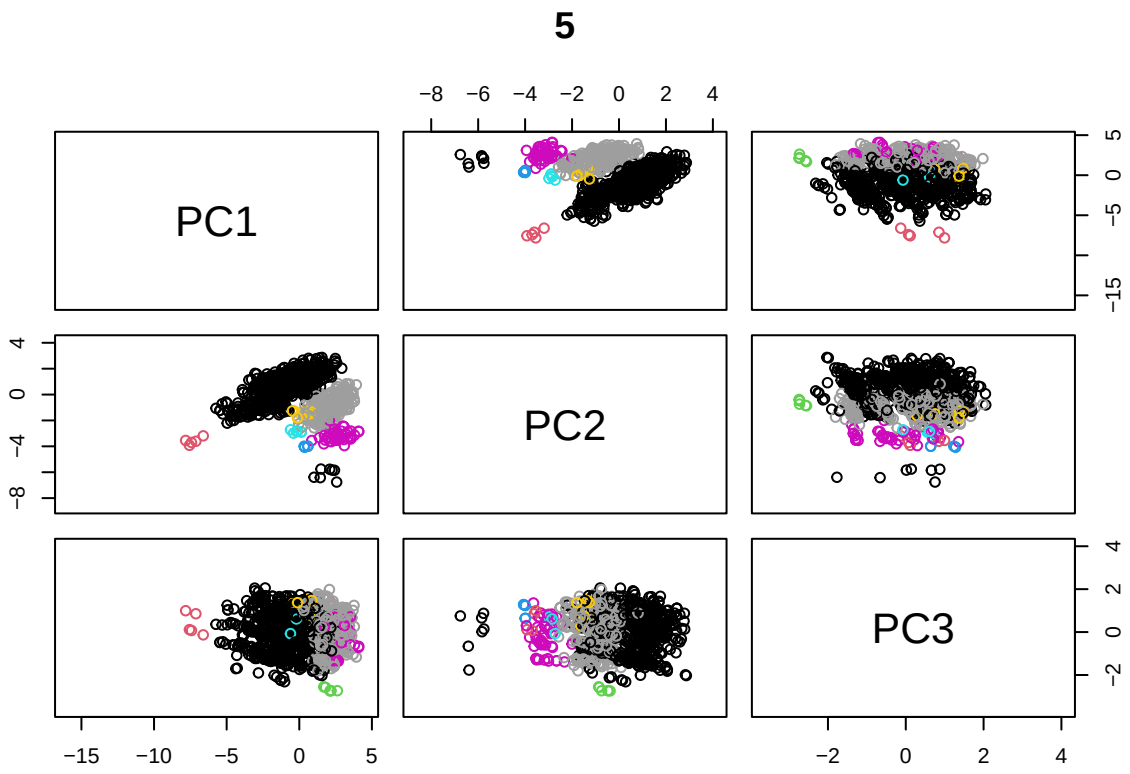
The only categorical variable you can recognize on the score plot was fuel the others should be overlapping ruling them out. Meaning neither pam or kmeans recognized any categorical variables.

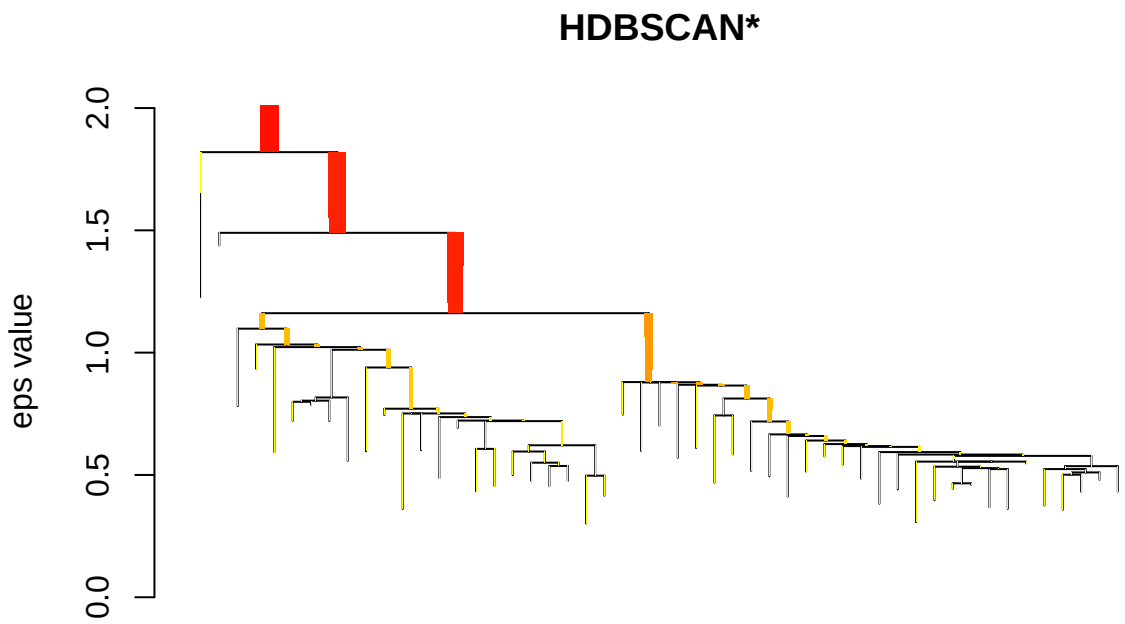
We would go with 1 cluster here.

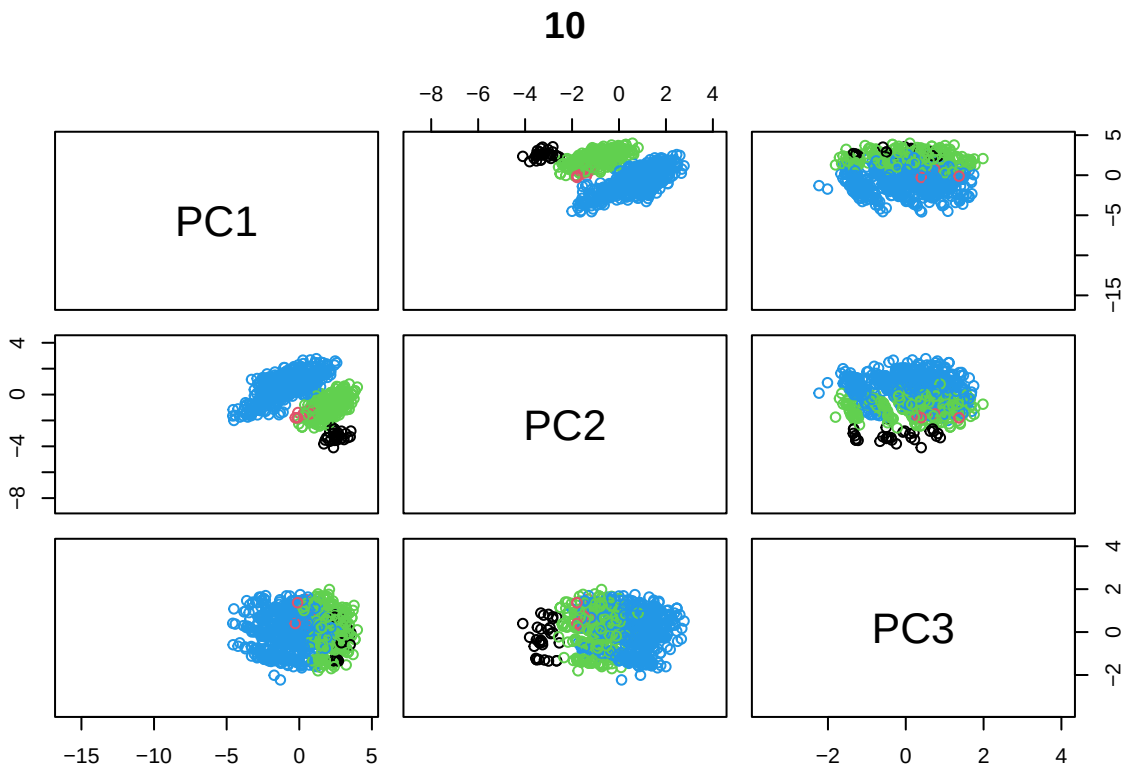
### 3.2 hierarchical clustering observations

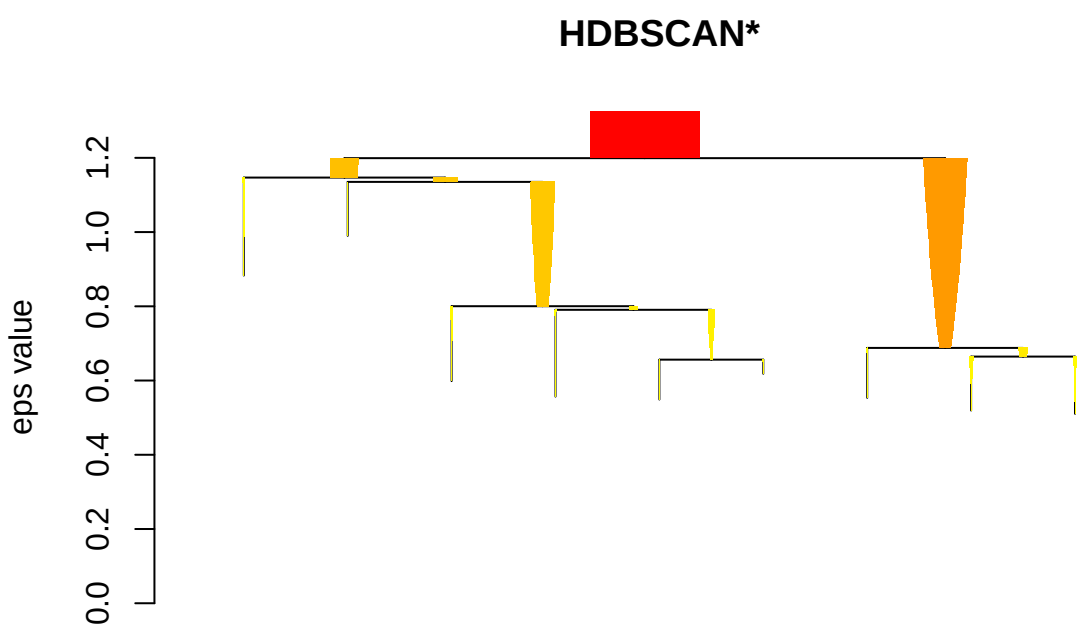
#### 3.2.1 hdbscan

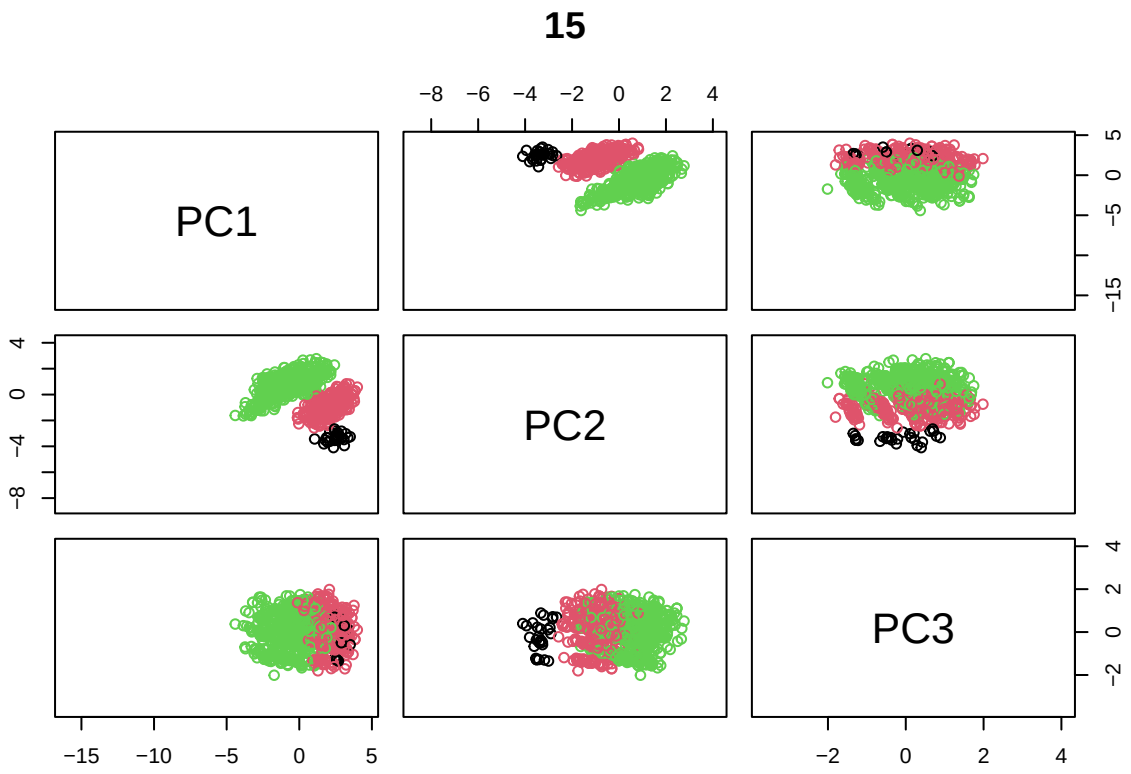
```
for (pts in c(5,10,15,20,25)) {
  Xtrain.hdbscan <- hdbscan(Xtrain.robpca.scaled,minPts = pts)
  pairs(Xtrain.robpca.scores, col=Xtrain.hdbscan$cluster, main =pts)
  plot(Xtrain.hdbscan)
}
```

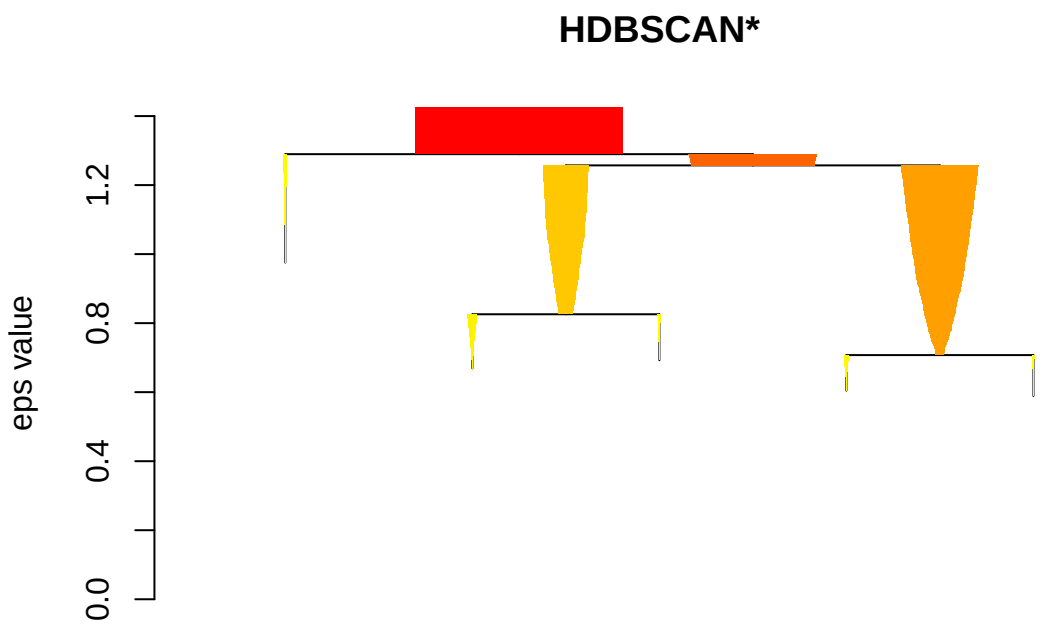

5
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
−15 −10 −5 0 5
−2 0 2 4

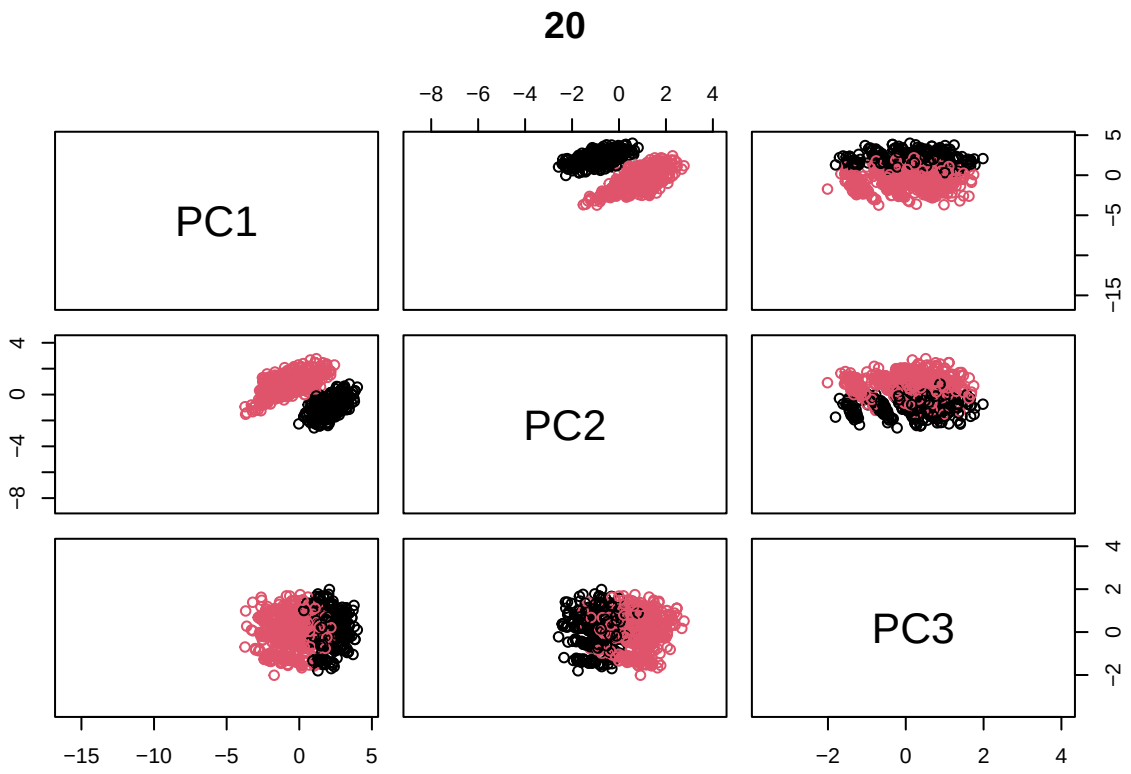

HDBSCAN*
eps value
0.0
0.5
1.0
1.5
2.0

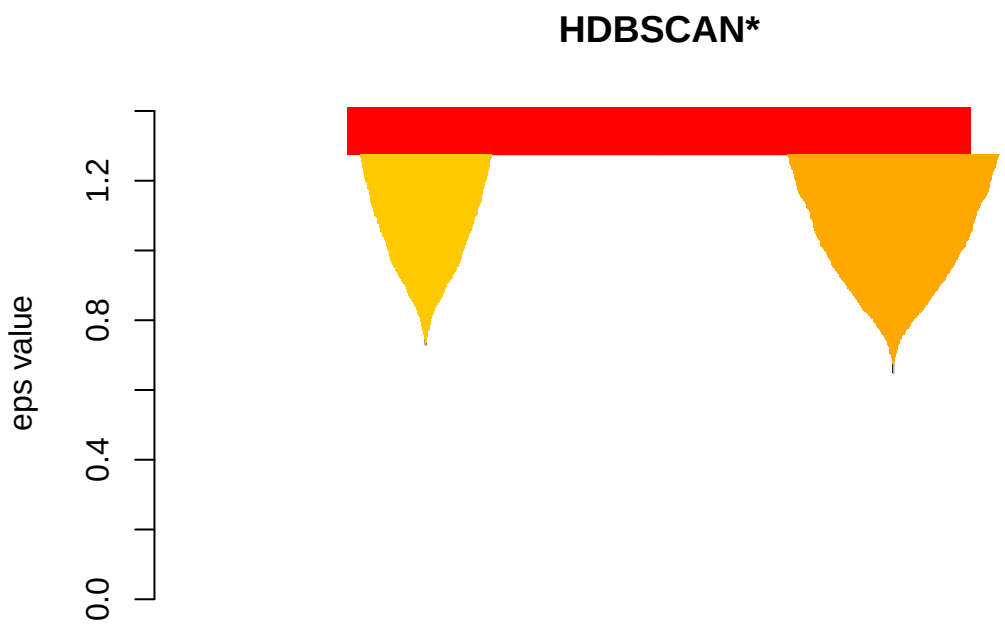

10
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2

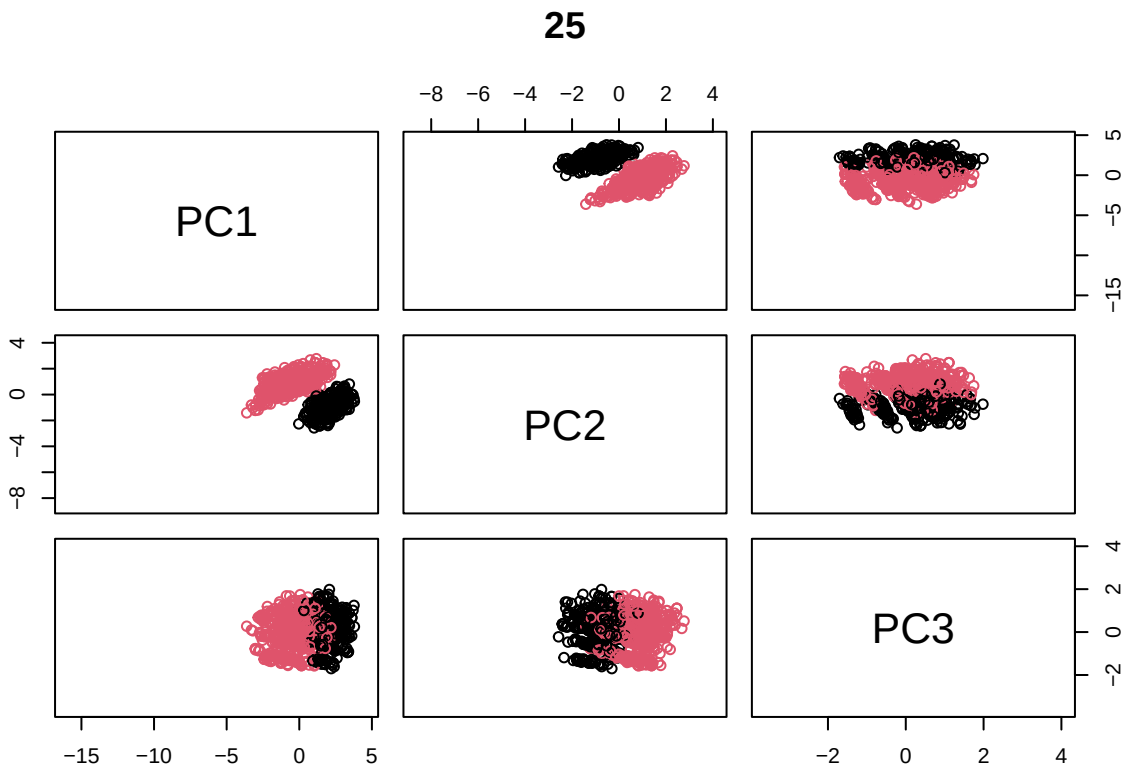

HDBSCAN*
eps value
0.0
0.2
0.4
0.6
0.8
1.0
1.2

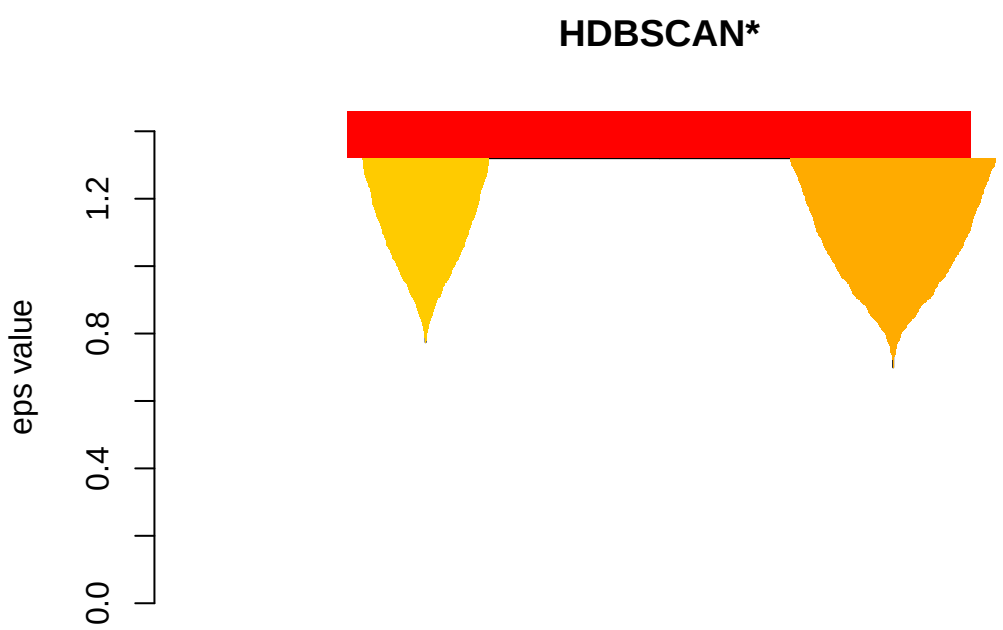

15
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
4 2 0 −2
−15 −10 −5 0 5
−2 0 2 4


HDBSCAN*
eps value
0.0
0.4
0.8
1.2


20
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
4 2 0 −2
−15 −10 −5 0 5
−2 0 2 4

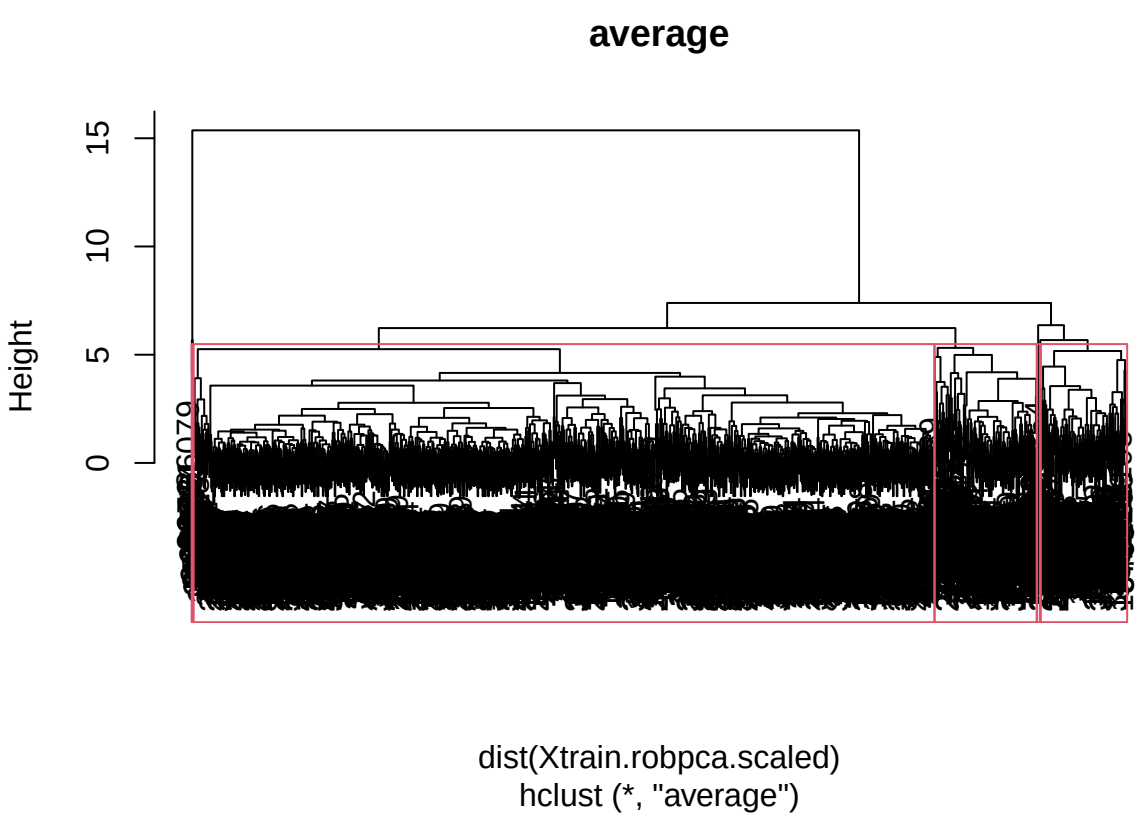

HDBSCAN*
eps value
0.0
0.4
0.8
1.2

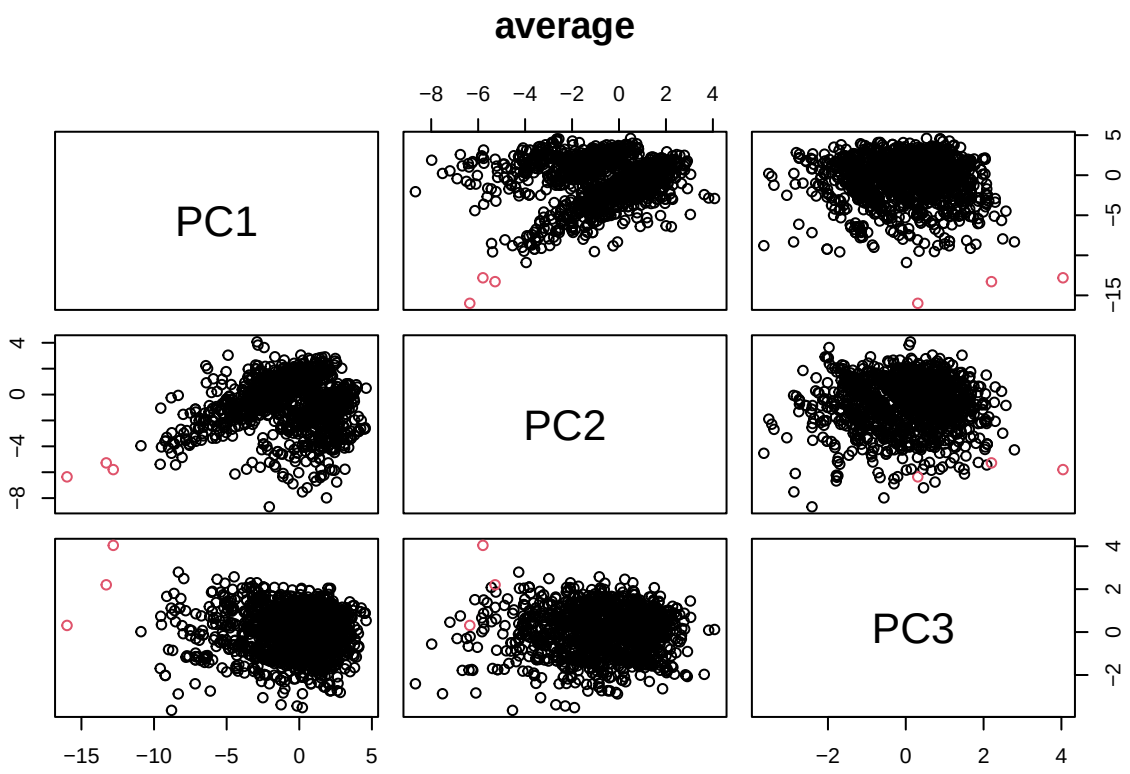

25
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
−15 −10 −5 0 5
−2 0 2 4
−8 −4 0 4
−15 −5 0 5
−2 0 2 4

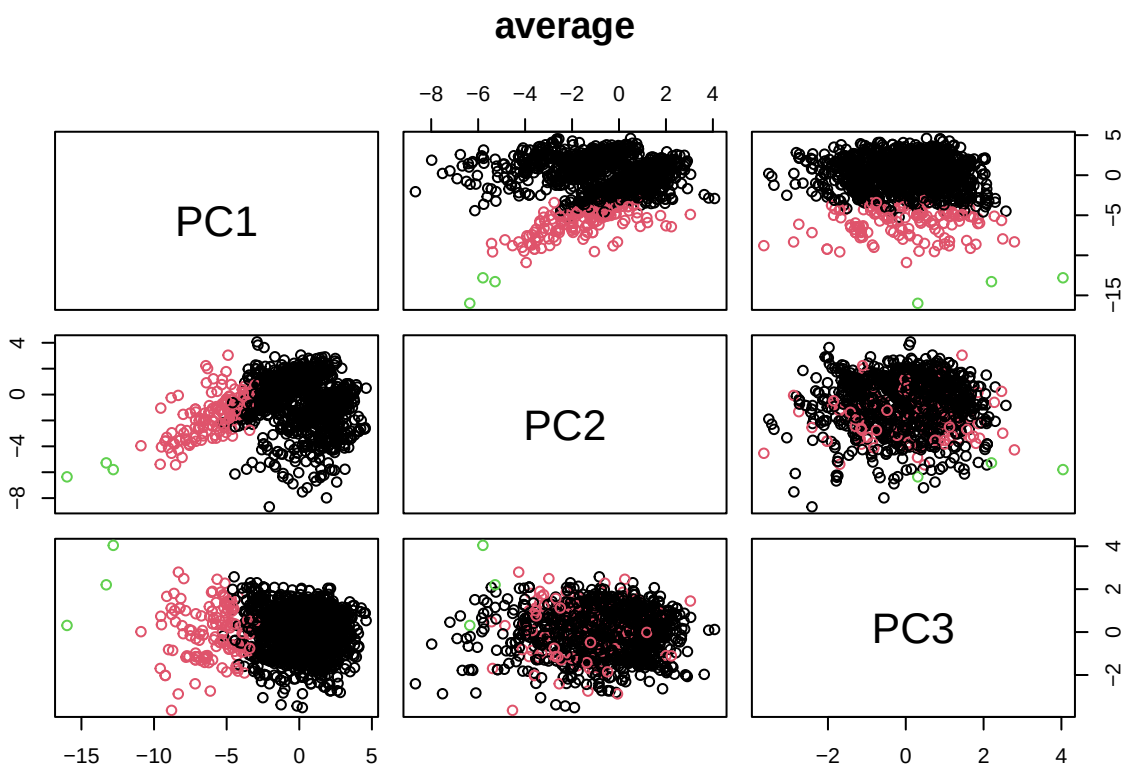


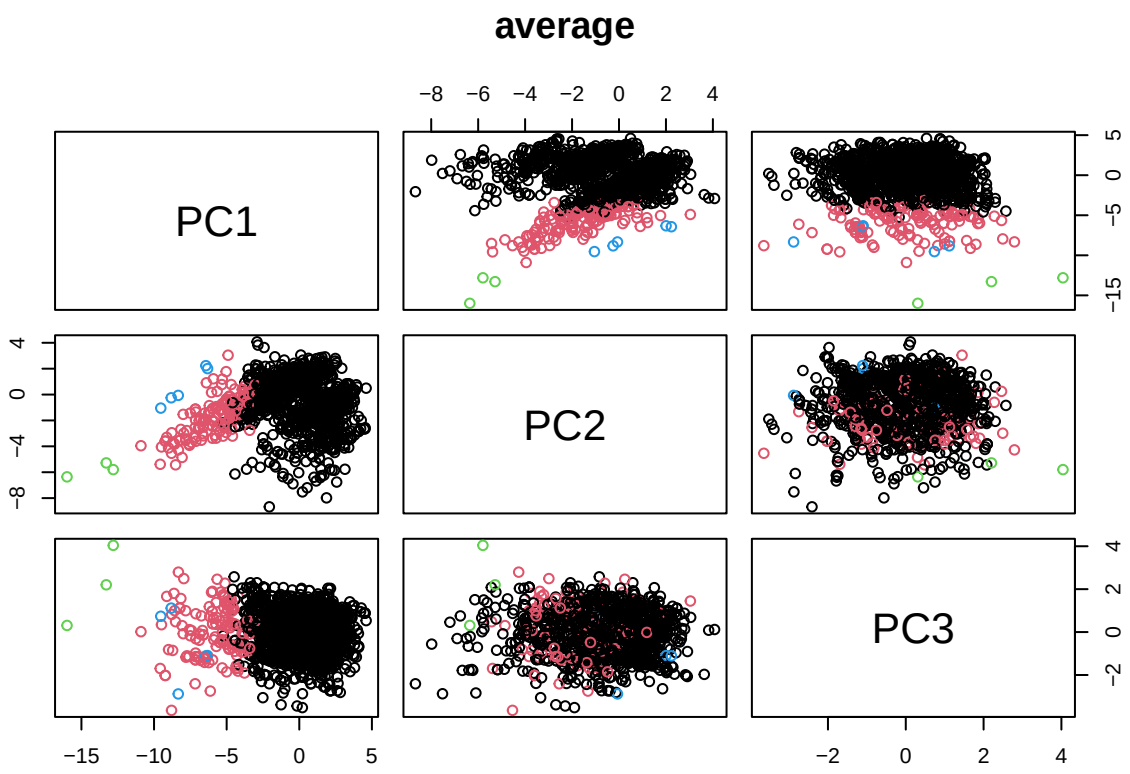

### 3.2.2 hclust

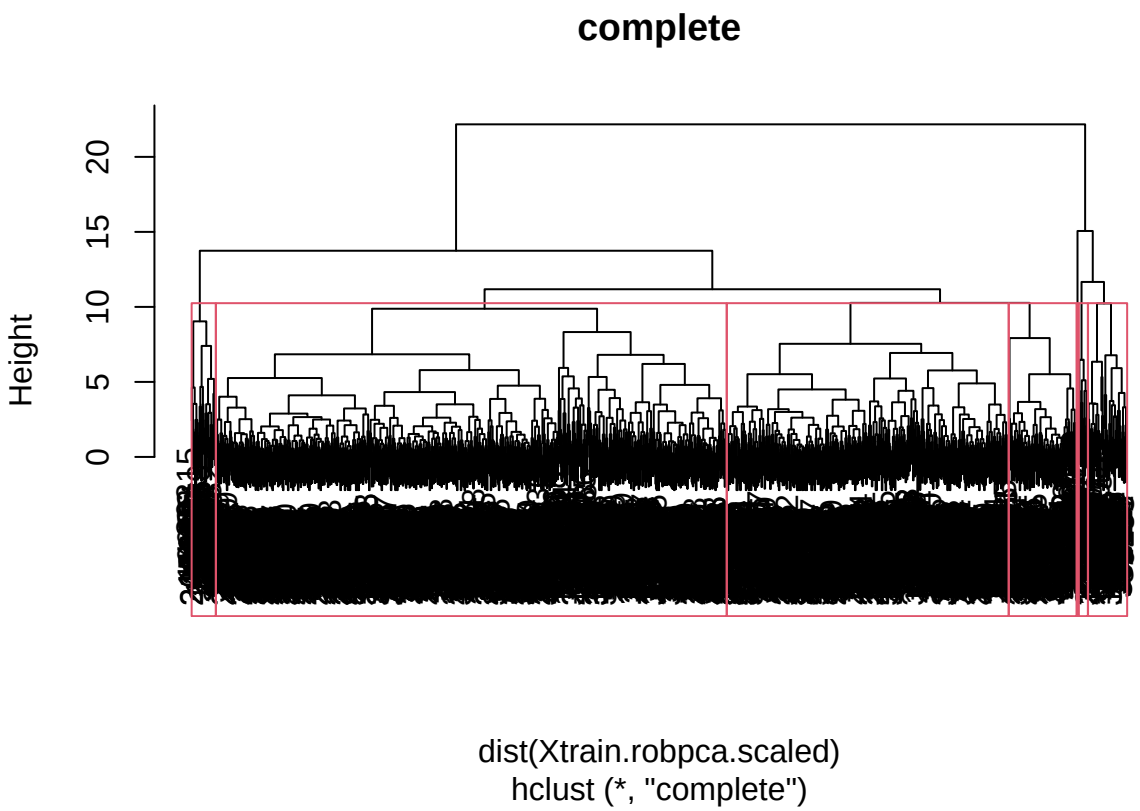
```
for (mt in c("average","complete","ward.D")) {
  Xtrain.hclust <- hclust(dist(Xtrain.robpca.scaled),method =mt)
  plot(Xtrain.hclust, main = mt)
  rect.hclust(Xtrain.hclust, k = 7)
  for (amount_cluster in c(2,3,4)){
  pairs(Xtrain.robpca.scores, col=cutree(Xtrain.hclust,k=amount_cluster), main = mt)
  }
}
```

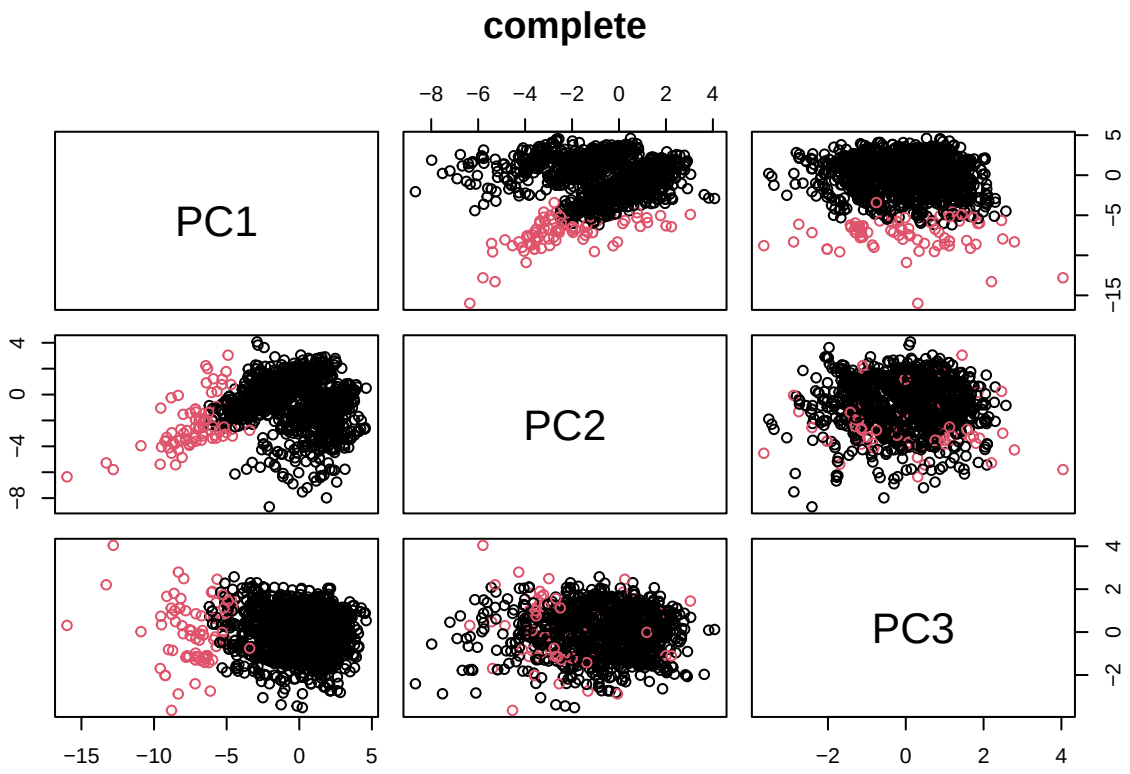

average
Height
0
5
10
15
dist(Xtrain.robpca.scaled)
hclust (*, "average")

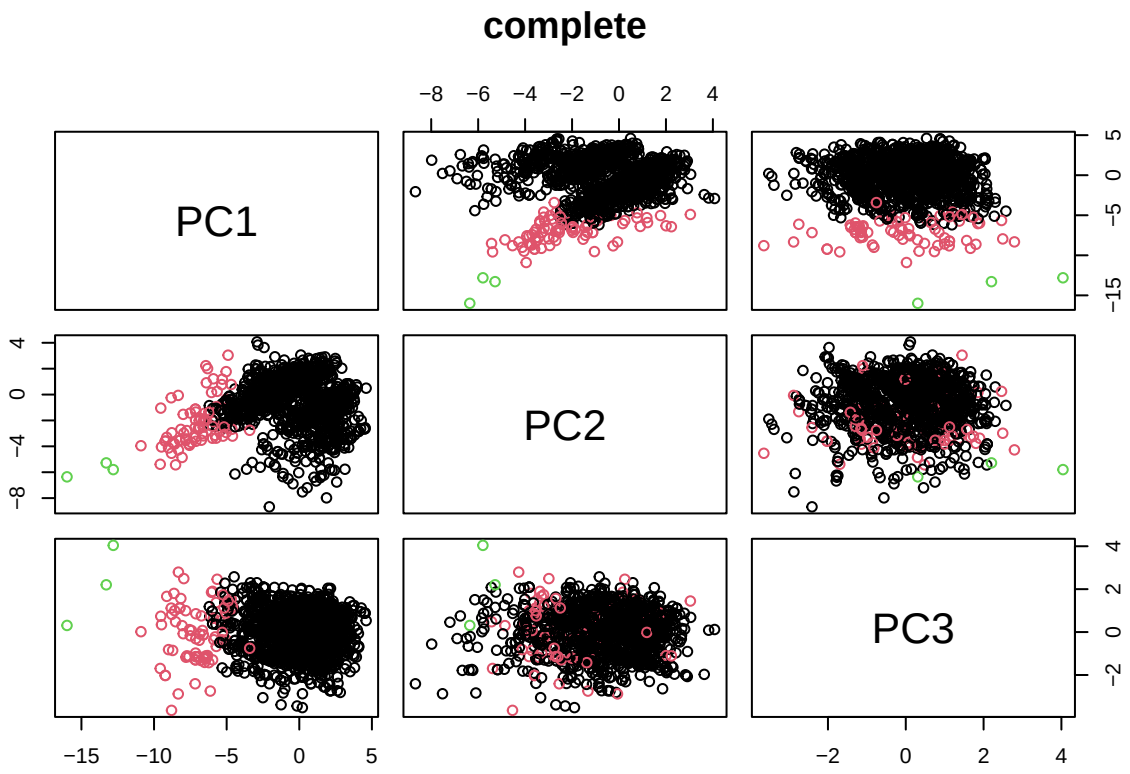

average
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
−15 −10 −5 0 5
−2 0 2 4

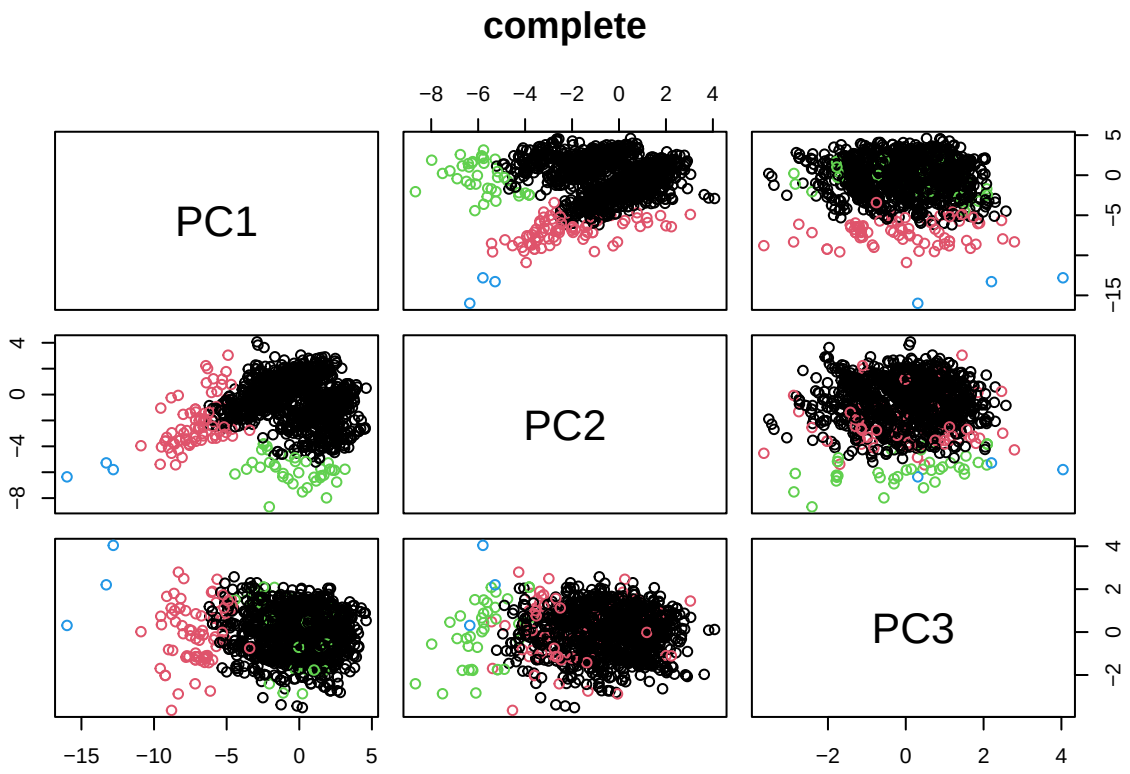

average
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
−15 −10 −5 0 5
−2 0 2 4
−8 −4 0 4
−15 −5 0 5
−2 0 2 4


average
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
−15 −10 −5 0 5
−2 0 2 4

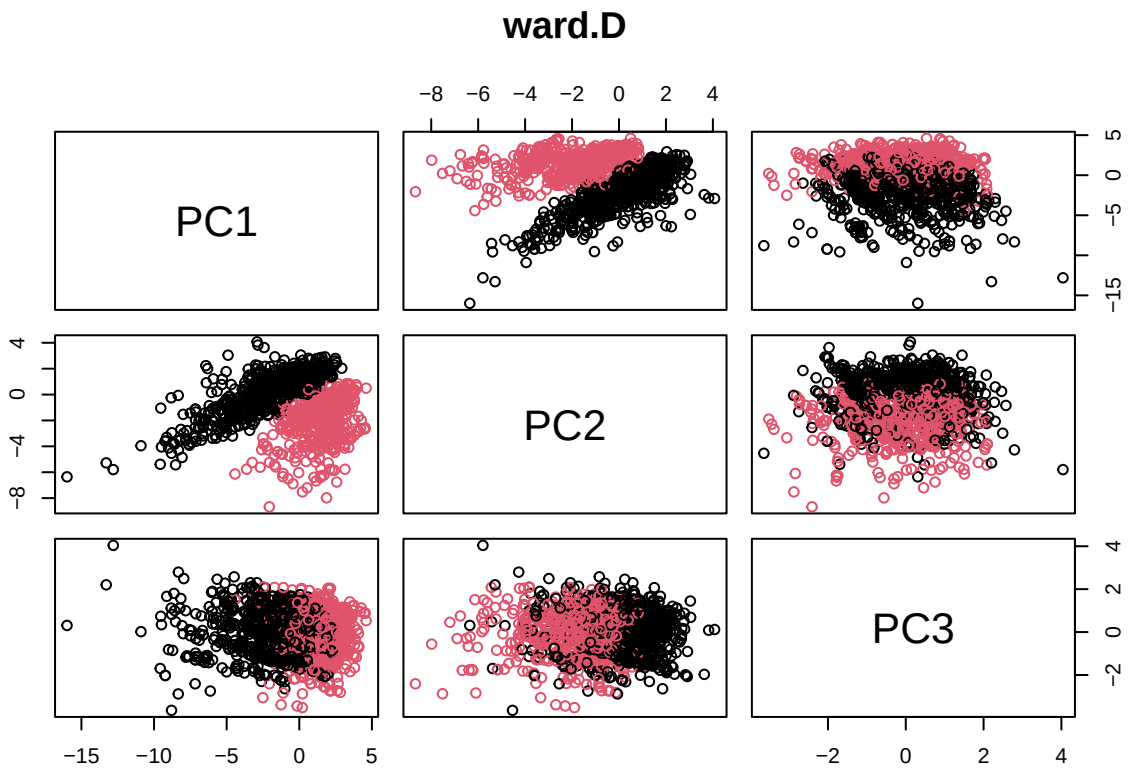

complete
Height
0
5
10
15
20
dist(Xtrain.robpca.scaled)
hclust (*, "complete")

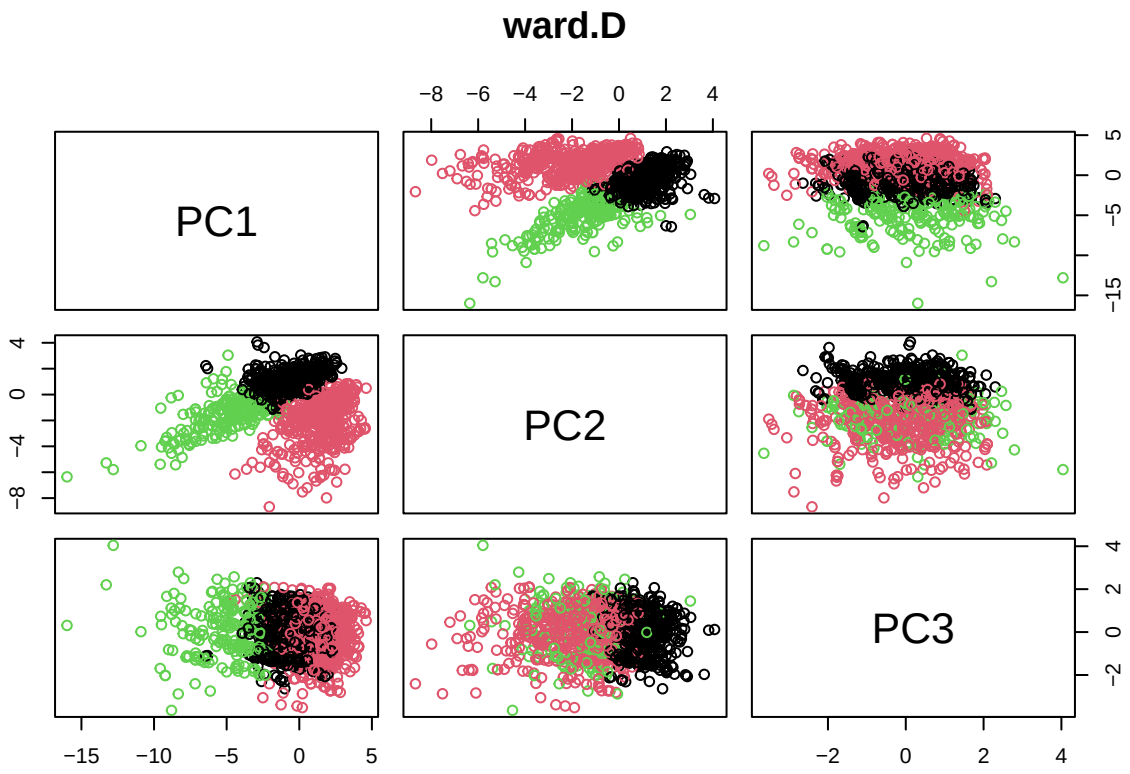

complete
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2

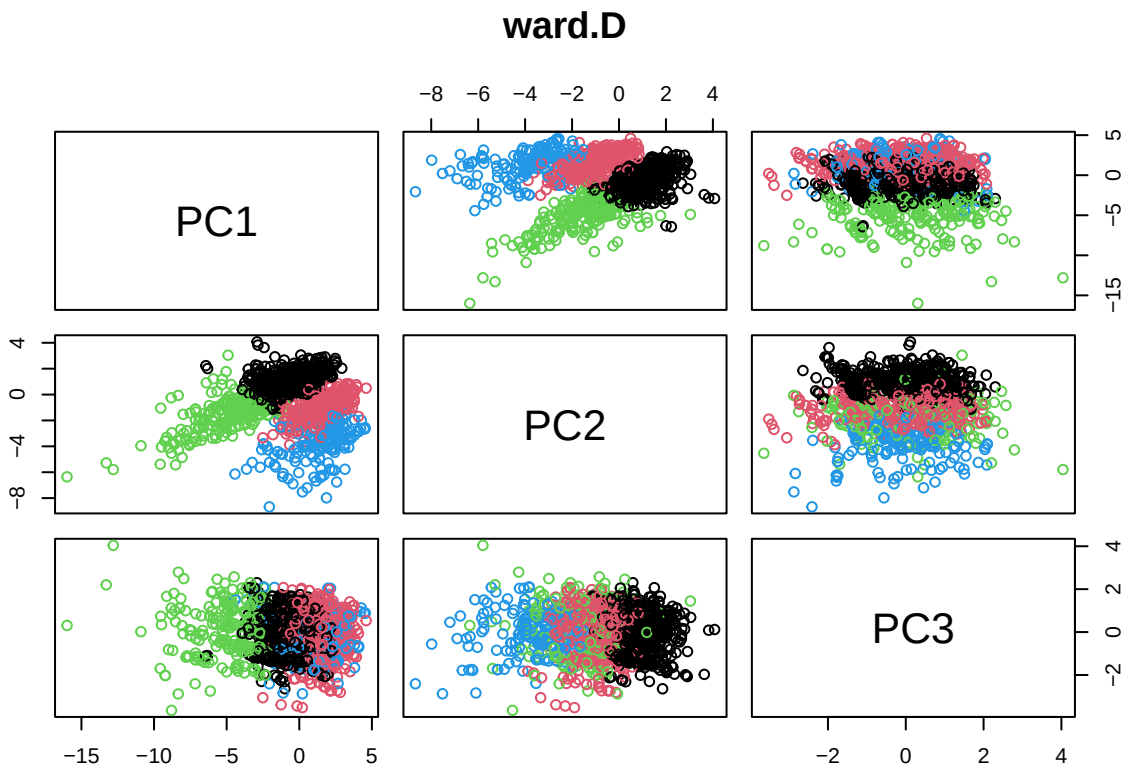

complete
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2


complete
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
4 2 0 −2
−15 −10 −5 0 5
−2 0 2 4


ward.D
Height
0
400
800
1200
dist(Xtrain.robpca.scaled)
hclust (*, "ward.D")

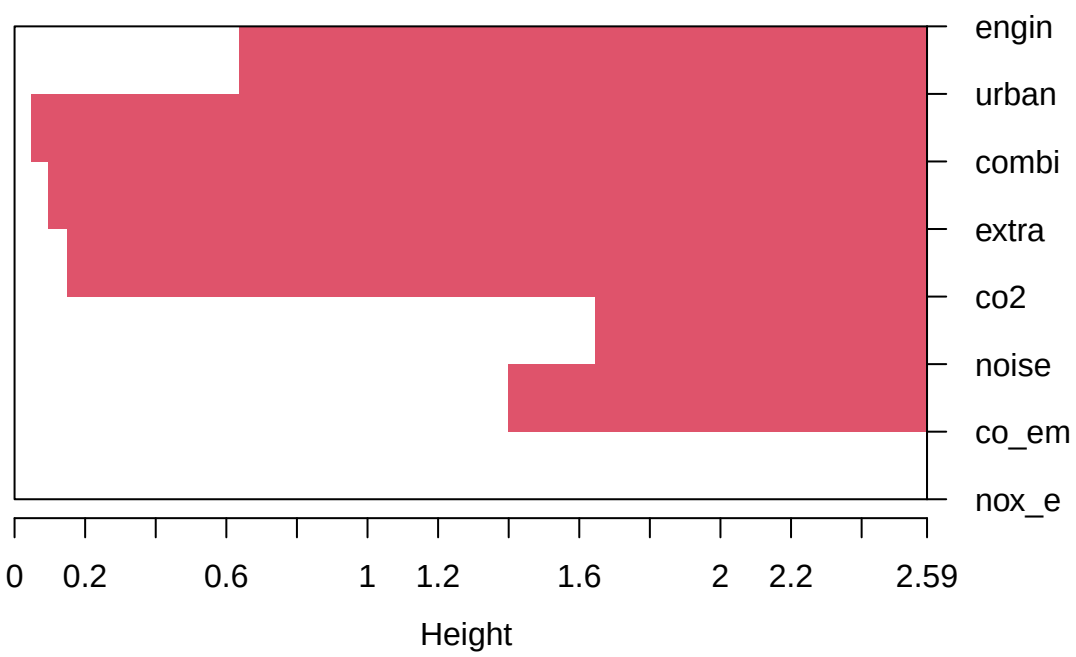

ward.D
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
5 0 −5 −15
4 0 −4 −8
−15 −10 −5 0 5
−2 0 2 4
4 2 0 −2

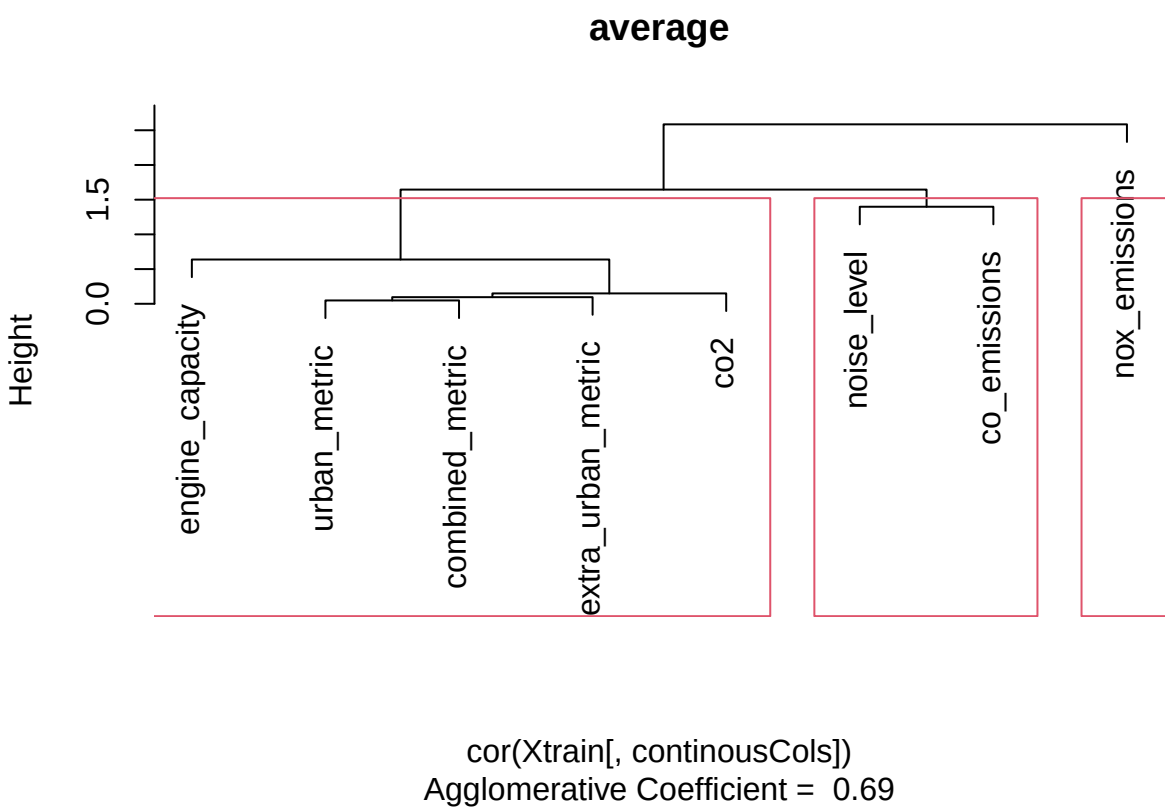

ward.D
PC1
PC2
PC3
−8 −6 −4 −2 0 2 4
−15 −10 −5 0 5
−2 0 2 4



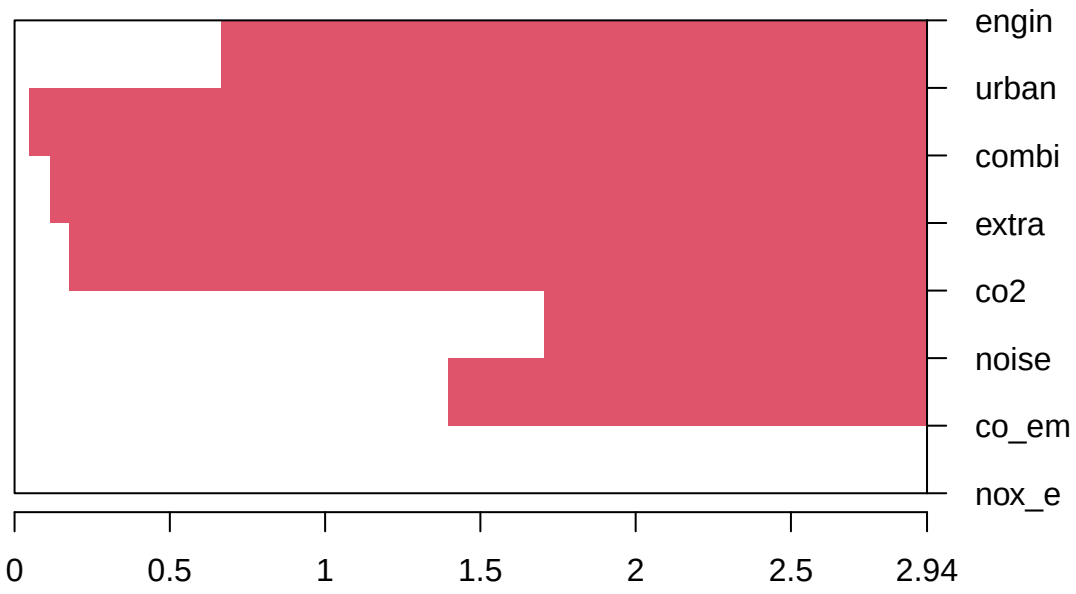

### 3.3 hierarchical clustering variables

```
for (mt in c("average","complete","ward")) {
  Xtrain.var.agnes <- agnes(cor(Xtrain[,continousCols]), method = mt)
  plot(Xtrain.var.agnes, main = mt)
  rect.hclust(Xtrain.var.agnes, k = 3)
}
```

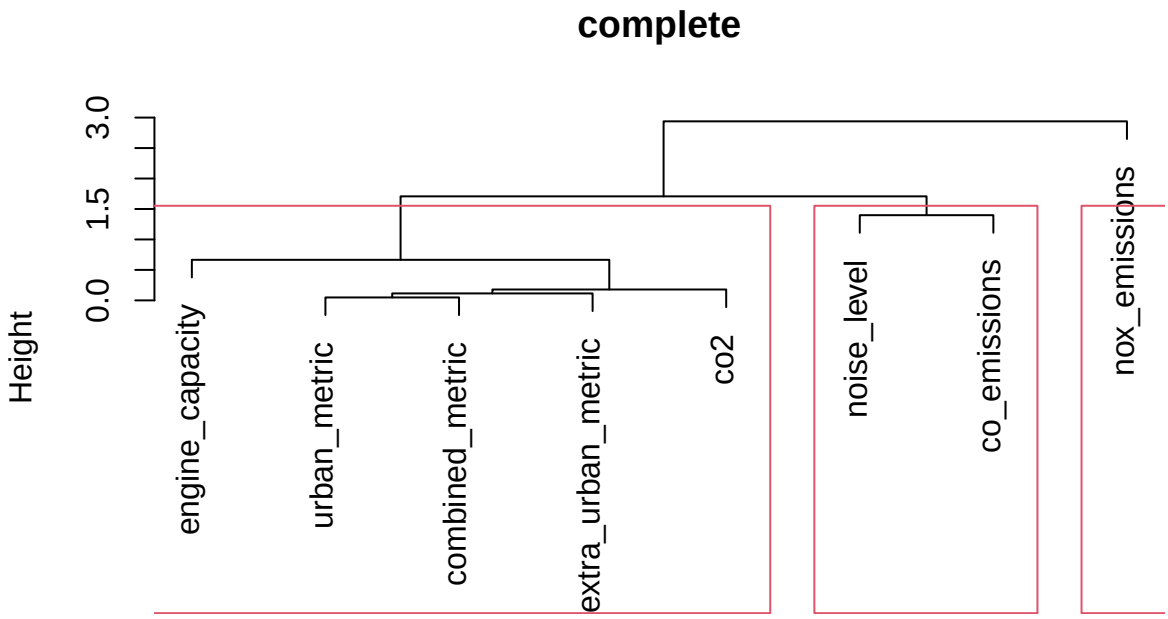

average
engin
urban
combi
extra
co2
noise
co_em
nox_e
0
0.2
0.6
1
1.2
1.6
2
2.2
2.59
Height
Agglomerative Coefficient = 0.69

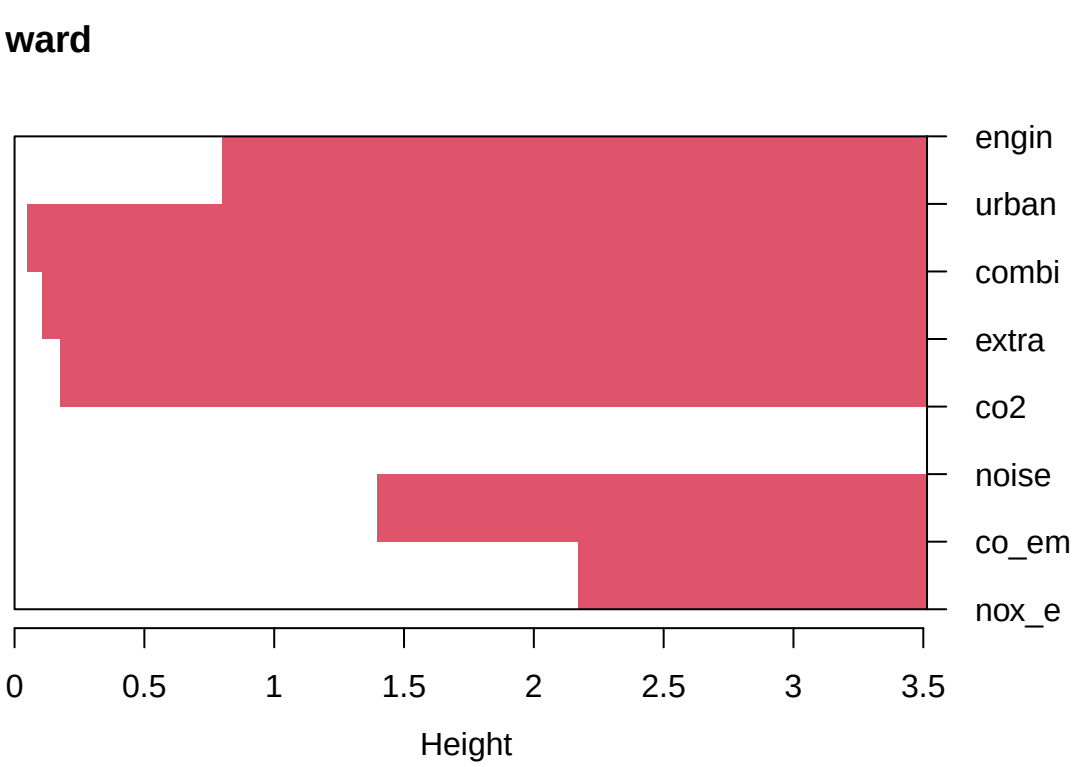

average
Height
0.0
1.5
engine_capacity
urban_metric
combined_metric
extra_urban_metric
co2
noise_level
co_emissions
nox_emissions
cor(Xtrain[, continousCols])
Agglomerative Coefficient = 0.69

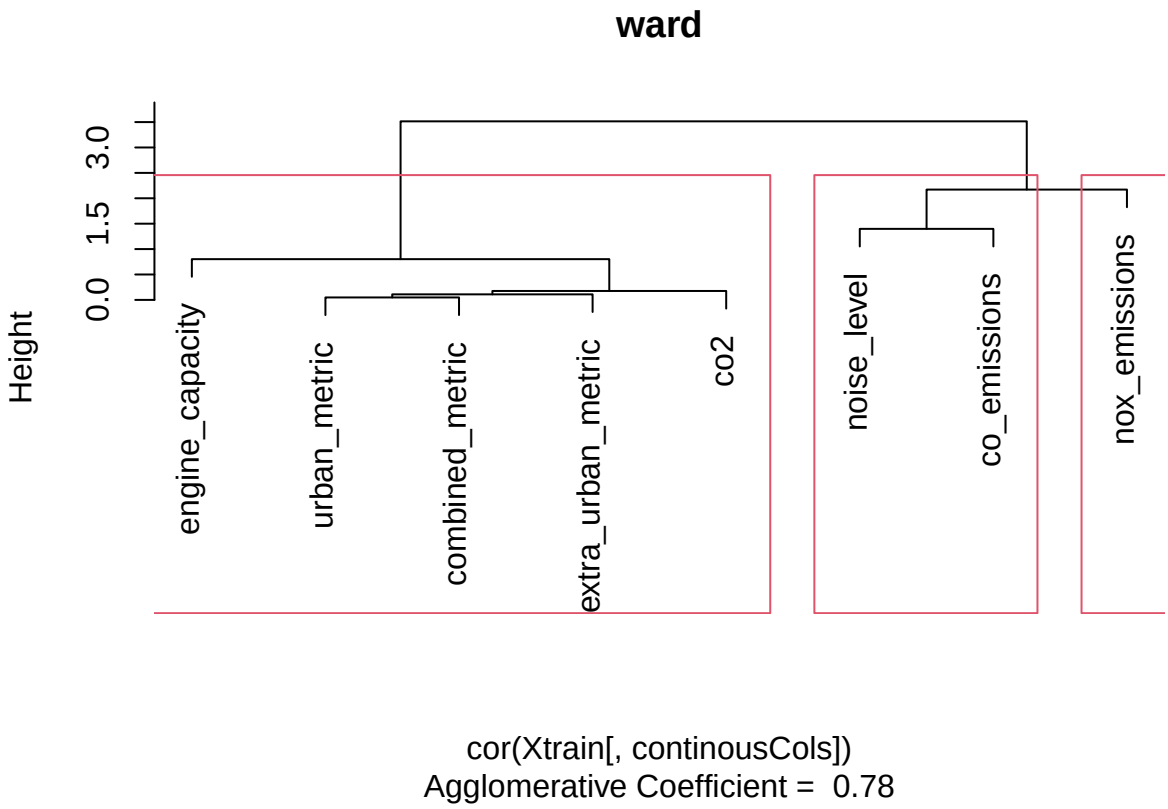
**complete**




Agglomerative Coefficient = 0.71


complete
Height
0.0
1.5
3.0
engine_capacity
urban_metric
combined_metric
extra_urban_metric
co2
noise_level
co_emissions
nox_emissions
cor(Xtrain[, continousCols])
Agglomerative Coefficient = 0.71

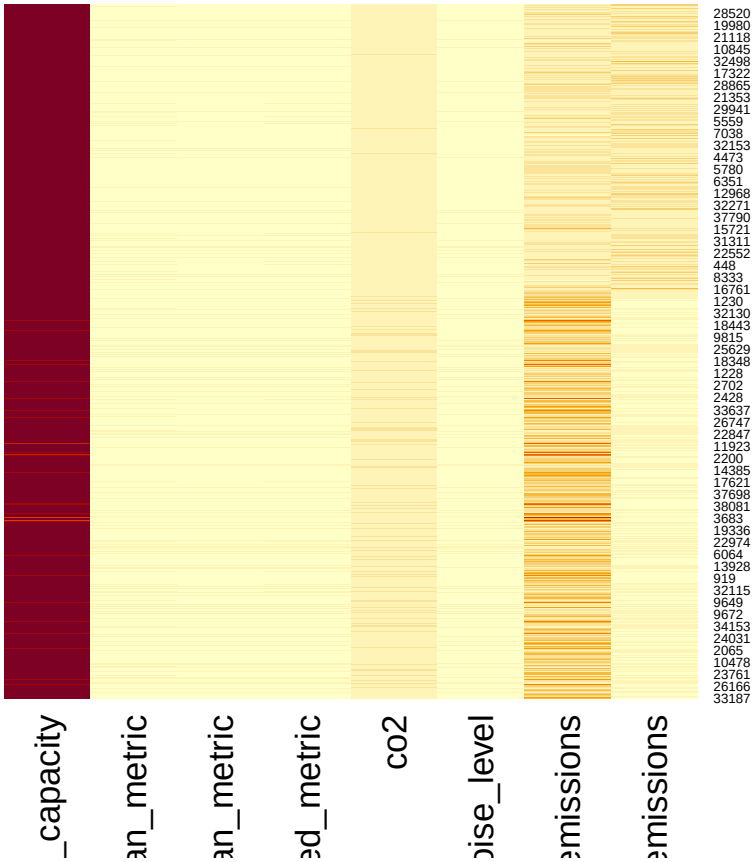

ward
engin
urban
combi
extra
co2
noise
co_em
nox_e
0
0.5
1
1.5
2
2.5
3
3.5
Height
Agglomerative Coefficient = 0.78



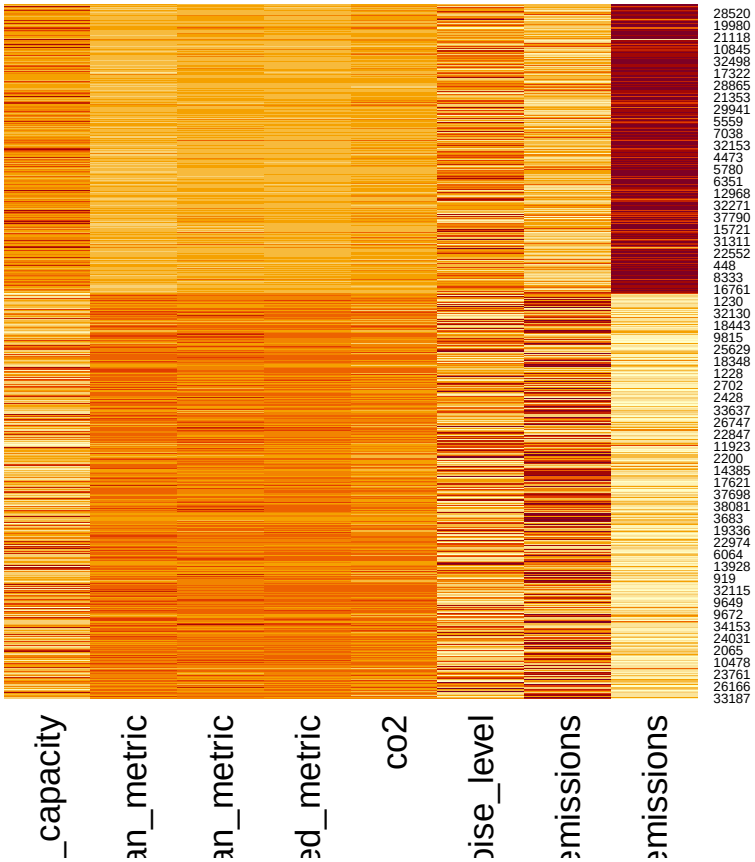

### 3.4 clustering that corresponds to fuel

Hclust with ward found clustering that correspond with fuel type.

```
Xtrain.hclust.ward <-cutree(hclust(dist(Xtrain.robpca.scaled), method = "ward.D"), k = 2)
Xtrain.var.agnes.ward <- cutree(agnes(cor(Xtrain[,continousCols]), method = "ward"),k=3)

Xtrain.sort_cluster <- Xtrain[,continousCols][order(Xtrain.hclust.ward),order(Xtrain.var.agnes.ward)]
Xtrain.sort_cluster.scaled <- Xtrain.robpca.scaled[order(Xtrain.hclust.ward),order(Xtrain.var.agnes.ward)]

heatmap(data.matrix(Xtrain.sort_cluster),Rowv = NA,Colv = NA )
```




```
heatmap(data.matrix(Xtrain.sort_cluster.scaled),Rowv = NA,Colv = NA)
```

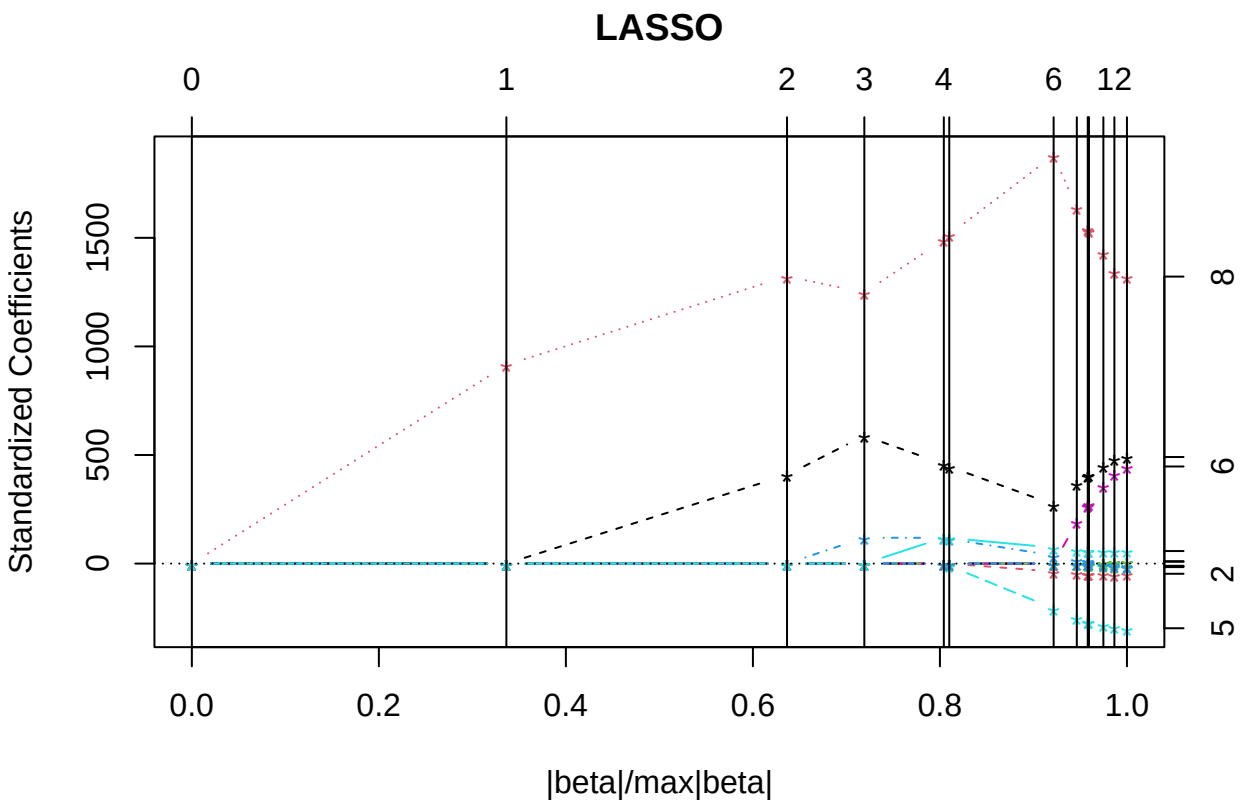



On the heatmap you can clearly see the conditional distribution on fuel and they are similar for similar variables.

# 4 Linear regression

## 4.1 linear model

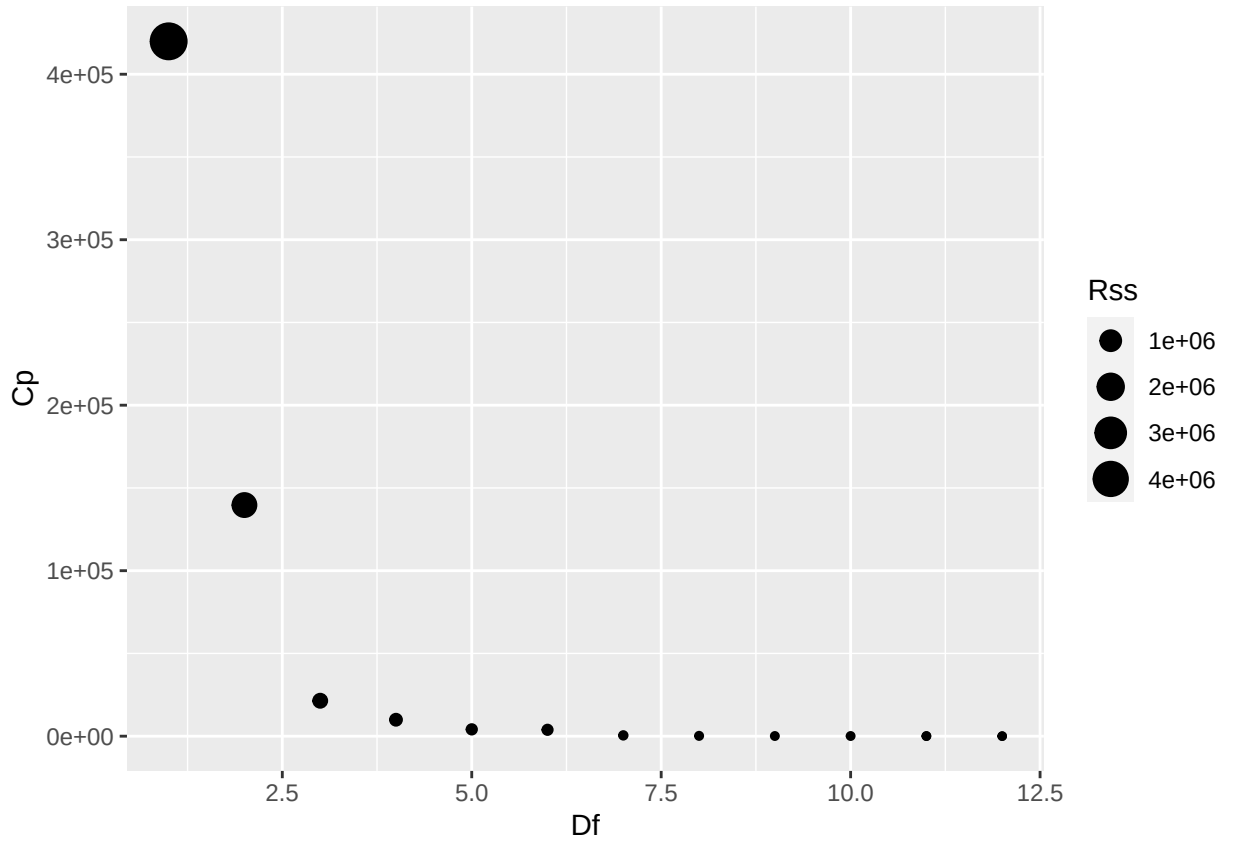
```
X <- model.matrix(co2 ~ ., data=subset(Xtrain, select = -inTrain))[,-1]
y <- Xtrain$co2
Xtrain.co2.lars.lasso <- lars(X, y, type = "lasso", intercept = TRUE, normalize = TRUE)

plot(Xtrain.co2.lars.lasso)
```




```
names(Xtrain.co2.lars.lasso)
```

```
##  [1] "call"       "type"       "df"         "lambda"     "R2"
##  [6] "RSS"        "Cp"         "actions"    "entry"      "Gamrat"
## [11] "arc.length" "Gram"       "beta"       "mu"         "normx"
## [16] "meanx"
```

```
ggplot(summary(Xtrain.co2.lars.lasso), aes(x = Df,y = Cp, size = Rss )) +geom_point()
```




```
print(Xtrain.co2.lars.lasso)
```

```
##
## Call:
## lars(x = X, y = y, type = "lasso", normalize = TRUE, intercept = TRUE)
## R-squared: 0.996
## Sequence of LASSO moves:
##      combined_metric extra_urban_metric engine_capacity nox_emissions
## Var                8                  7               4            11
## Step               1                  2               3             4
##      fuel_typePetrol euro_standard5 urban_metric transmission_typeManual
## Var                5              2           6                       3
## Step               5              6           7                       8
##      noise_level euro_standard4 co_emissions engine_capacity engine_capacity
## Var            9              1           10              -4               4
## Step           9             10           11              12              13
```

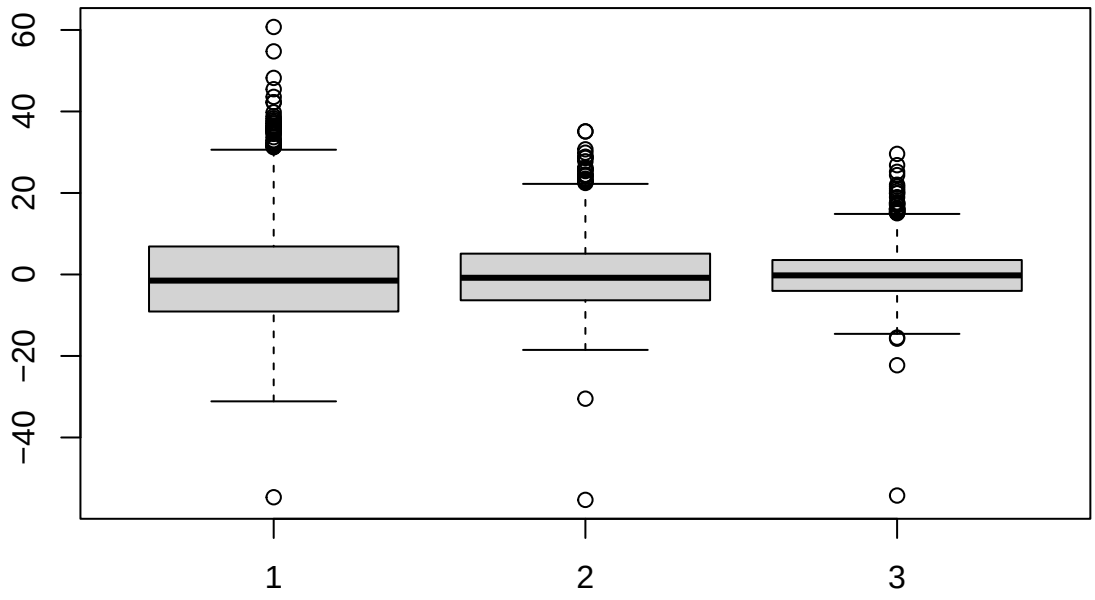
```
Xtrain.co2.lars.lasso.coef = coef(Xtrain.co2.lars.lasso)[4,]
Xtrain.co2.lars.lasso.coef
```

```
##          euro_standard4          euro_standard5 transmission_typeManual
##             0.000000000             0.000000000             0.000000000
##         engine_capacity        fuel_typePetrol            urban_metric
##             0.003632645             0.000000000             0.000000000
##      extra_urban_metric        combined_metric            noise_level
##             9.416196873            13.495366978             0.000000000
##            co_emissions          nox_emissions
```

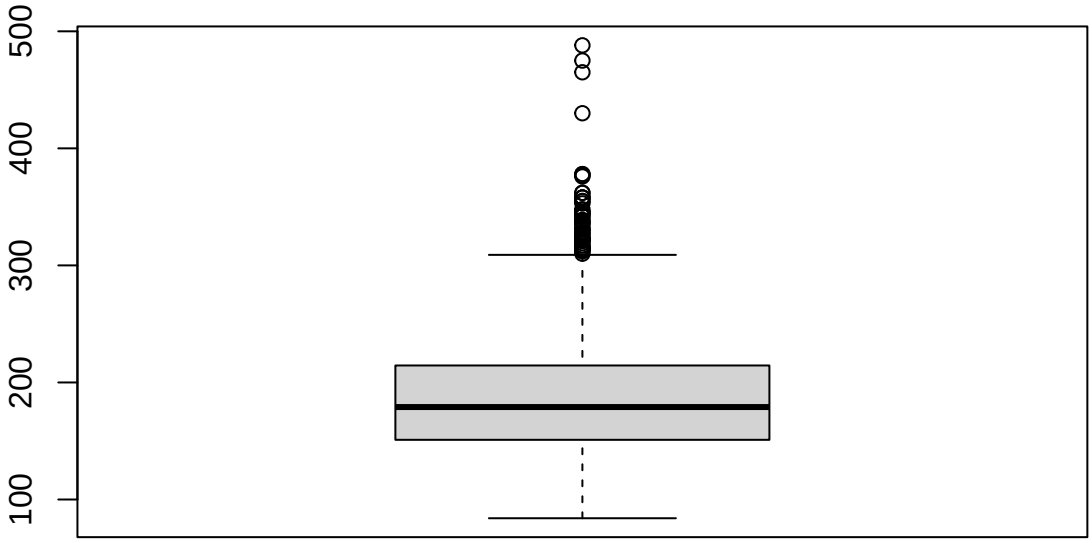
```
##             0.000000000             0.000000000
```

### 4.2 assumptions linear model

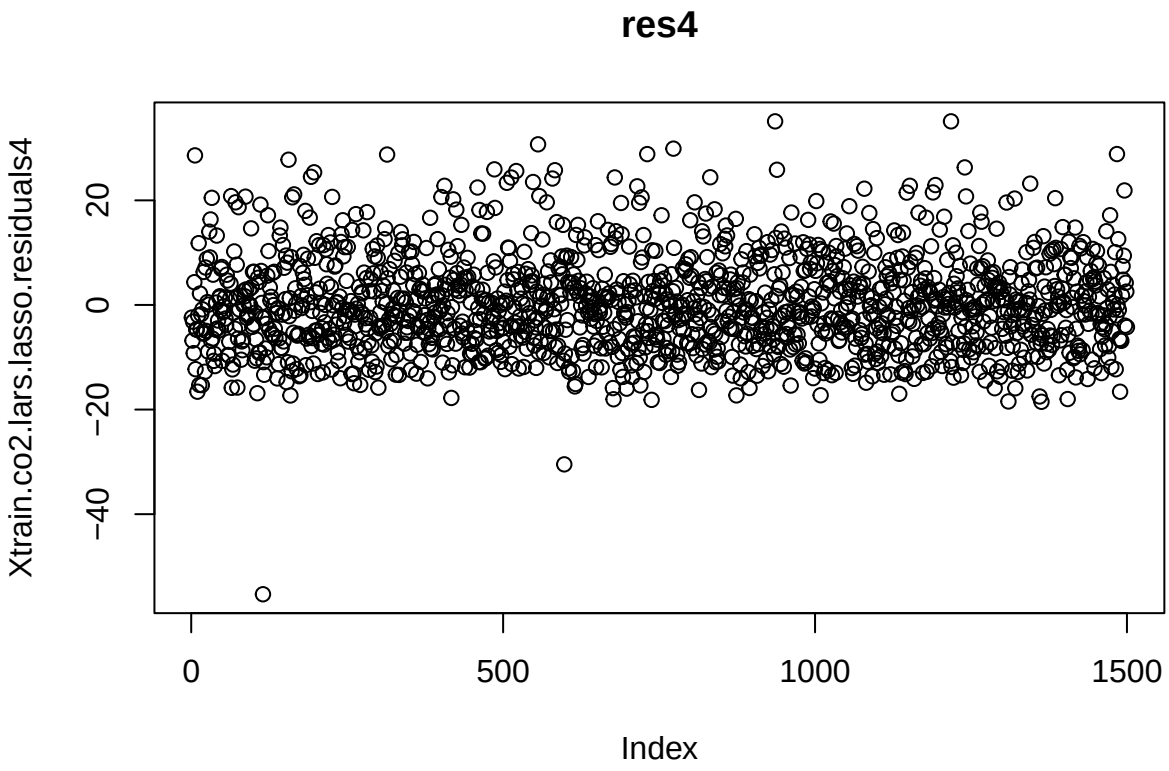
```
Xtrain.co2.lars.lasso.residuals3 <- y  - predict(Xtrain.co2.lars.lasso,X,3)$fit
Xtrain.co2.lars.lasso.residuals4 <- y  - predict(Xtrain.co2.lars.lasso,X,4)$fit
Xtrain.co2.lars.lasso.residuals5 <- y  - predict(Xtrain.co2.lars.lasso,X,5)$fit
boxplot(Xtrain.co2.lars.lasso.residuals3,
        Xtrain.co2.lars.lasso.residuals4,
        Xtrain.co2.lars.lasso.residuals5)
```



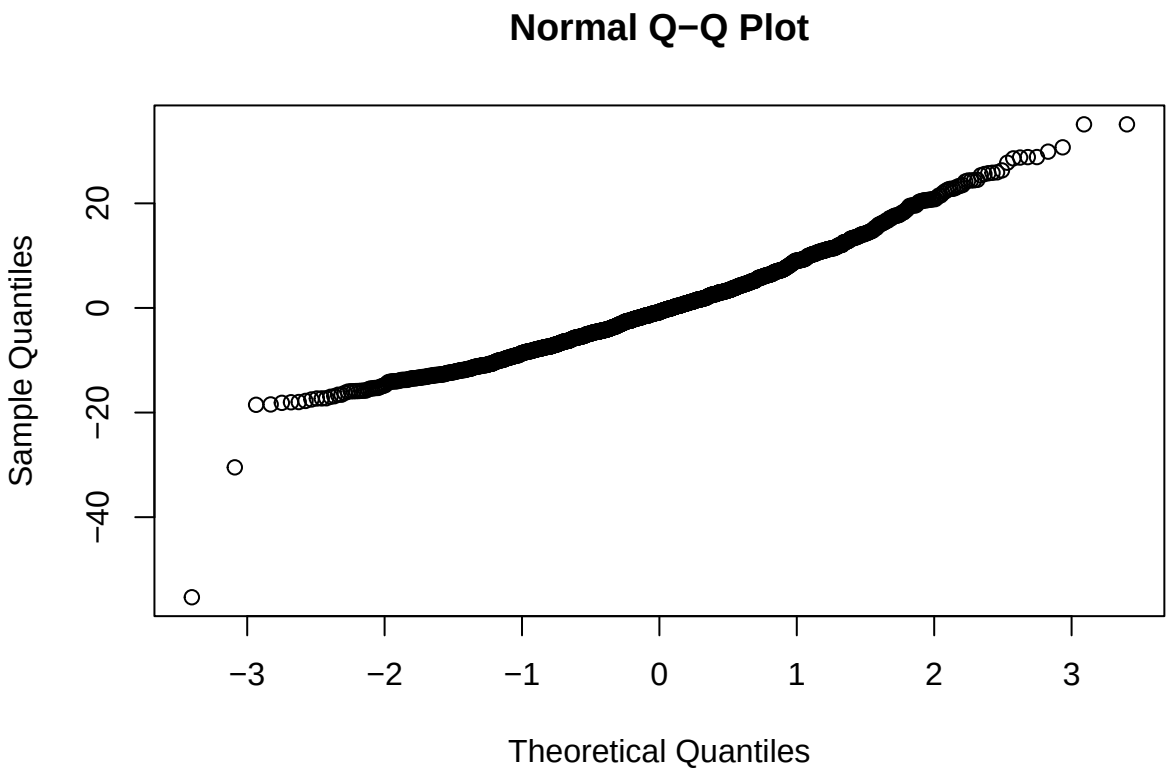

```
boxplot(Xtrain$co2)
```




```
plot(Xtrain.co2.lars.lasso.residuals4, main ="res4")
```




```
qqnorm(Xtrain.co2.lars.lasso.residuals4)
```




### 4.3 ANOVA

We don't like ANOVA's.

```
summary(Xtrain.co2.lars.lasso)
```

```
## LARS/LASSO
## Call: lars(x = X, y = y, type = "lasso", normalize = TRUE, intercept = TRUE)
##    Df     Rss         Cp
## 0   1 4500018 419868.577
## 1   2 1507425 139654.210
## 2   3  244659  21415.055
## 3   4  122109   9941.879
## 4   5   59997   4127.871
## 5   6   57121   3860.646
## 6   7   21031    483.311
## 7   8   17873    189.543
## 8   9   16952    105.287
## 9  10   16907    103.082
## 10 11   16871    101.734
## 11 12   16152     36.444
## 12 11   15931     13.730
## 13 12   15891     12.000
```

### 4.4 metrics

```
print(paste("R2:",Xtrain.co2.lars.lasso$R2[4]))
```

```
## [1] "R2: 0.972864769725914"
```

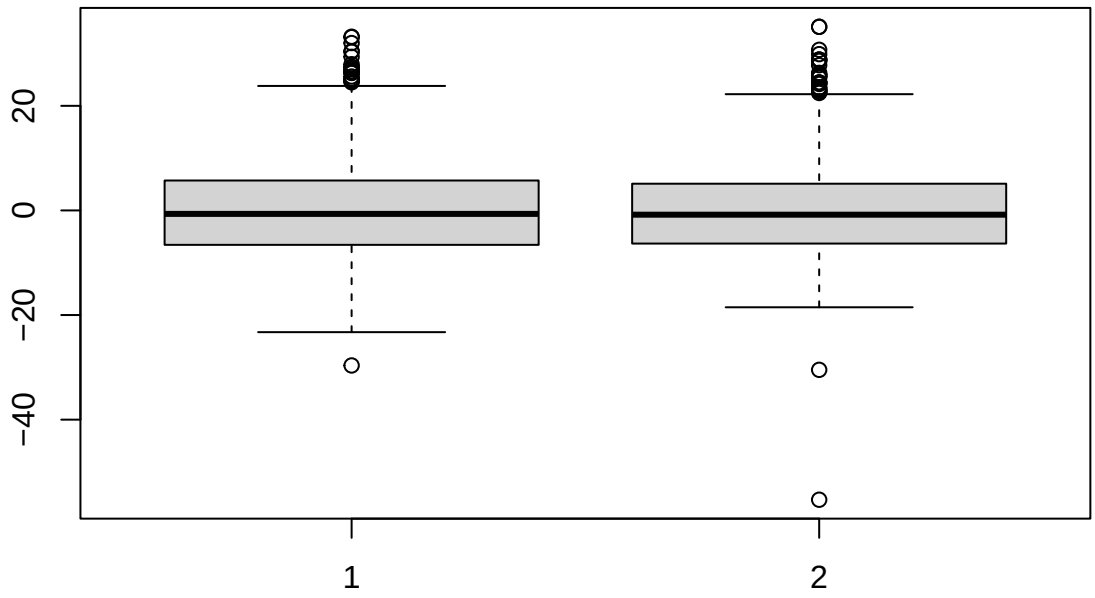
```
print(paste("RMSE:",sqrt(Xtrain.co2.lars.lasso$RSS[4]/1500)))
```

```
## [1] "RMSE: 9.02252820927992"
```

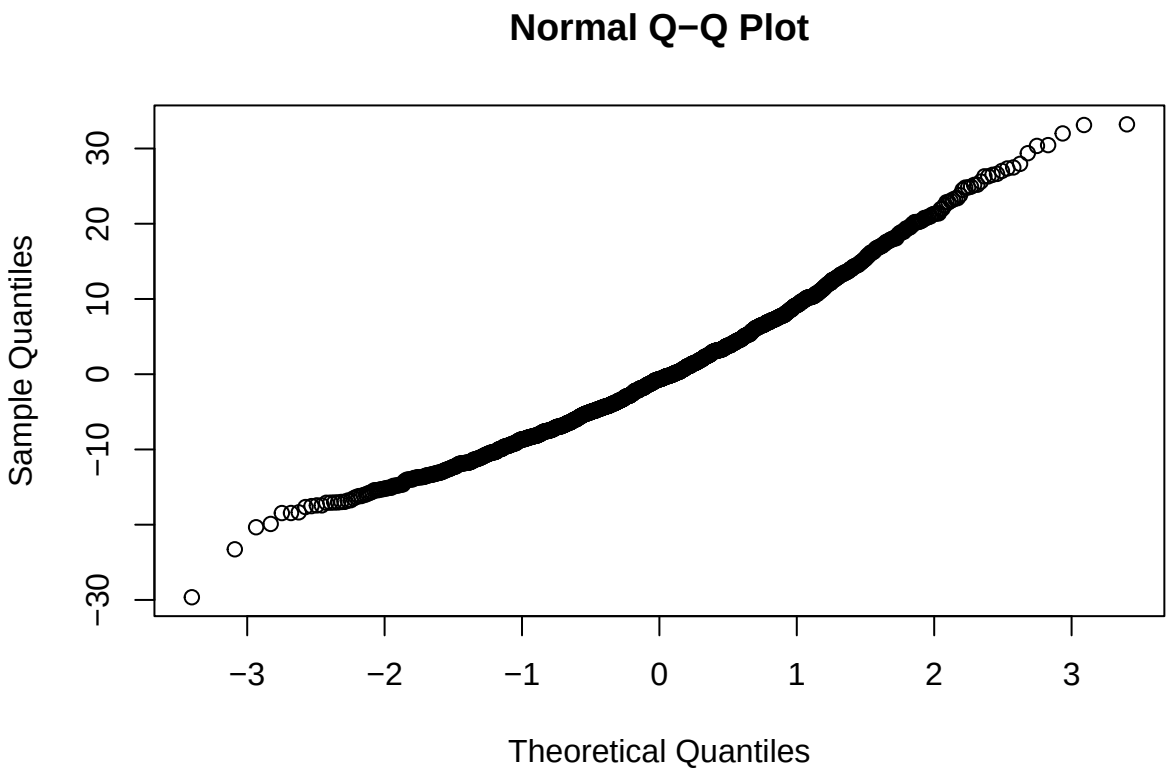
```
Xv <- model.matrix(co2 ~ ., data=subset(Xval, select = -inTrain))[,-1]
Xval.co2.lars.lasso.residuals4 <- Xval[,"co2"]  - predict(Xtrain.co2.lars.lasso,Xv,4)$fit
print(paste("RMSEP:",sqrt(sum(Xval.co2.lars.lasso.residuals4^2)/1500)))
```

```
## [1] "RMSEP: 9.20487312663879"
```

```
boxplot(Xval.co2.lars.lasso.residuals4,Xtrain.co2.lars.lasso.residuals4)
```




```
qqnorm(Xval.co2.lars.lasso.residuals4)
```




We found nothing wrong here.

### 4.5 confidence interval

```
#test_car <- c(euro_standard4 = 1,
#               euro_standard5 = 0,
#               transmission_typeManual =0,
#               engine_capacity =1924,
#               fuel_typePetrol =1,
#               urban_metric =9.899,
#               extra_urban_metric=6.898,
#               combined_metric =8.436,
#               noise_level = 71, #should be transformed
#               co_emissions=142.35,
#               nox_emissions= 429)
#test_car <- data.frame(1,0,0,1924,1,9.899,6.898,8.436, 71,142.35,429)
#
#
#predict(Xtrain.co2.lars.lasso,test_car,4)$fit
```

We can't get predict to work ... (it's a lot of typing) the way you could find a confidence interval is by using the fact that the conditional distribution of X is approximately normal (1 of the assumptions that we checked). The prediction gives the mean and the variance (which we assume to be constant) can be estimated using the residuals of the validation set.

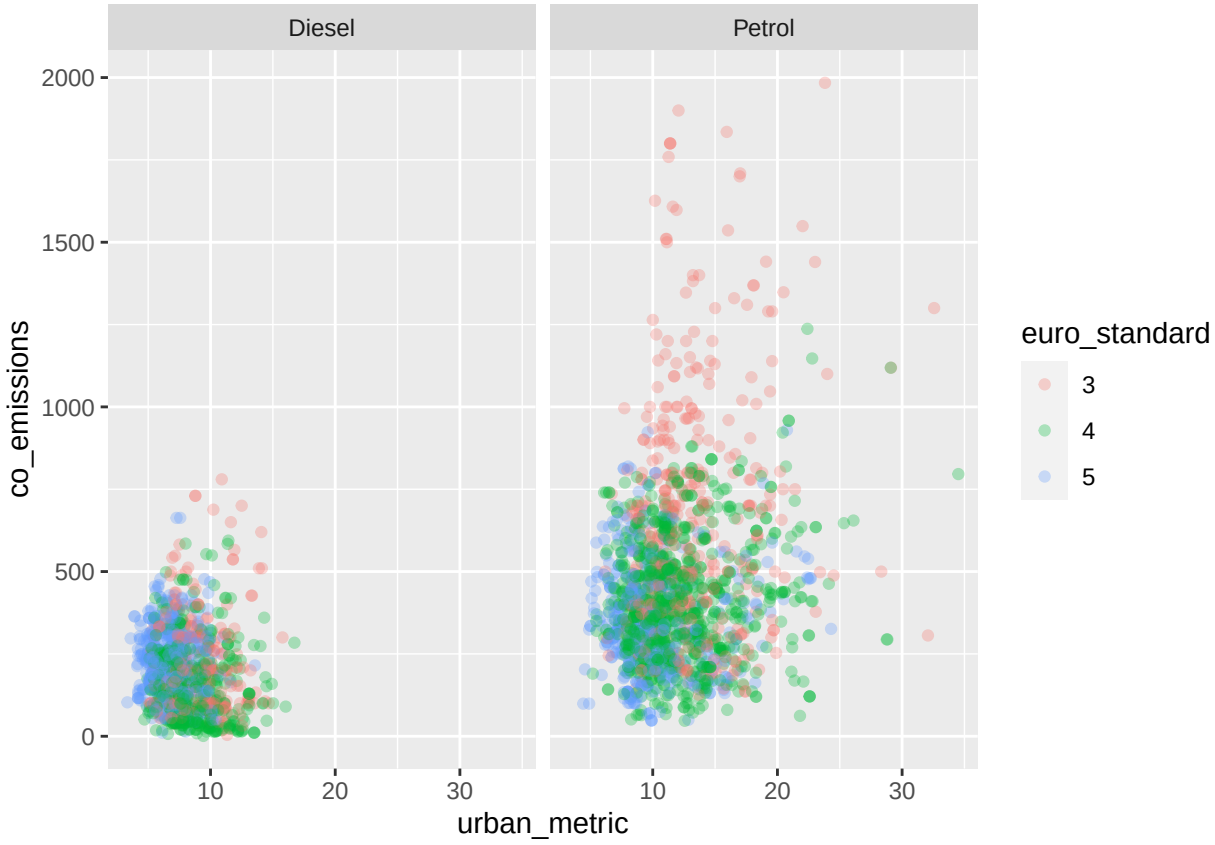
The model is chosen so that is performs well on the majority of the data (or on the underlying distribution you get the data from). The second car has an unusual high engine capacity making the prediction that this

model would give for it untrustworthy.

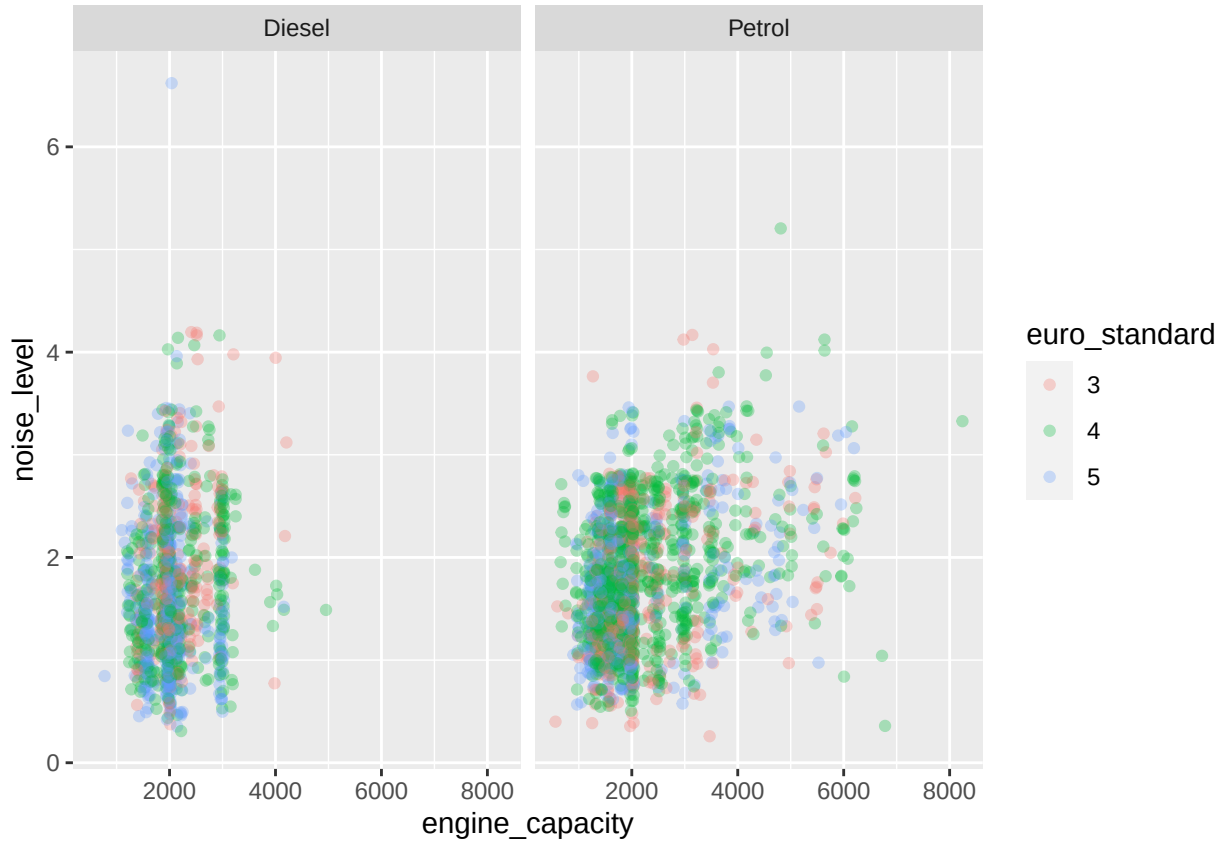
# 5 Classification

## 5.1 LDA and QDA

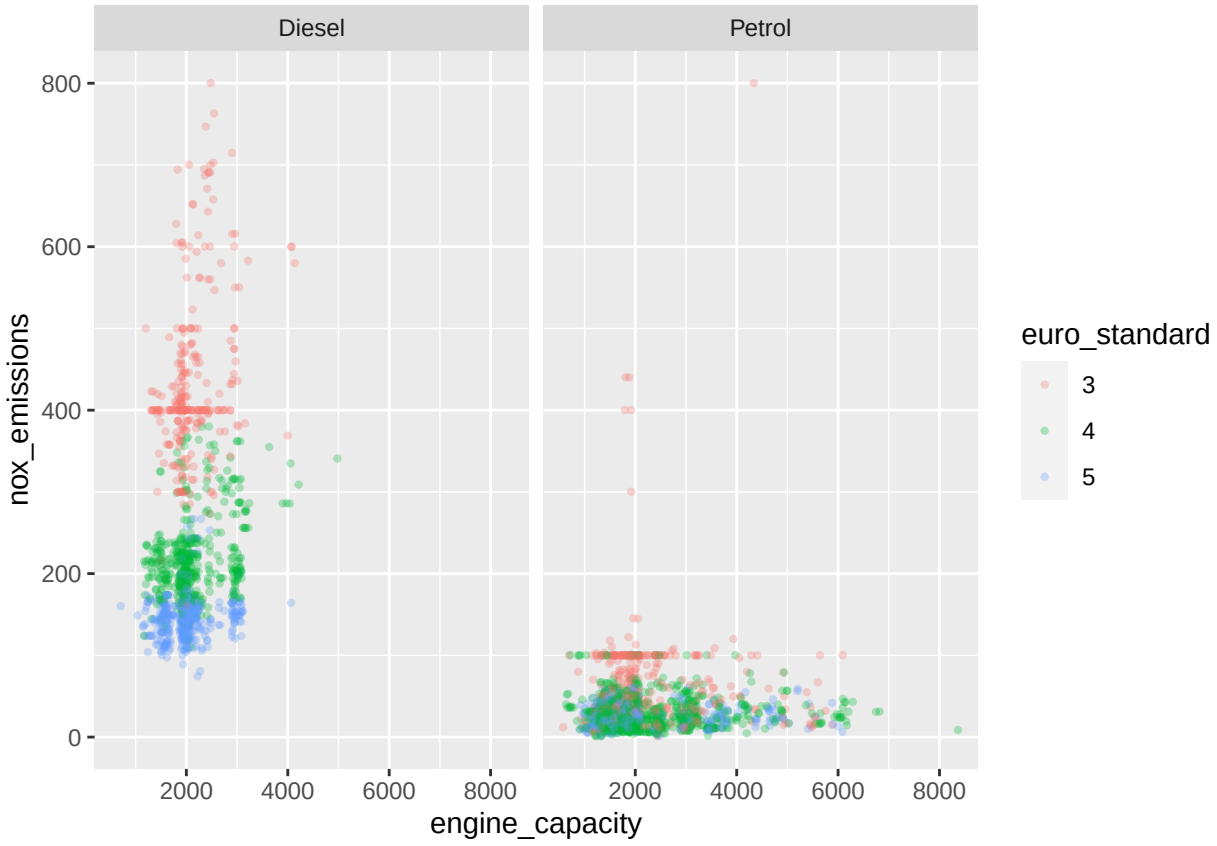
```
ggplot(subset.trans, aes(x = urban_metric, y = co_emissions, color = euro_standard)) +
  geom_jitter(alpha = 0.3 ) +
  facet_grid(~fuel_type)
```




```
ggplot(subset.trans, aes(x = engine_capacity, y = noise_level, color = euro_standard)) +
  geom_jitter(alpha = 0.3, width=50, height = 0.3 ) +
  facet_grid(~fuel_type)
```




```
ggplot(subset.trans, aes(x = engine_capacity, y = nox_emissions, color = euro_standard)) +
  geom_jitter(alpha = 0.3, width=100, height = 0.3, size = 0.8 ) +
  facet_grid(~fuel_type)
```




### 5.1.1 LDA

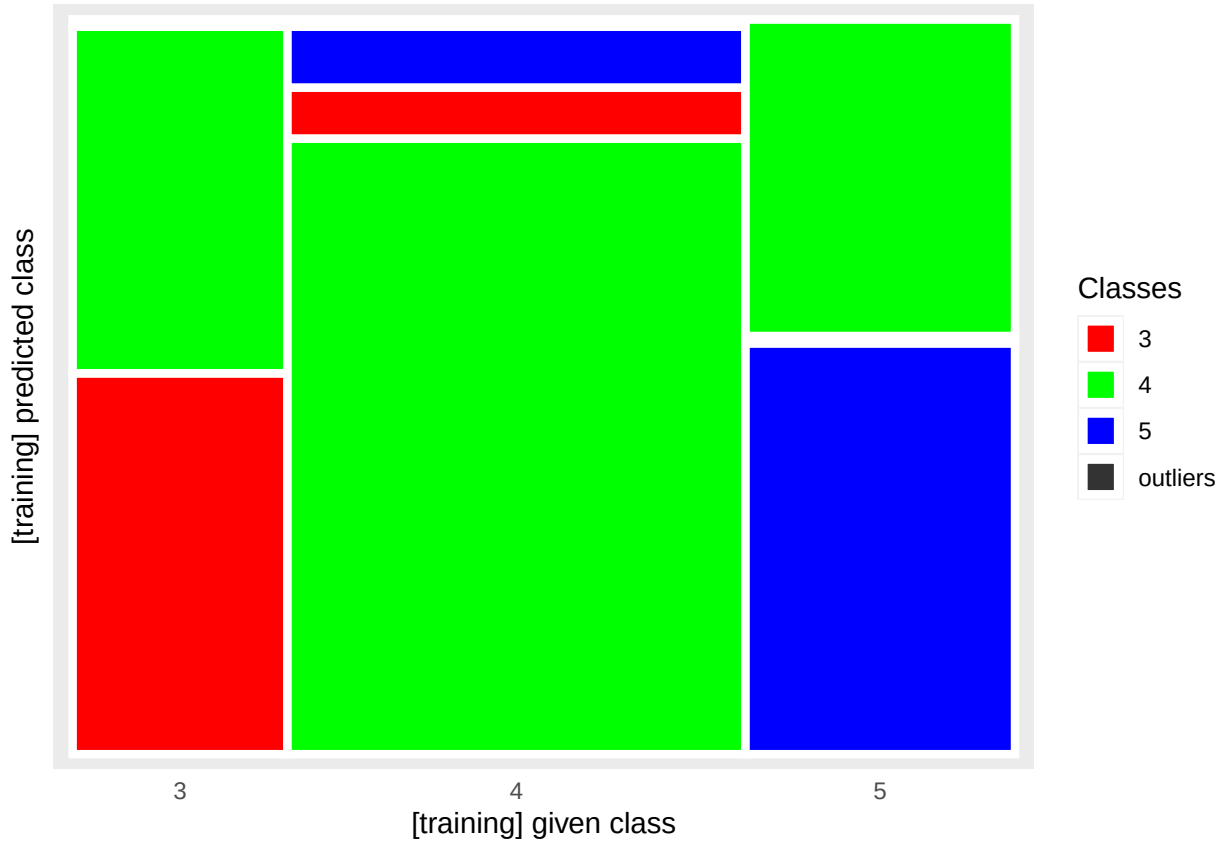
```
Xtrain.roblda<- vcr.da.train(Xtrain[,c("engine_capacity","urban_metric","nox_emissions")], Xtrain$euro_s
table(Xtrain$euro_standard)
```

```
##
##   3   4   5
## 342 729 429
```

```
confmat.vcr(Xtrain.roblda, cutoff = 0.99)
```

```
##
## Confusion matrix:
##       predicted
## given   3   4   5 outl
##     3 178 162   2    0
##     4  51 616  61    1
##     5   4 185 240    0
##
## The accuracy is 69%.
```

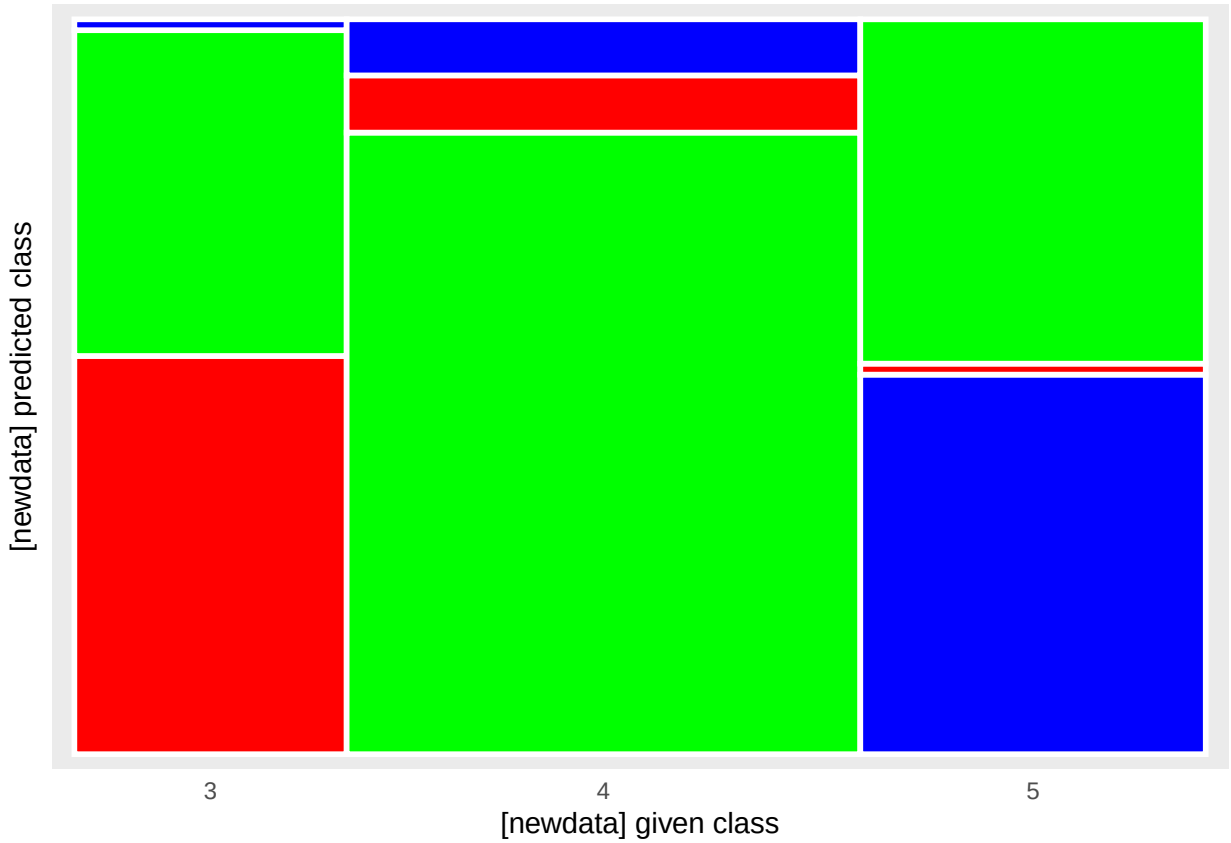
```
stackedplot(Xtrain.roblda, separSize = 1.5, minSize = 1, showLegend = TRUE)
```




```
Xval.roblda <- vcr.da.newdata(Xval[,c("engine_capacity","urban_metric","nox_emissions")], Xval$euro_star
confmat.vcr(Xval.roblda)
```

```
##
## Confusion matrix:
##       predicted
## given   3   4   5
##     3 198 162   1
##     4  53 576  52
##     5   6 215 237
##
## The accuracy is 67.4%.
```

```
stackedplot(Xval.roblda)
```




### 5.1.2 QDA

```
# ill conditioned
#Xtrain.robqda<- vcr.da.train(Xtrain[,c("engine_capacity","urban_metric","noise_level","nox_emissions")]
Xtrain.robqda<- vcr.da.train(Xtrain[,c("urban_metric","nox_emissions")], Xtrain$euro_standard, rule = "
```

```
## Warning in transfo(farness, type = "YJ", robust = TRUE, prestandardize = FALSE, : The data has 33.6%
## this function was designed for less than 25% of outliers.
```

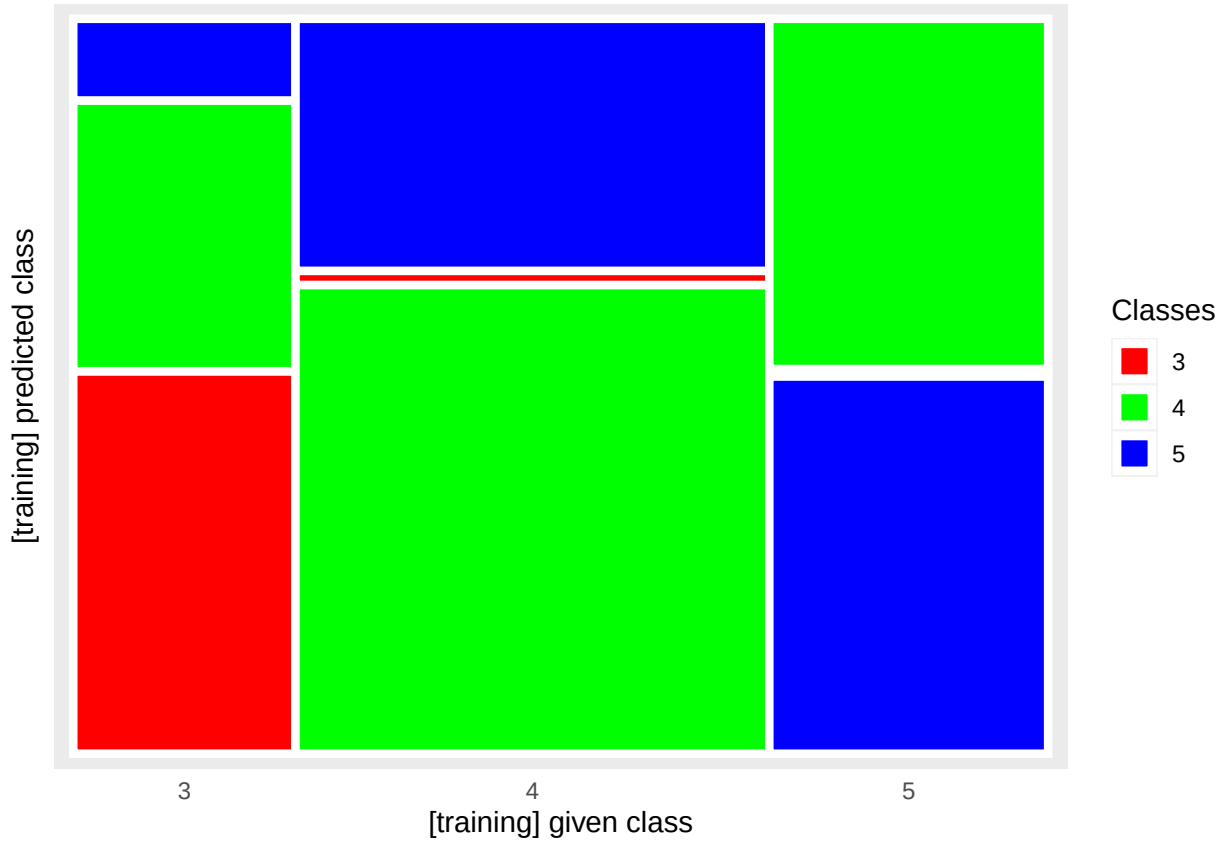
```
table(Xtrain$euro_standard)
```

```
##
##   3   4   5
## 342 729 429
```

```
confmat.vcr(Xtrain.robqda, cutoff = 0.99)
```

```
##
## Confusion matrix:
##       predicted
## given   3   4   5
##     3 178 126  38
##     4  14 465 250
##     5   1 206 222
##
## The accuracy is 57.67%.
```

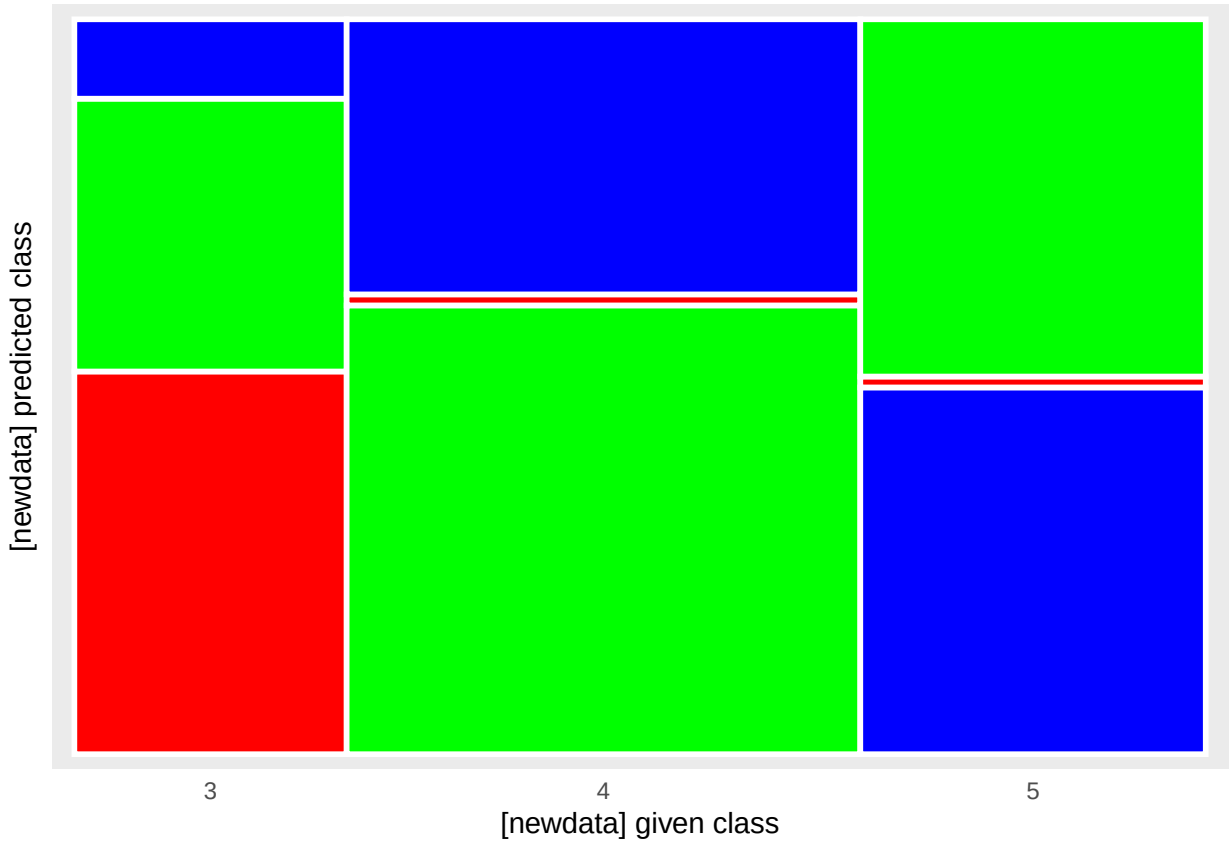
```
stackedplot(Xtrain.robqda, separSize = 1.5, minSize = 1, showLegend = TRUE)
```




```
Xval.robqda <- vcr.da.newdata(Xval[,c("urban_metric","nox_emissions")], Xval$euro_standard,Xtrain.robqda
confmat.vcr(Xval.robqda)
```

```
##
## Confusion matrix:
##       predicted
## given   3   4   5
##     3 188 134  39
##     4  10 416 255
##     5   2 225 231
##
## The accuracy is 55.67%.
```

```
stackedplot(Xval.robqda)
```




### 5.2 knn

```
Xtrain.knn <- knn(train = Xtrain[,c("engine_capacity","urban_metric","nox_emissions")], test=Xval[,c("e
control <- trainControl(method = "cv", number = 20)
grid <- expand.grid(k = 1:20)

results <- train(x = Xtrain[,c("engine_capacity","urban_metric","nox_emissions")], y = Xtrain$euro_stan
# Print the results
print(results)
```

```
## k-Nearest Neighbors
##
## 1500 samples
##    3 predictor
##    3 classes: '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (20 fold)
## Summary of sample sizes: 1425, 1426, 1426, 1425, 1426, 1425, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.7554229  0.6084378
##    2  0.7326735  0.5716795
##    3  0.7306643  0.5681785
##    4  0.7120237  0.5331082
```

```
##    5  0.7126826  0.5332559
##    6  0.6973656  0.5083775
##    7  0.7080680  0.5244721
##    8  0.6860318  0.4879758
##    9  0.6933917  0.5000878
##   10  0.6934718  0.4993921
##   11  0.6881470  0.4902156
##   12  0.6760932  0.4689374
##   13  0.6721107  0.4619904
##   14  0.6741555  0.4609677
##   15  0.6755339  0.4631147
##   16  0.6674967  0.4486401
##   17  0.6728658  0.4579142
##   18  0.6754794  0.4600345
##   19  0.6781283  0.4644695
##   20  0.6768042  0.4598230
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```
Xtrain.knn1 <- vcr.knn.train(Xtrain[,c("engine_capacity","urban_metric","nox_emissions")], Xtrain$euro_s
```

```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 29.81% o
## this function was designed for less than 25% of outliers.
```

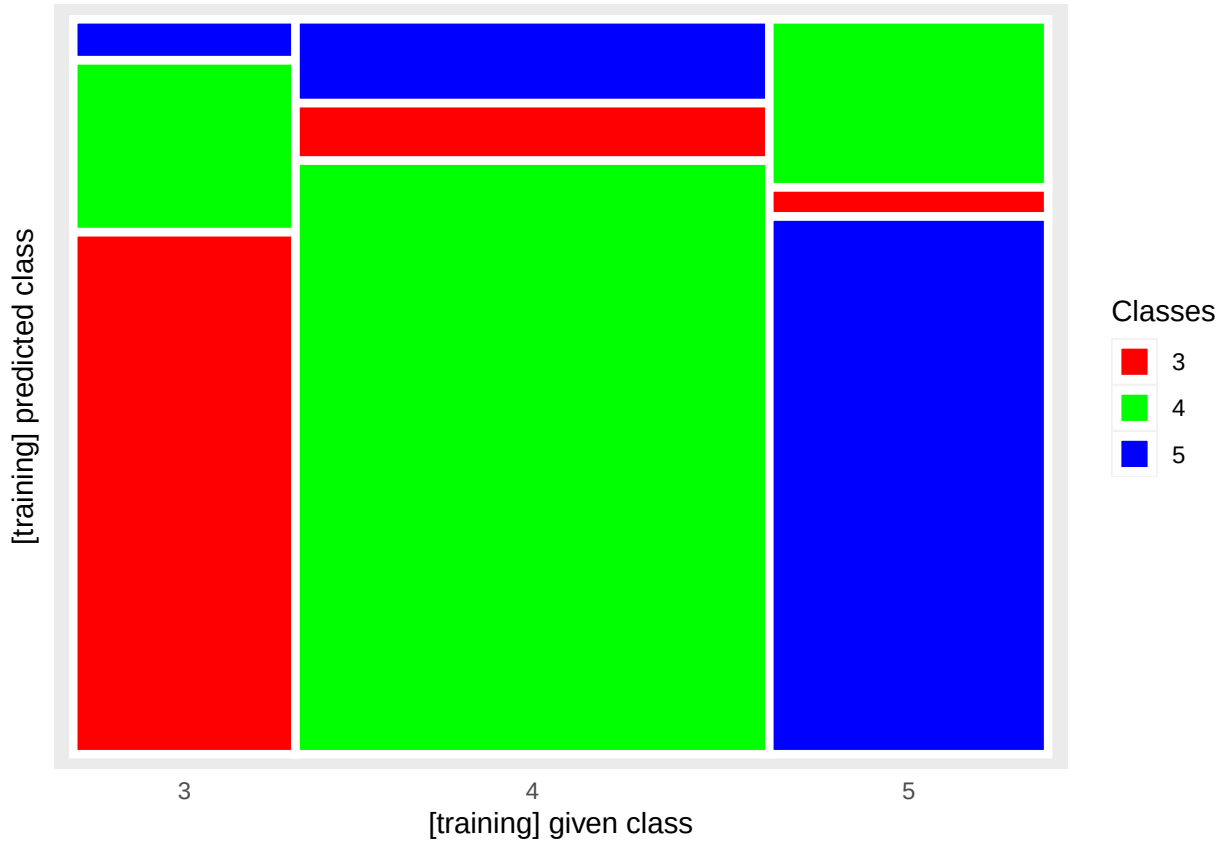
```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 29.81% o
## this function was designed for less than 25% of outliers.
```

```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 30.5% of
## this function was designed for less than 25% of outliers.
```

```
confmat.vcr(Xtrain.knn1, cutoff = 0.99)
```

```
##
## Confusion matrix:
##       predicted
## given   3   4   5
##     3 243  80  19
##     4  57 589  83
##     5  17  98 314
##
## The accuracy is 76.4%.
```

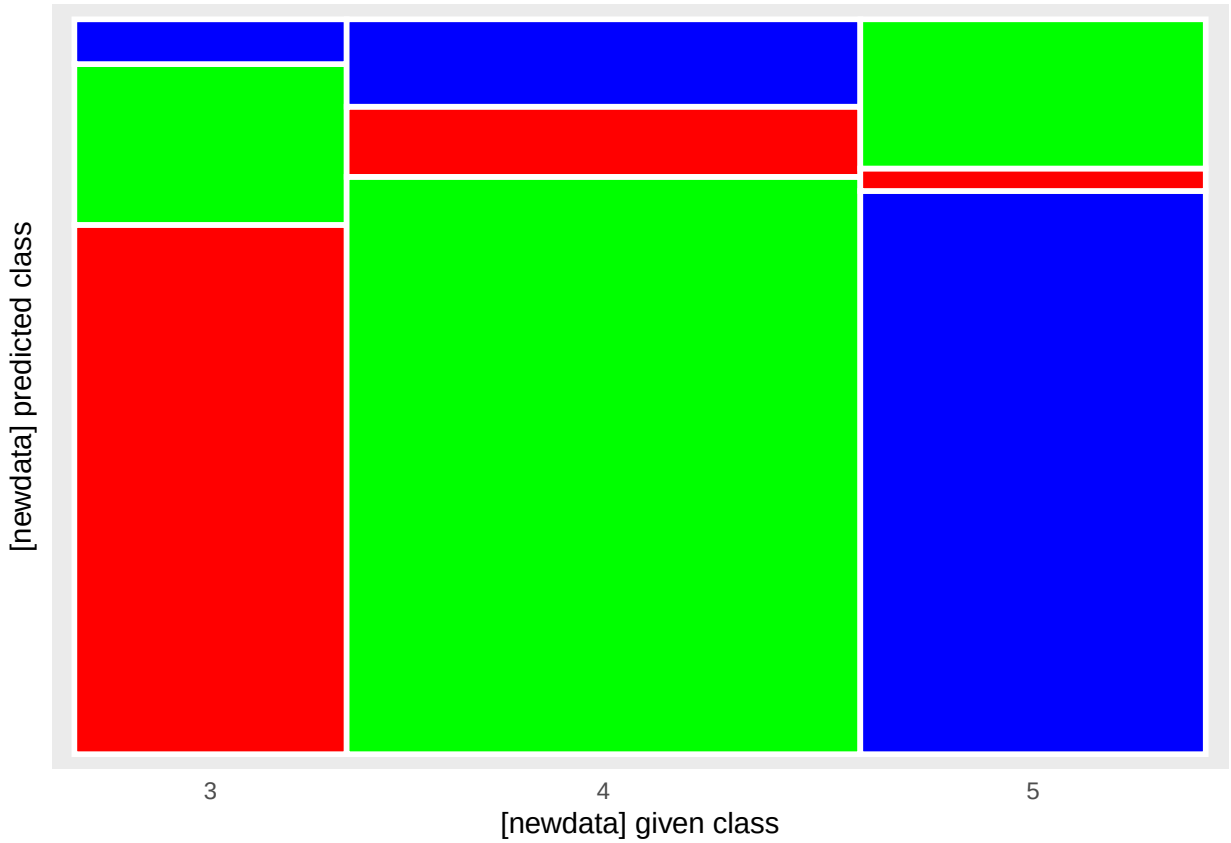
```
stackedplot(Xtrain.knn1, separSize = 1.5, minSize = 1, showLegend = TRUE)
```




```
Xval.knn1 <- vcr.knn.newdata(Xval[,c("engine_capacity","urban_metric","nox_emissions")], Xval$euro_stand
confmat.vcr(Xval.knn1)
```

```
##
## Confusion matrix:
##       predicted
## given   3   4   5
##     3 260  79  22
##     4  65 535  81
##     5  14  93 351
##
## The accuracy is 76.4%.
```

```
stackedplot(Xval.knn1)
```




### 5.3 prefered model

We prefer knn because of its simplicity and higher accuracy.

```
Xtrain.diesel <- subset(Xtrain, fuel_type == "Diesel")
Xtrain.petrol <- subset(Xtrain, fuel_type == "Petrol")

Xtrain.knn1.diesel <- vcr.knn.train(Xtrain.diesel[,c("engine_capacity","urban_metric","nox_emissions")]
```

```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 27.69% o
## this function was designed for less than 25% of outliers.
```

```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 25% of o
## this function was designed for less than 25% of outliers.
```

```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 32.81% o
## this function was designed for less than 25% of outliers.
```

```
Xtrain.knn1.petrol <- vcr.knn.train(Xtrain.petrol[,c("engine_capacity","urban_metric","nox_emissions")]
```

```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 31.32% o
## this function was designed for less than 25% of outliers.
```

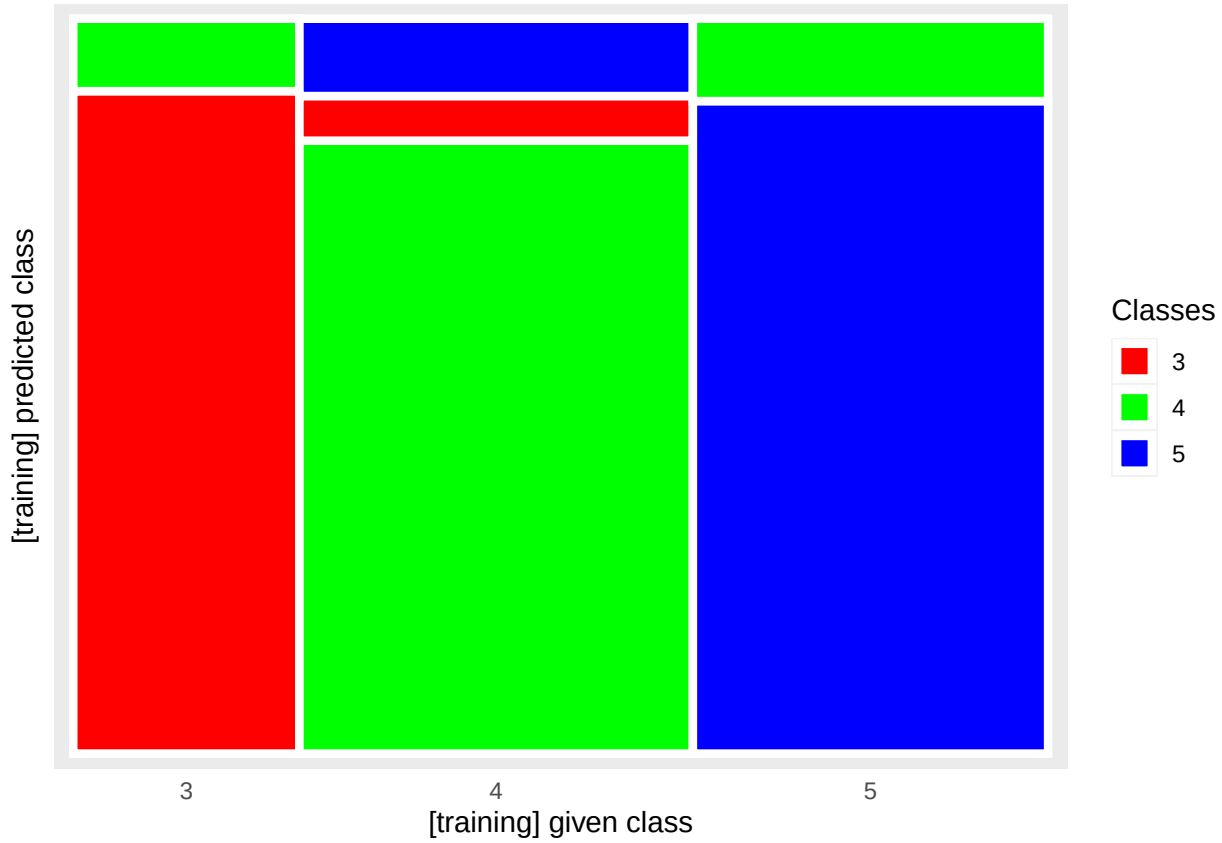
```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 26.93% o
## this function was designed for less than 25% of outliers.
```

```
## Warning in cellWise::transfo(X = sfar, robust = TRUE, prestandardize = FALSE, : The data has 27.03% o
## this function was designed for less than 25% of outliers.
```

```
confmat.vcr(Xtrain.knn1.diesel, cutoff = 0.99)
```

```
##
## Confusion matrix:
##       predicted
## given   3   4   5
##     3 128  14   0
##     4  15 206  26
##     5   0  25 198
##
## The accuracy is 86.93%.
```

```
stackedplot(Xtrain.knn1.diesel, separSize = 1.5, minSize = 1, showLegend = TRUE)
```




```
confmat.vcr(Xtrain.knn1.petrol, cutoff = 0.99)
```

```
##
## Confusion matrix:
##       predicted
## given   3   4   5
##     3 118  66  16
##     4  44 383  55
##     5  13  72 121
##
## The accuracy is 70.05%.
```

```
stackedplot(Xtrain.knn1.petrol, separSize = 1.5, minSize = 1, showLegend = TRUE)
```