Exercise 7

Tobias Raidl, 11717659

2023-12-12

Contents


```
cars = read.csv("cardata.csv")
cars = na.omit(cars)
df = cars[, -c(1:9,15,16,18)]
```

1

Use the function pfa() from the package StatDA for principal factor analysis, with the argument scores="regression" to also obtain scores. Inspect the biplot and try to interpret the factors.

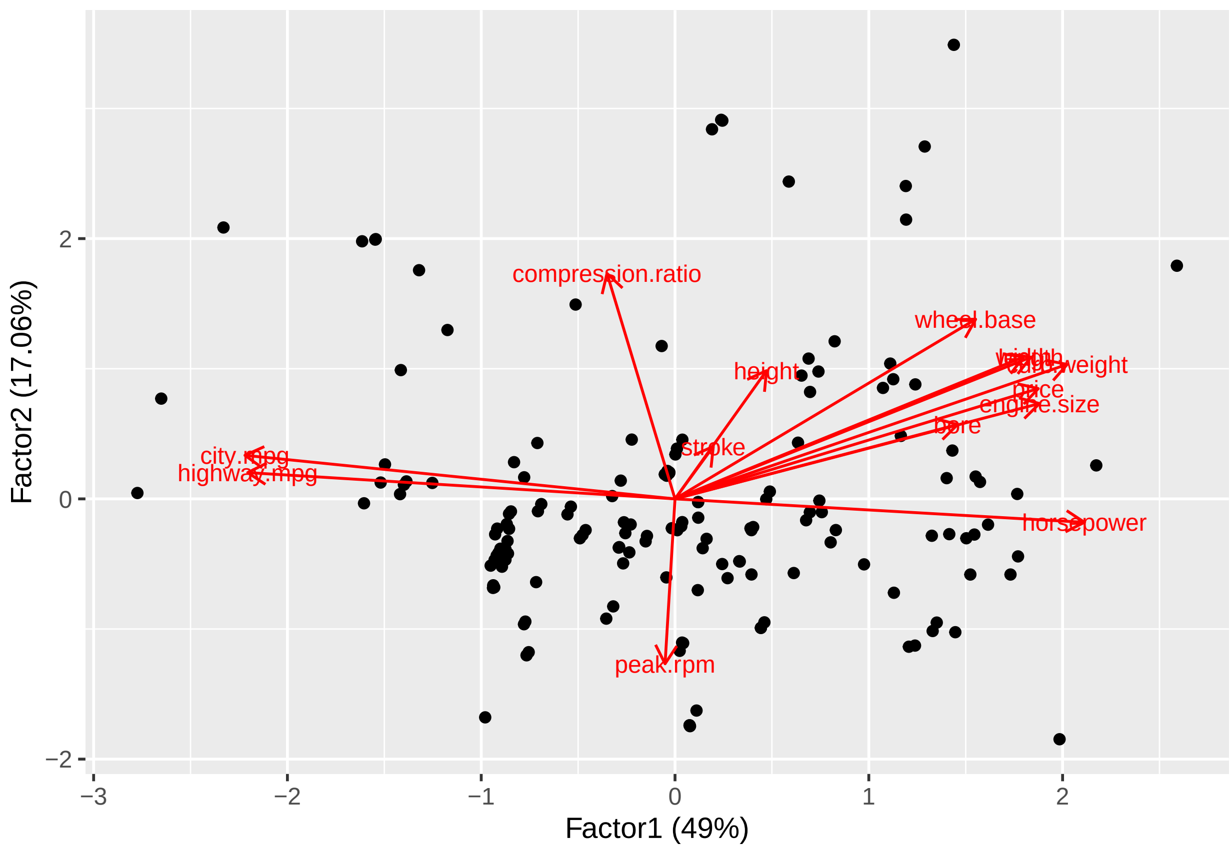
```
library(StatDA)
```

```
## Warning: package 'StatDA' was built under R version 4.3.2

## Loading required package: sgeostat

## Registered S3 method overwritten by 'geoR':
##   method         from
##   plot.variogram sgeostat
```

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.3.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
df_scaled = scale(df)
pfa = pfa(df_scaled, factors=2, scores="regression")
autoplot(pfa, data=df_scaled, loadings = TRUE, loadings.colour = 'red',
         loadings.label = TRUE, loadings.label.size = 3)
```




```
pfa
```

```
##
## Call:
## pfa(x = df_scaled, factors = 2, scores = "regression")
##
## Uniquenesses:
##        wheel.base            length             width            height
##             0.199             0.153             0.177             0.778
##       curb.weight       engine.size              bore            stroke
##             0.041             0.243             0.544             0.963
## compression.ratio        horsepower          peak.rpm          city.mpg
##             0.420             0.165             0.701             0.064
##       highway.mpg             price
##             0.089             0.214
##
## Loadings:
##                   Factor1 Factor2
## wheel.base         0.669   0.595
## length             0.792   0.469
## width              0.776   0.470
## height             0.203   0.425
## curb.weight        0.872   0.445
## engine.size        0.811   0.316
## bore               0.628   0.246
## stroke                     0.174
```

```
## compression.ratio -0.151   0.747
## horsepower         0.911
## peak.rpm                  -0.546
## city.mpg          -0.956   0.144
## highway.mpg       -0.950
## price              0.808   0.365
##
##                Factor1 Factor2
## SS loadings      6.859   2.388
## Proportion Var   0.490   0.171
## Cumulative Var   0.490   0.661
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is NA on 64 degrees of freedom.
## The p-value is NA
```

The first factor covers variables like horsepower, size, weight, highway.mpg and city.mpg (horizontal), and the second factor covers variables such as peak.rpm, height and compression rate. Wheel base for example is similarly covered by both factors.

2

Estimate robustly mean and covariance using the MCD estimator. Do the same as above, but provide the results from the MCD to the covmat argument to obtain a robust factor analysis solution. Note that scaling should then also be done robustly. Does the interpretation of the factors change?

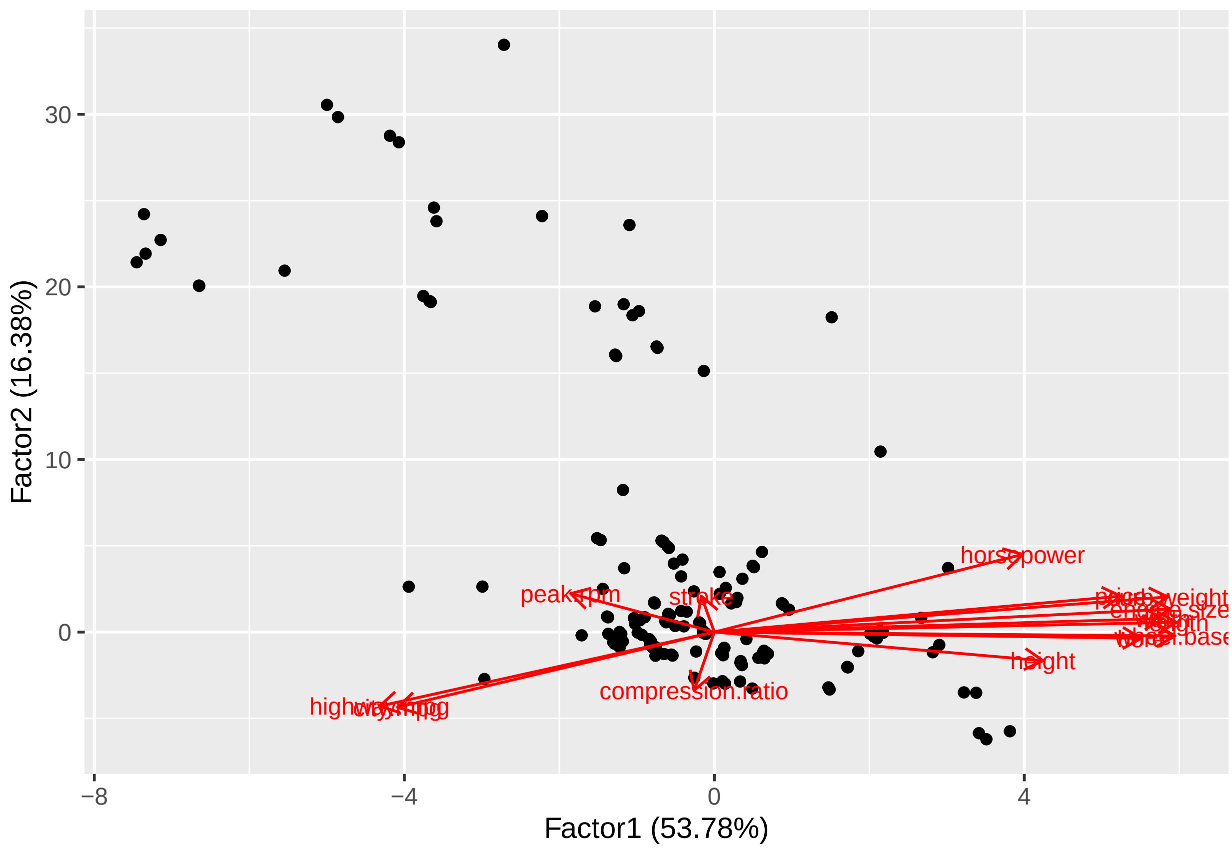
```r
library(robustbase)
```

```
## Warning: package 'robustbase' was built under R version 4.3.2
```

```r
library(DescTools)
```

```
## Warning: package 'DescTools' was built under R version 4.3.2
```

```r
covmat = covMcd(df)$cov
df_robscaled = RobScale(df)
set.seed(69)
robust_pfa = pfa(df_robscaled, factors=2, covmat=covmat, scores="regression")
autoplot(robust_pfa, data=df_robscaled, loadings = TRUE, loadings.colour = 'red',
         loadings.label = TRUE, loadings.label.size = 3)
```




```r
robust_pfa
```

```
##
## Call:
## pfa(x = df_robscaled, factors = 2, covmat = covmat, scores = "regression")
##
## Uniquenesses:
##        wheel.base            length             width            height
##             0.123             0.111             0.154             0.486
##       curb.weight       engine.size              bore            stroke
##             0.049             0.098             0.245             0.892
## compression.ratio        horsepower          peak.rpm          city.mpg
##             0.714             0.106             0.792             0.115
##       highway.mpg             price
##             0.081             0.211
##
## Loadings:
##                   Factor1 Factor2
## wheel.base         0.936
## length             0.939
## width              0.911   0.124
## height             0.668  -0.261
## curb.weight        0.921   0.320
## engine.size        0.926   0.211
## bore               0.867
## stroke                     0.327
```

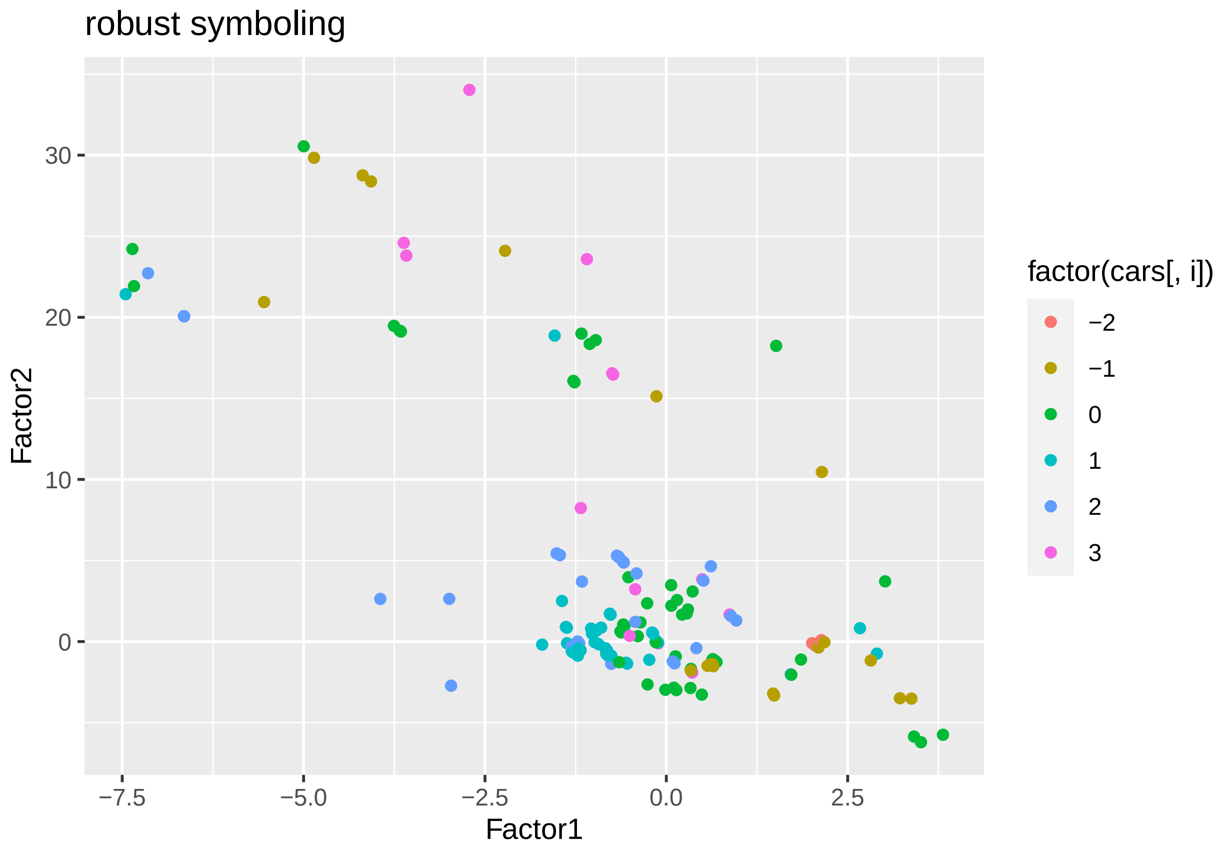
```
## compression.ratio         -0.533
## horsepower         0.626   0.708
## peak.rpm          -0.292   0.351
## city.mpg          -0.644  -0.686
## highway.mpg       -0.680  -0.676
## price              0.826   0.327
##
##                Factor1 Factor2
## SS loadings      7.530   2.293
## Proportion Var   0.538   0.164
## Cumulative Var   0.538   0.702
##
## The degrees of freedom for the model is 64 and the fit was NA
```

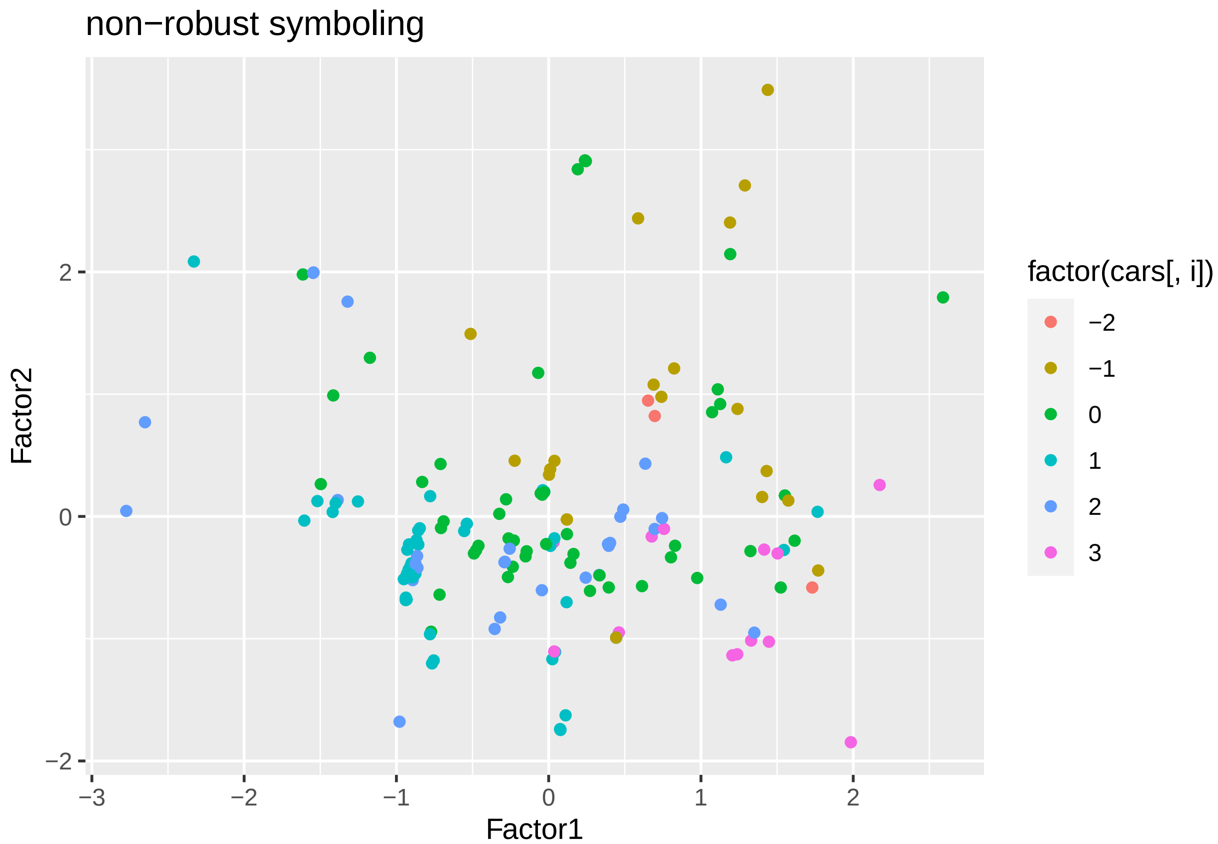
The explained variance by the first 2 factors is now ~5% higher than in the non-robust approach.

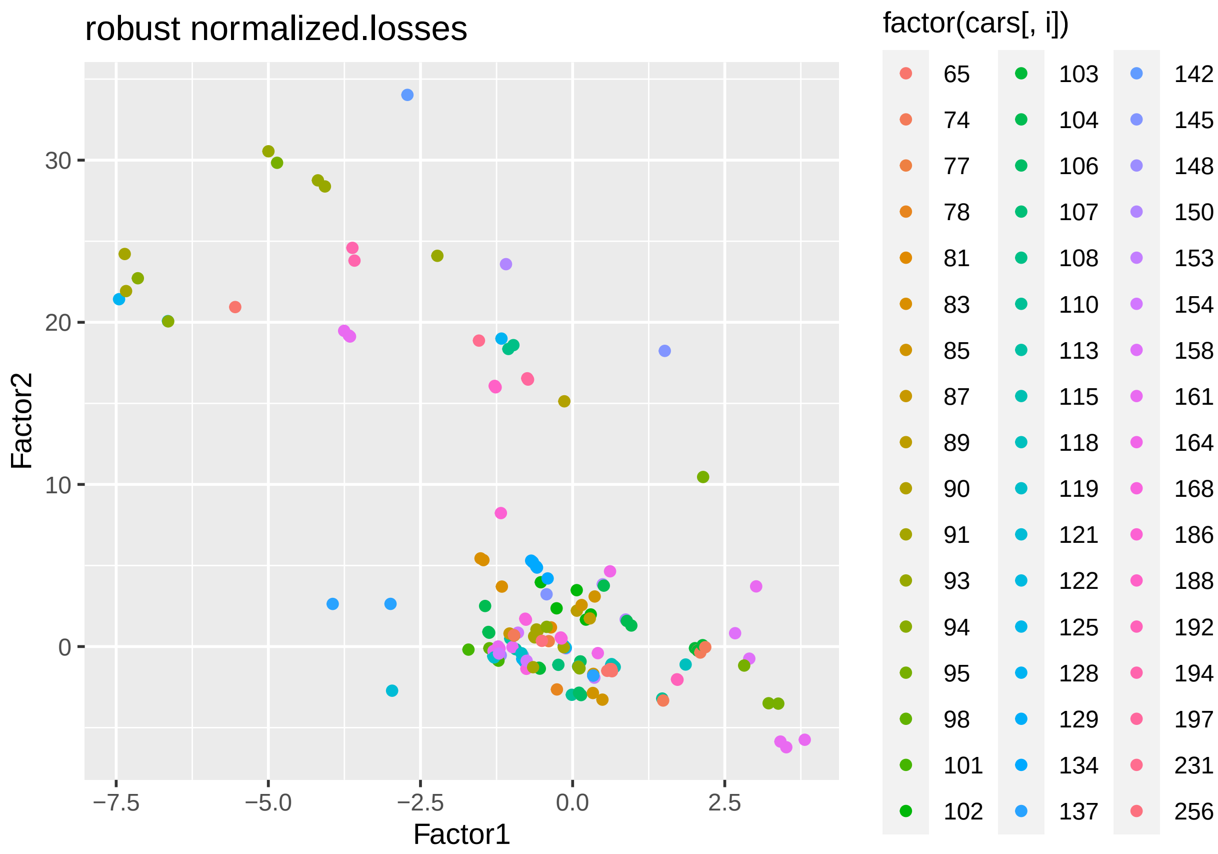
3

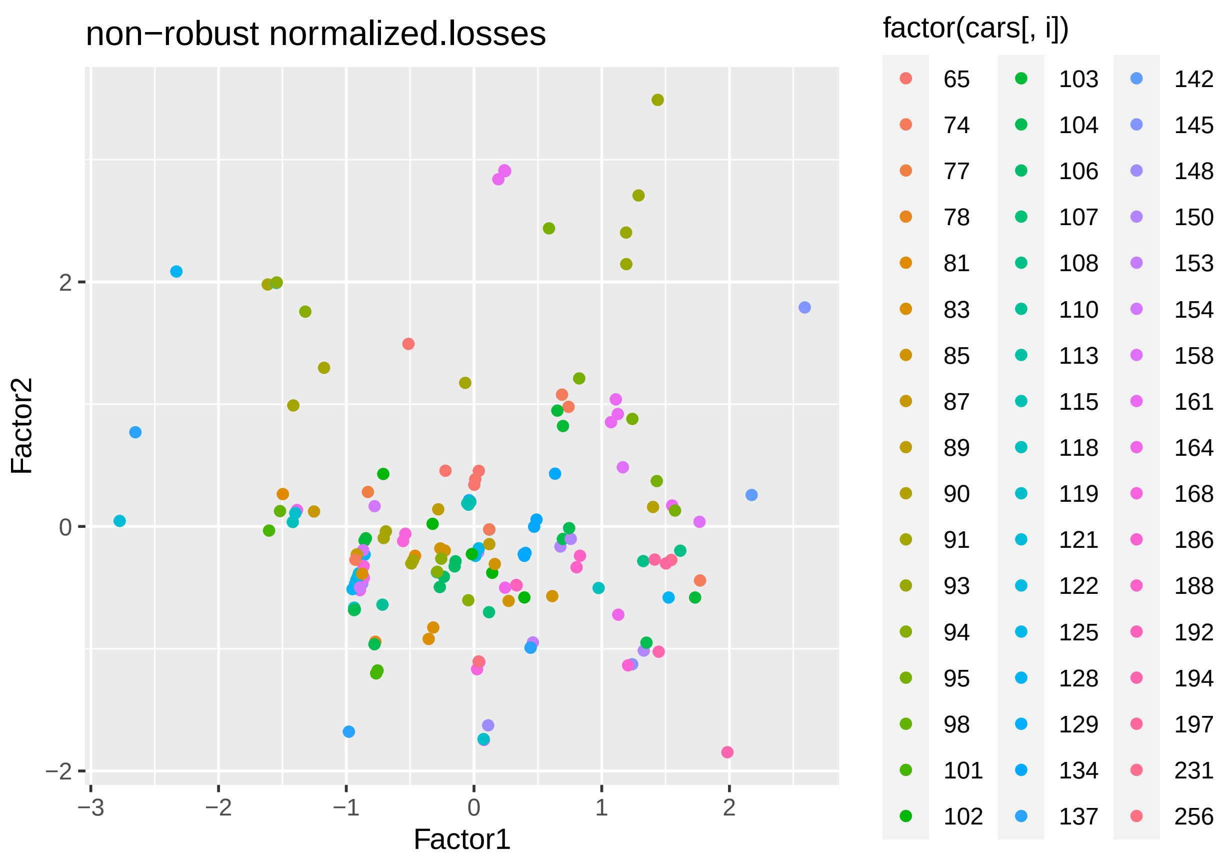
Look at the scores from robust factor analysis and try to identify a variable from the data set which is explaining outliers in the scores. So, essentially, plot the scores, with color according to other (categorical) variables). Do the same for the non-robust scores. Does the same variable also lead to an explanation of the outlyingness?

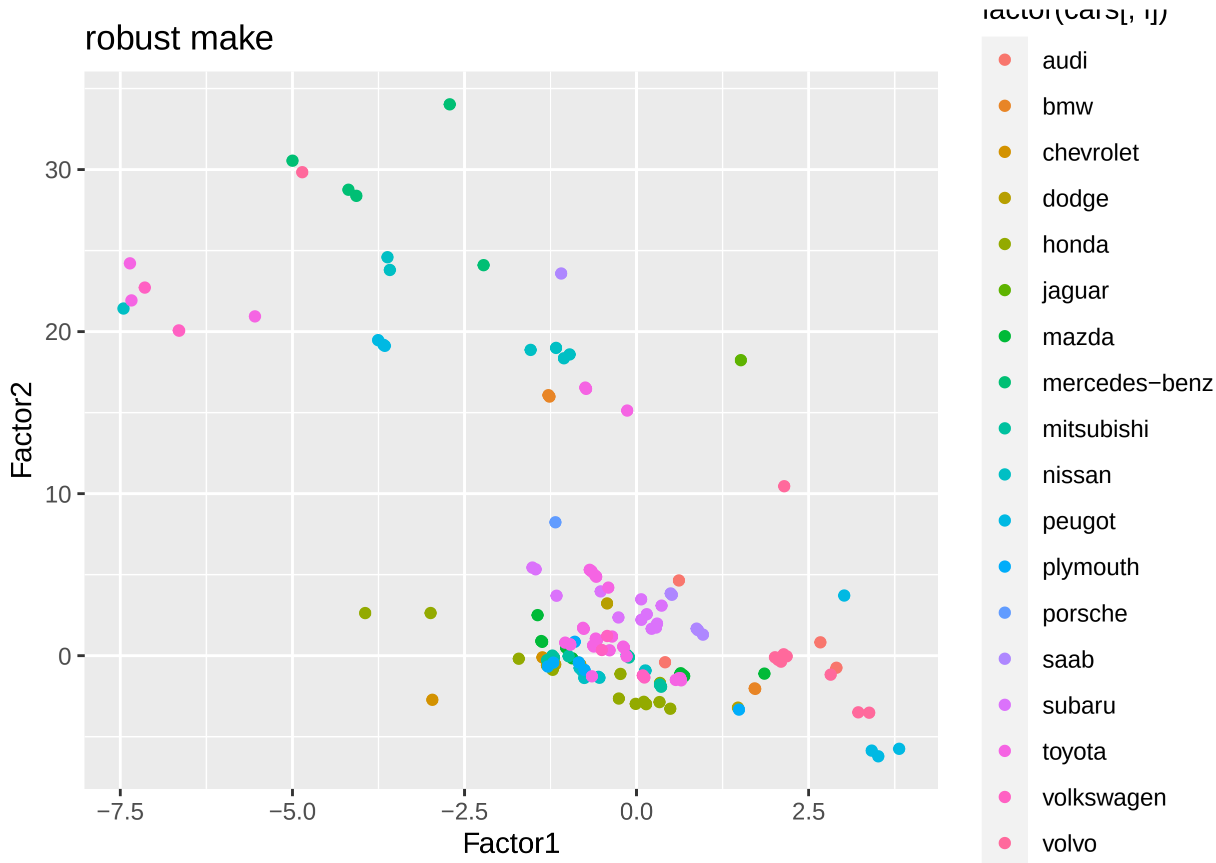
```r
library(ggplot2)

facvars = c(1:9,15,16,18)
for(i in facvars) {
  print(
    ggplot(robust_pfa$scores, aes(x=Factor1, y=Factor2, col=factor(cars[,i]))) +
      geom_point() +
      ggtitle(paste("robust", names(cars)[i]))
  )
    print(
    ggplot(pfa$scores, aes(x=Factor1, y=Factor2, col=factor(cars[,i]))) +
      geom_point() +
      ggtitle(paste("non-robust", names(cars)[i]))
  )
}
```

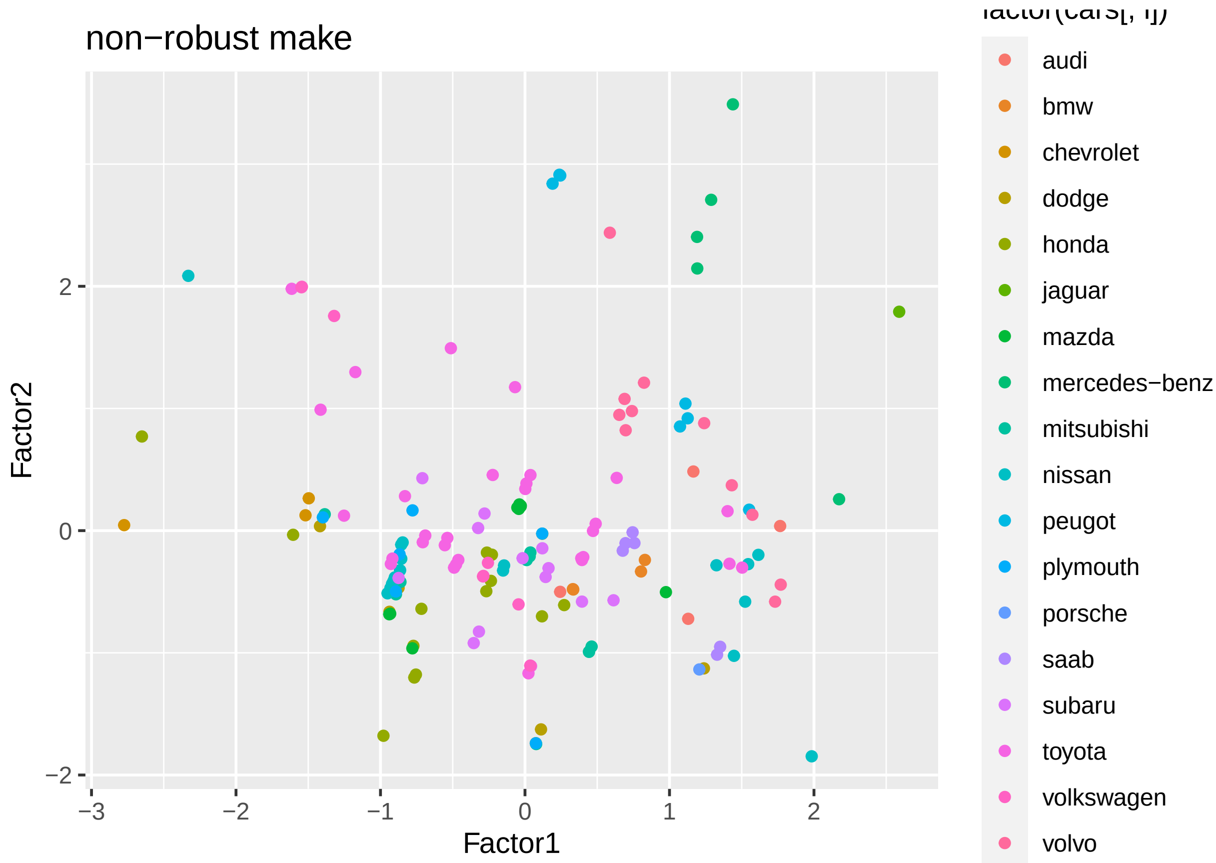

robust symboling
Factor2
Factor1
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
factor(cars[, i])
−2
−1
0
1
2
3

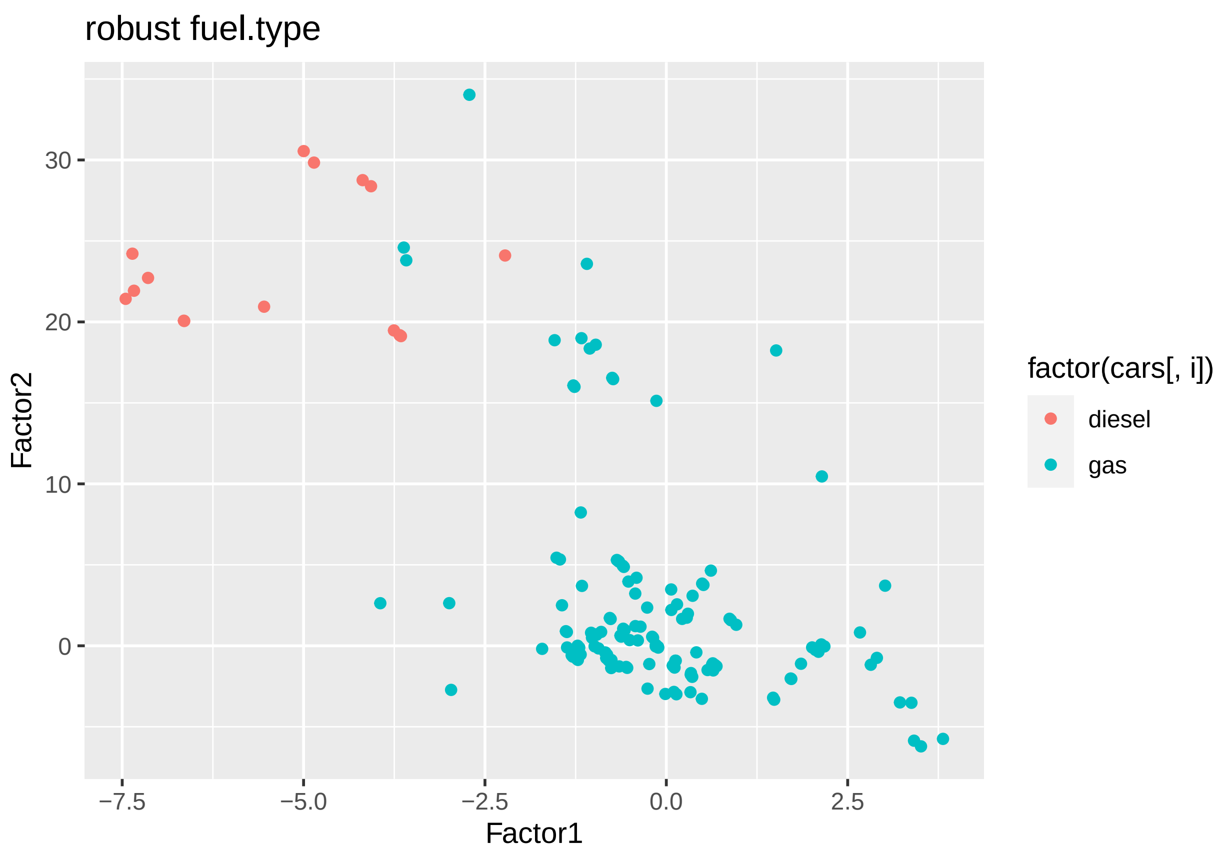

non−robust symboling
Factor2
Factor1
2
0
−2
−3
−2
−1
0
1
2
factor(cars[, i])
−2
−1
0
1
2
3

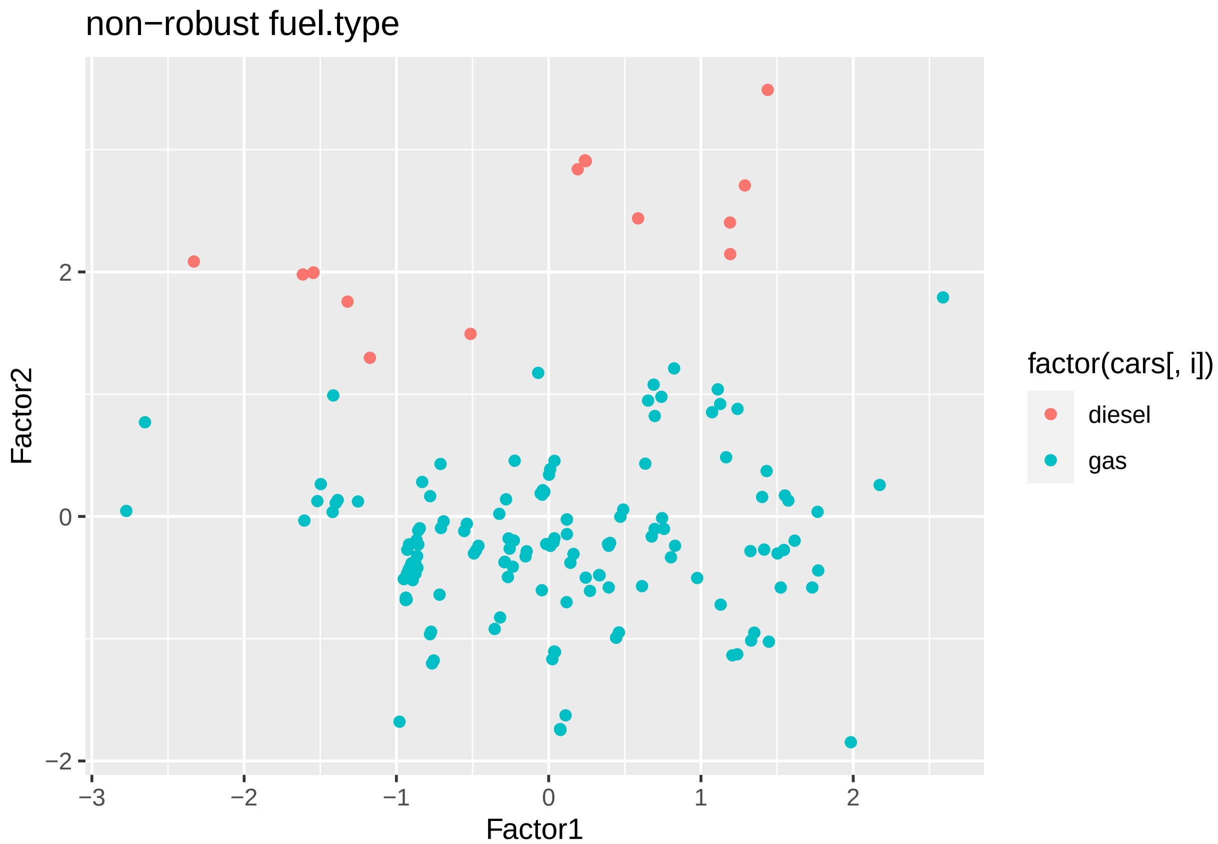

robust normalized.losses
Factor2
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
Factor1
factor(cars[, i])
65
74
77
78
81
83
85
87
89
90
91
93
94
95
98
101
102
103
104
106
107
108
110
113
115
118
119
121
122
125
128
129
134
137
142
145
148
150
153
154
158
161
164
168
186
188
192
194
197
231
256

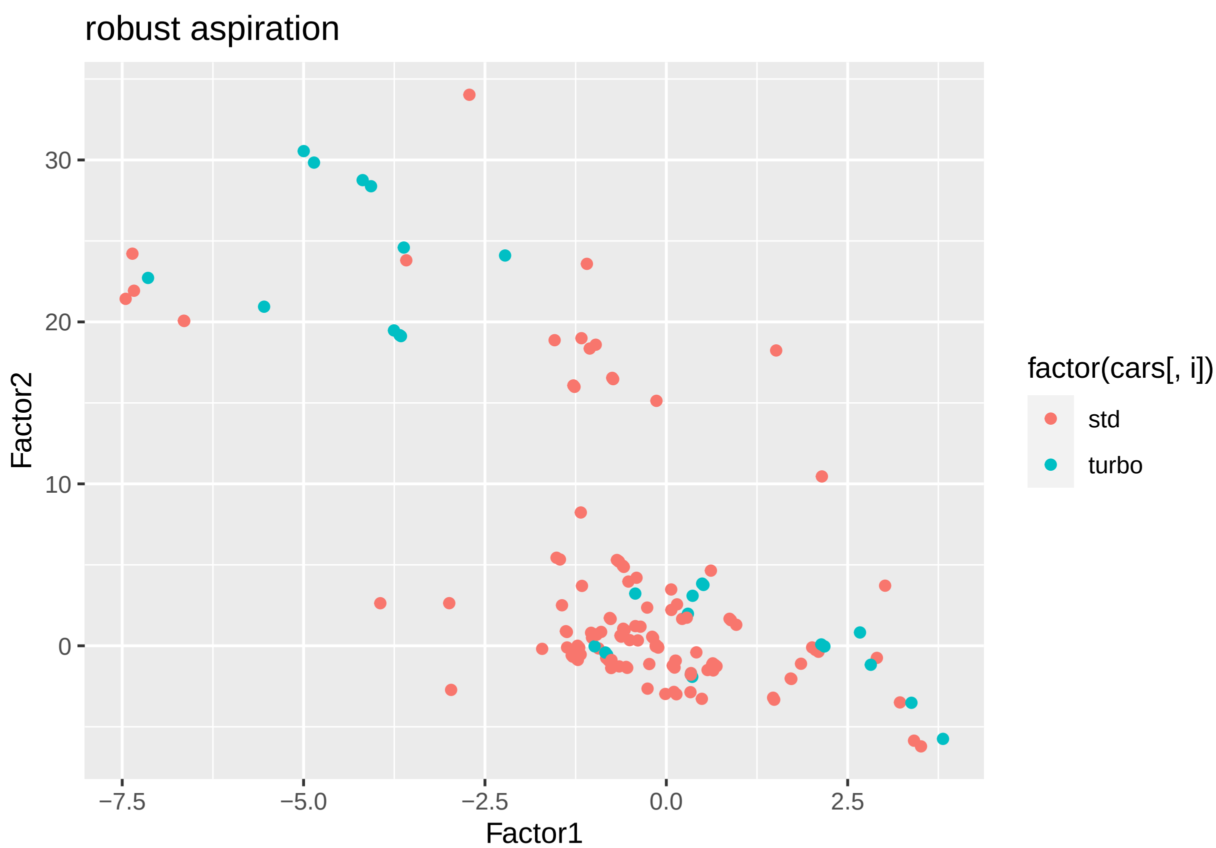

non−robust normalized.losses
Factor2
Factor1
2
0
−2
−3
−2
−1
0
1
2
factor(cars[, i])
65
74
77
78
81
83
85
87
89
90
91
93
94
95
98
101
102
103
104
106
107
108
110
113
115
118
119
121
122
125
128
129
134
137
142
145
148
150
153
154
158
161
164
168
186
188
192
194
197
231
256

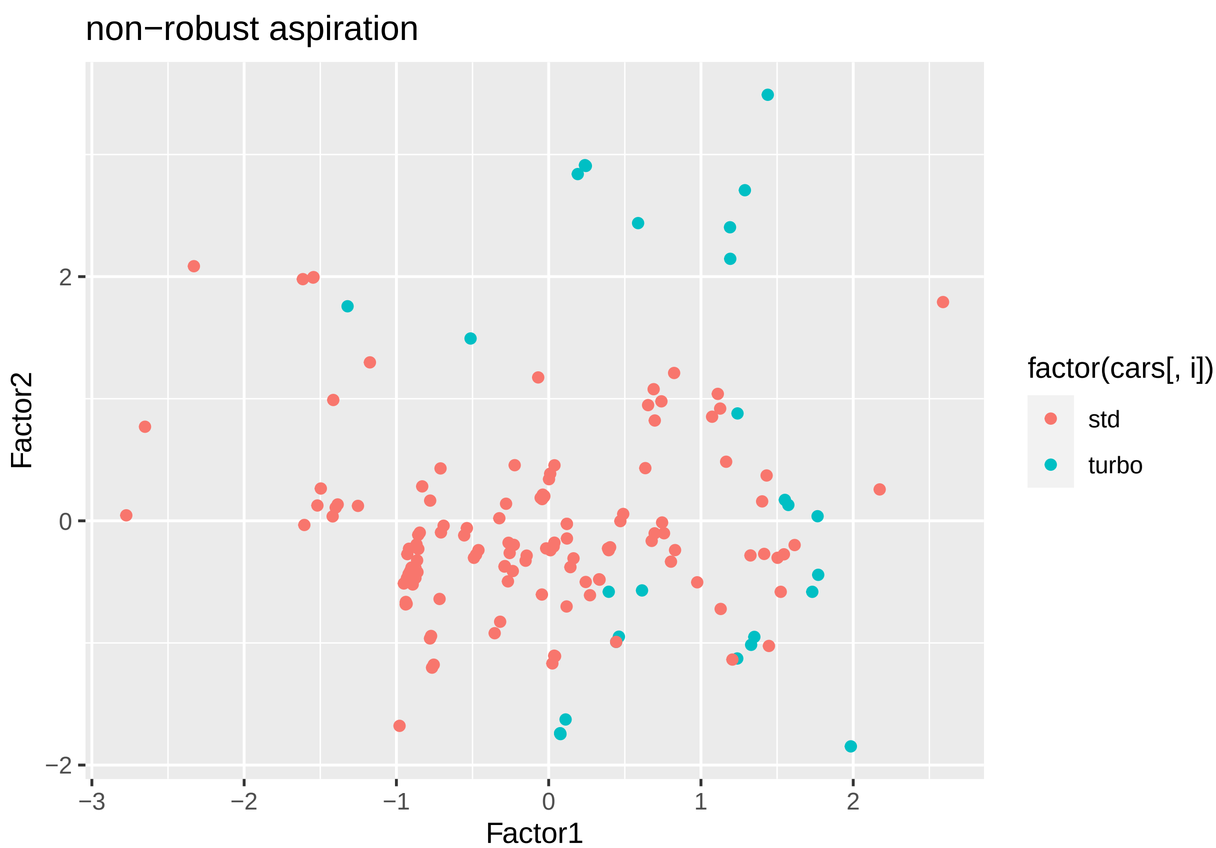

robust make
factor(cars[, 1])
audi
bmw
chevrolet
dodge
honda
jaguar
mazda
mercedes−benz
mitsubishi
nissan
peugot
plymouth
porsche
saab
subaru
toyota
volkswagen
volvo
Factor2
Factor1
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5


non−robust make
Factor2
Factor1
2
0
−2
−3
−2
−1
0
1
2
audi
bmw
chevrolet
dodge
honda
jaguar
mazda
mercedes−benz
mitsubishi
nissan
peugot
plymouth
porsche
saab
subaru
toyota
volkswagen
volvo

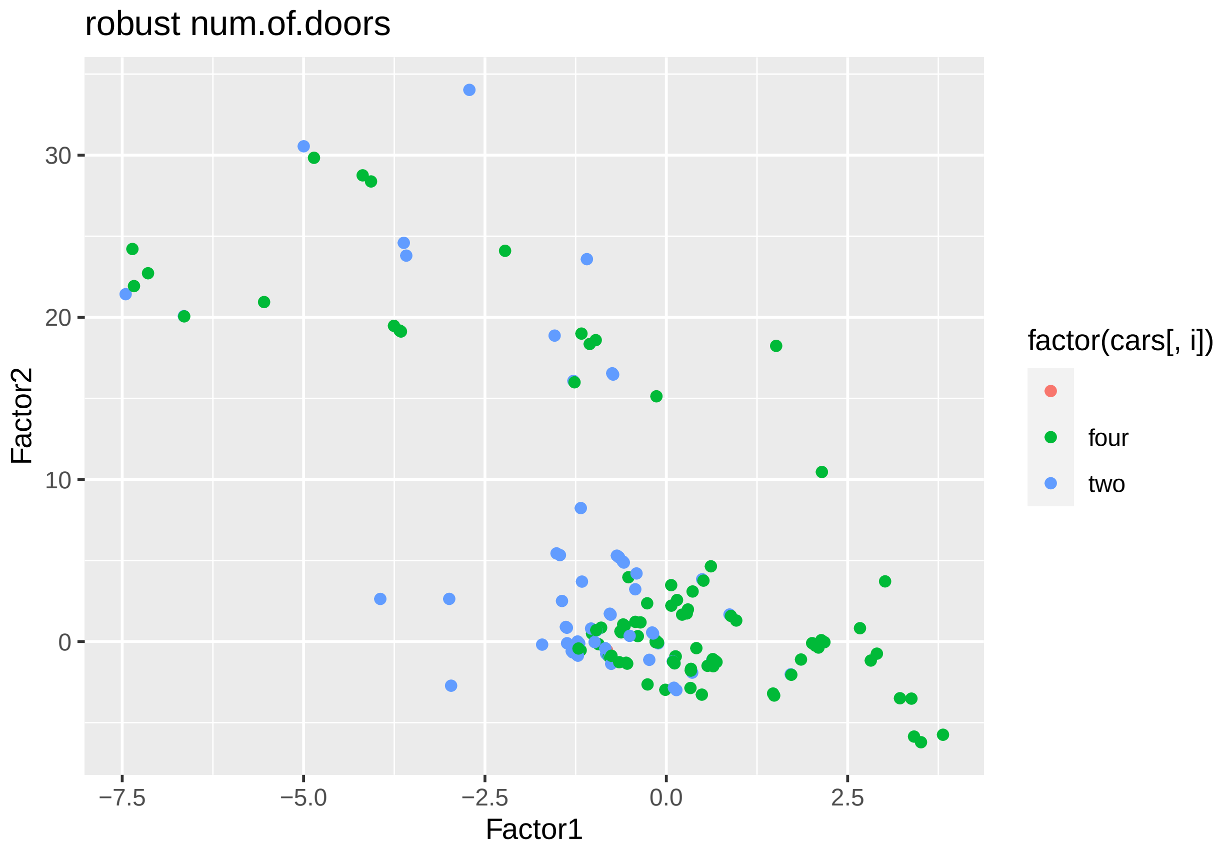

robust fuel.type
Factor2
Factor1
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
factor(cars[, i])
diesel
gas


non−robust fuel.type
Factor2
Factor1
2
0
−2
−3
−2
−1
0
1
2
factor(cars[, i])
diesel
gas

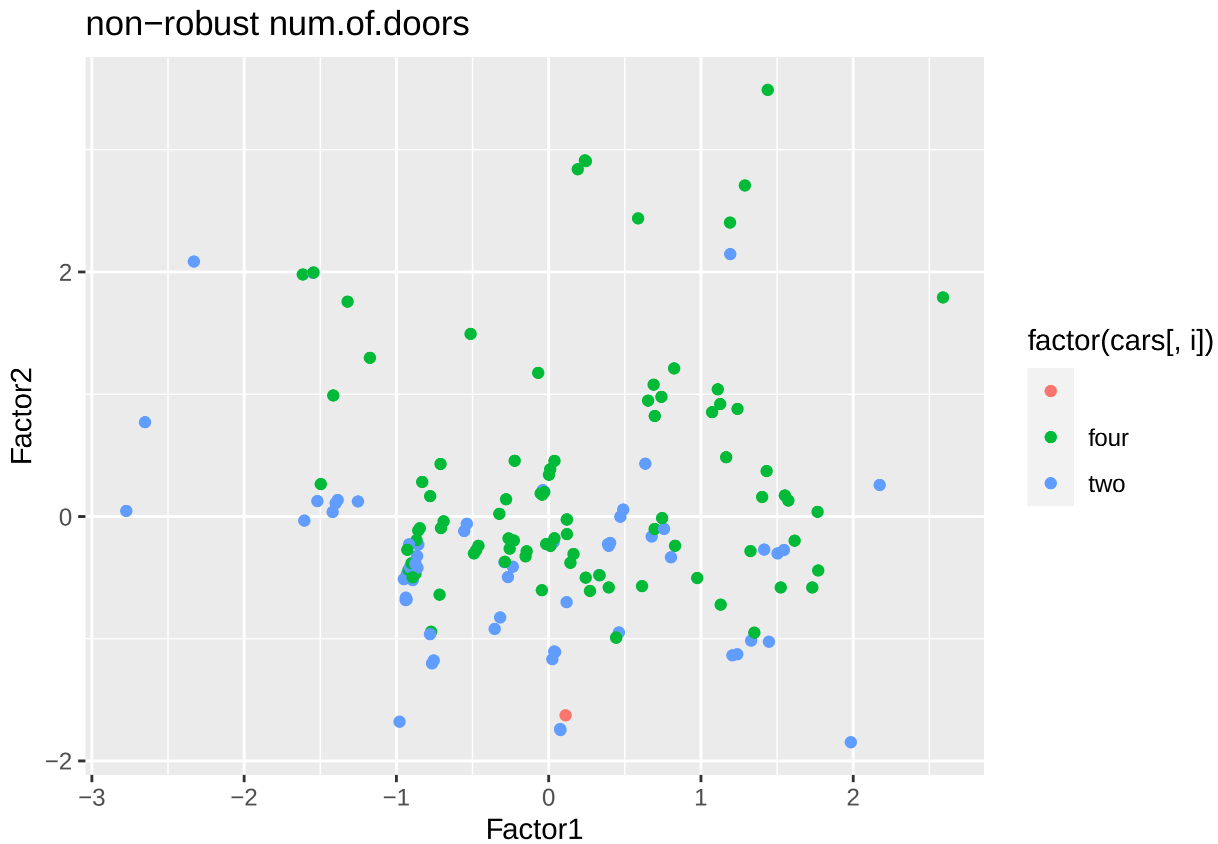

robust aspiration
Factor2
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
Factor1
factor(cars[, i])
std
turbo

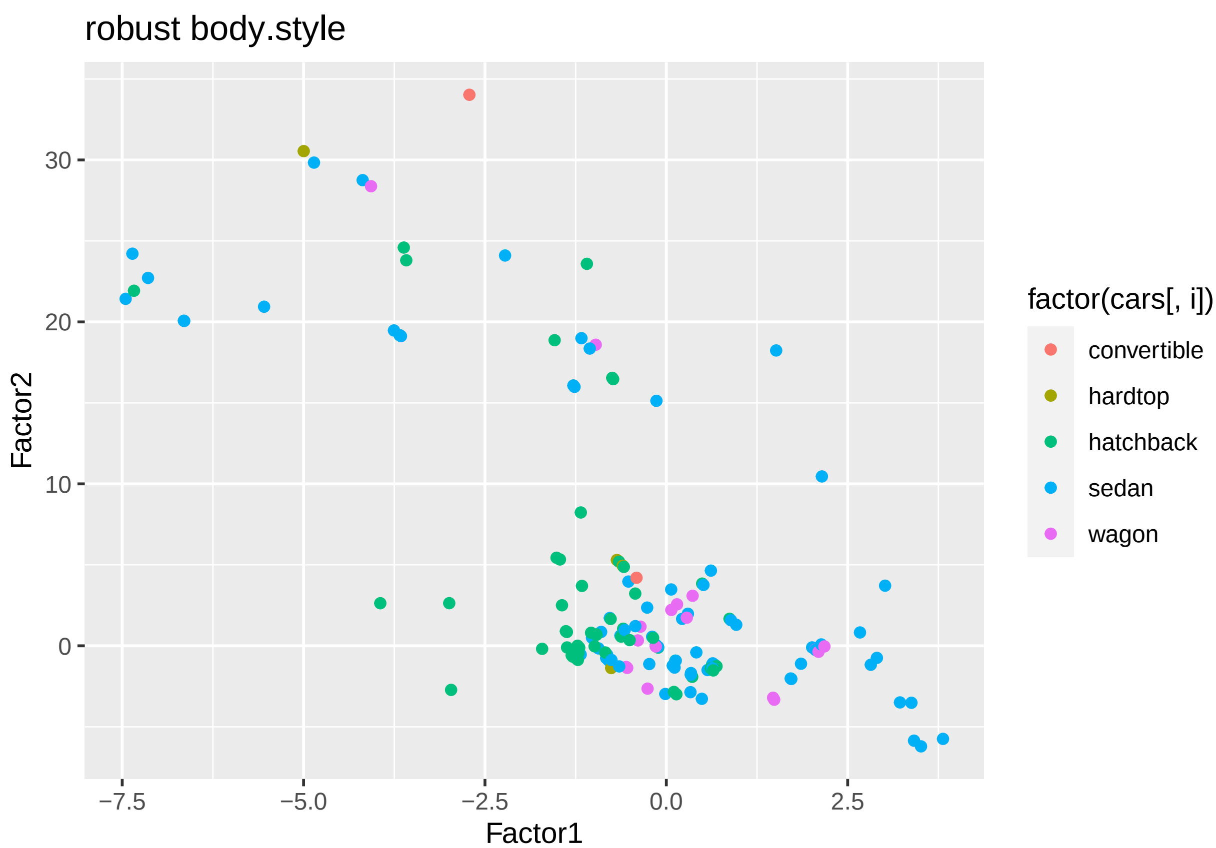


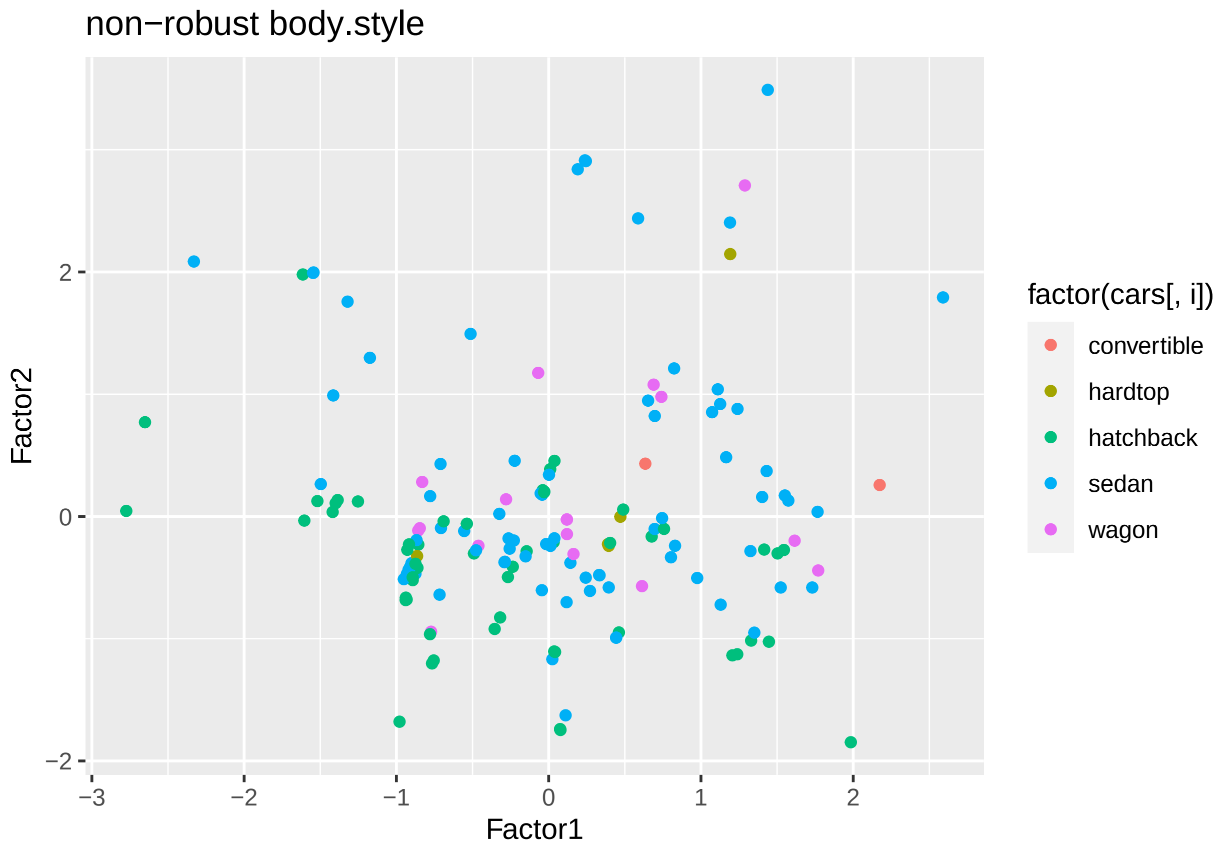
non−robust aspiration
Factor2
Factor1
2
0
−2
−3
−2
−1
0
1
2
factor(cars[, i])
std
turbo

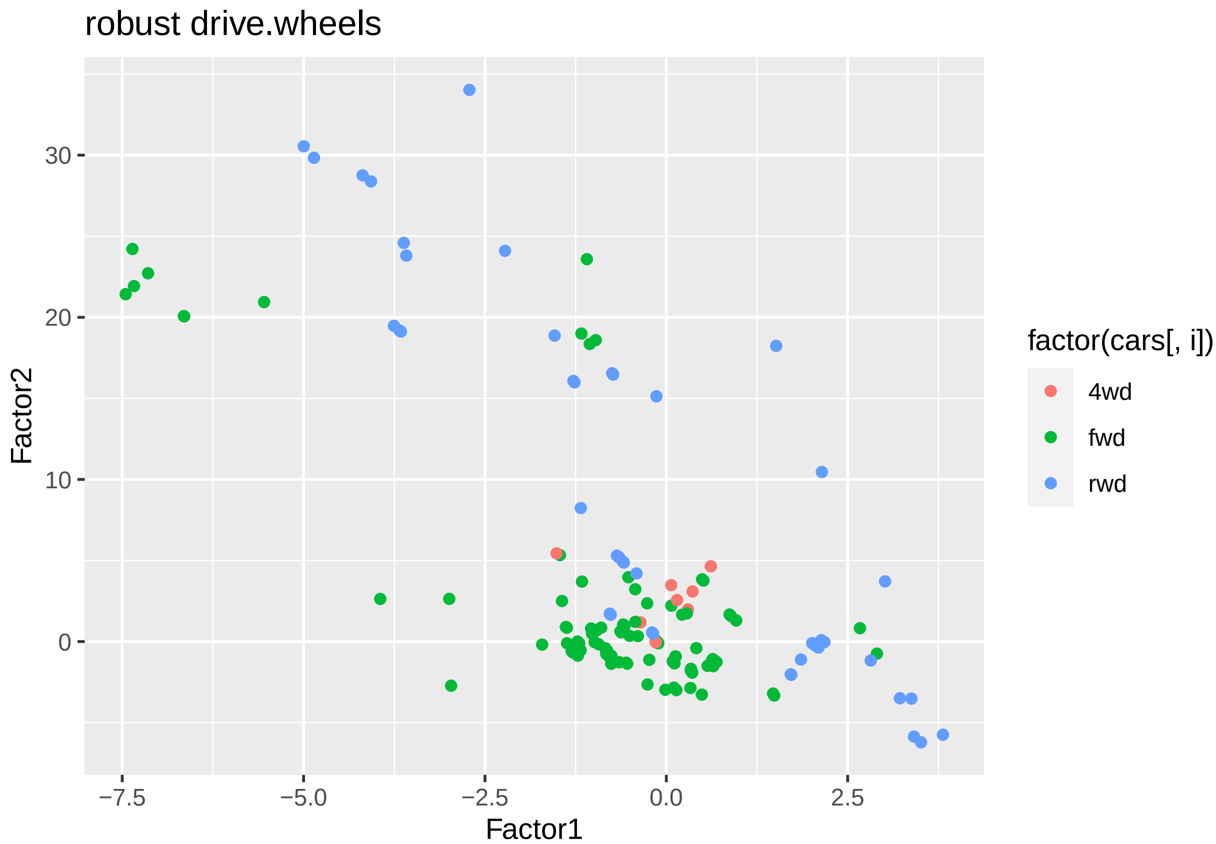


robust num.of.doors
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
Factor1
Factor2
factor(cars[, i])
four
two

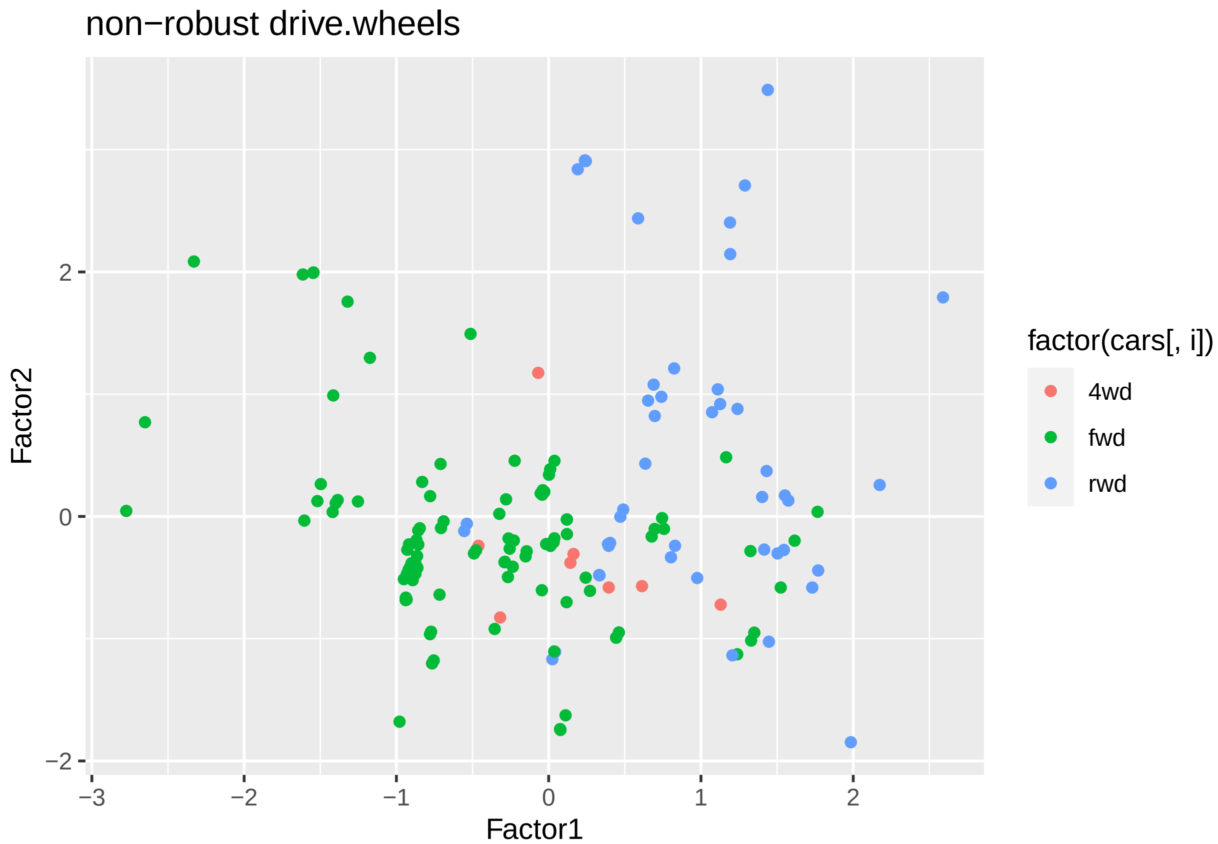

non−robust num.of.doors
Factor2
2
0
−2
−3
−2
−1
0
1
2
Factor1
factor(cars[, i])
four
two

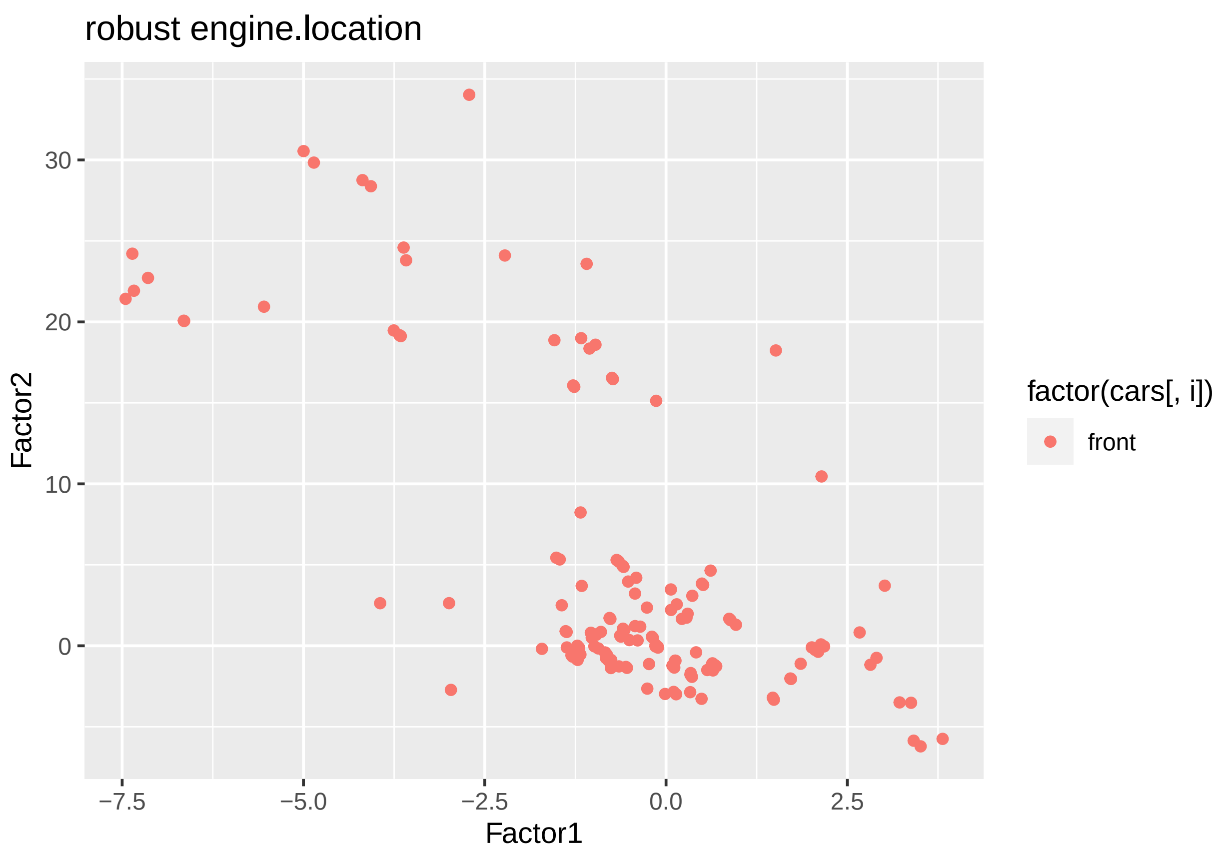

robust body.style
Factor2
Factor1
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
factor(cars[, i])
convertible
hardtop
hatchback
sedan
wagon


non−robust body.style
Factor2
Factor1
2
0
−2
−3
−2
−1
0
1
2
factor(cars[, i])
convertible
hardtop
hatchback
sedan
wagon

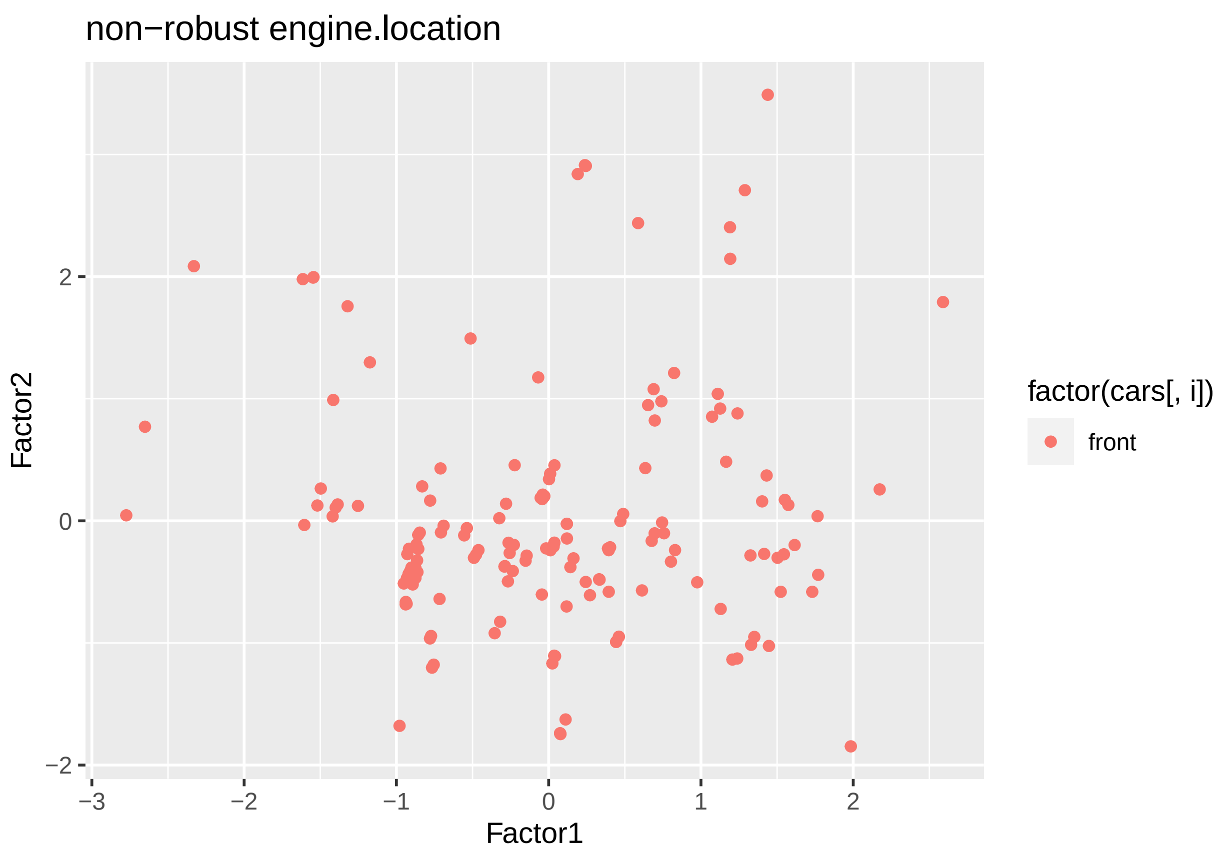


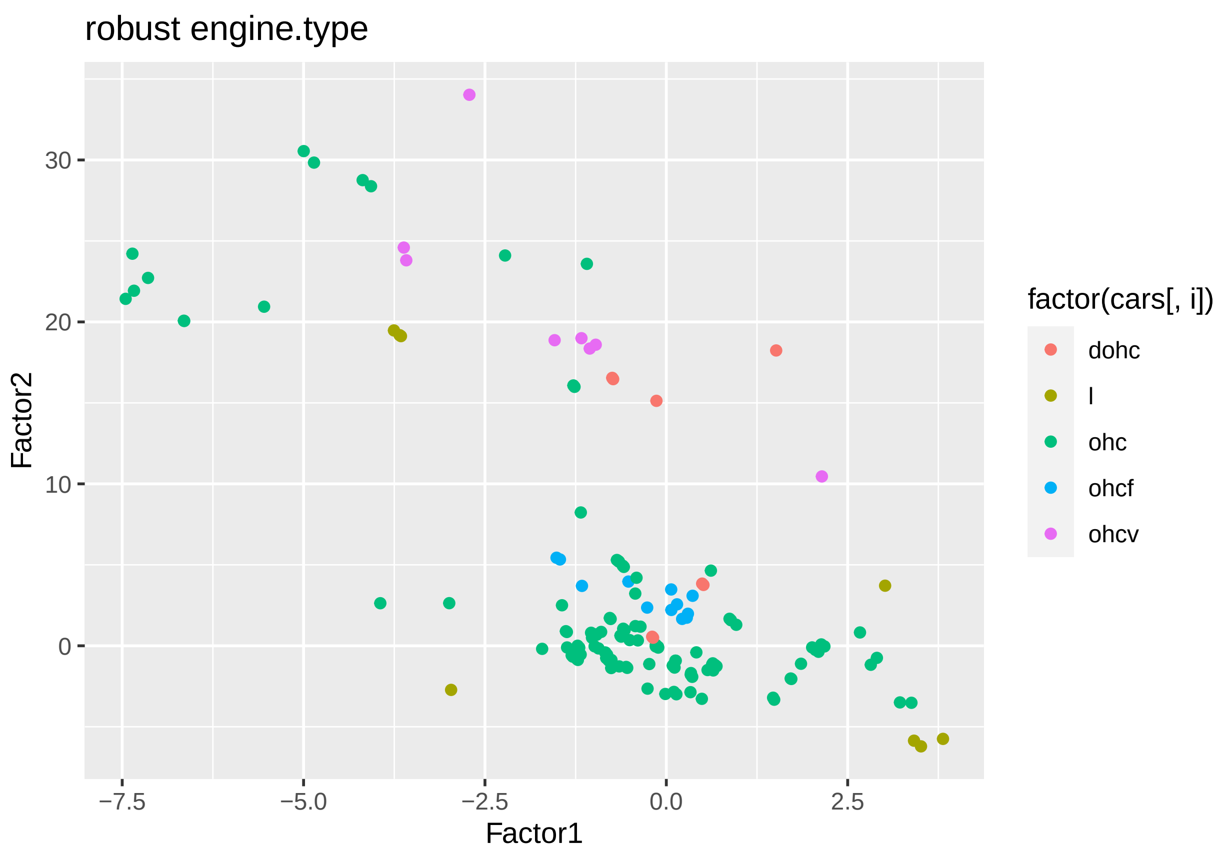
robust drive.wheels
Factor2
Factor1
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
factor(cars[, i])
4wd
fwd
rwd


non−robust drive.wheels
Factor2
Factor1
2
0
−2
−3
−2
−1
0
1
2
factor(cars[, i])
4wd
fwd
rwd

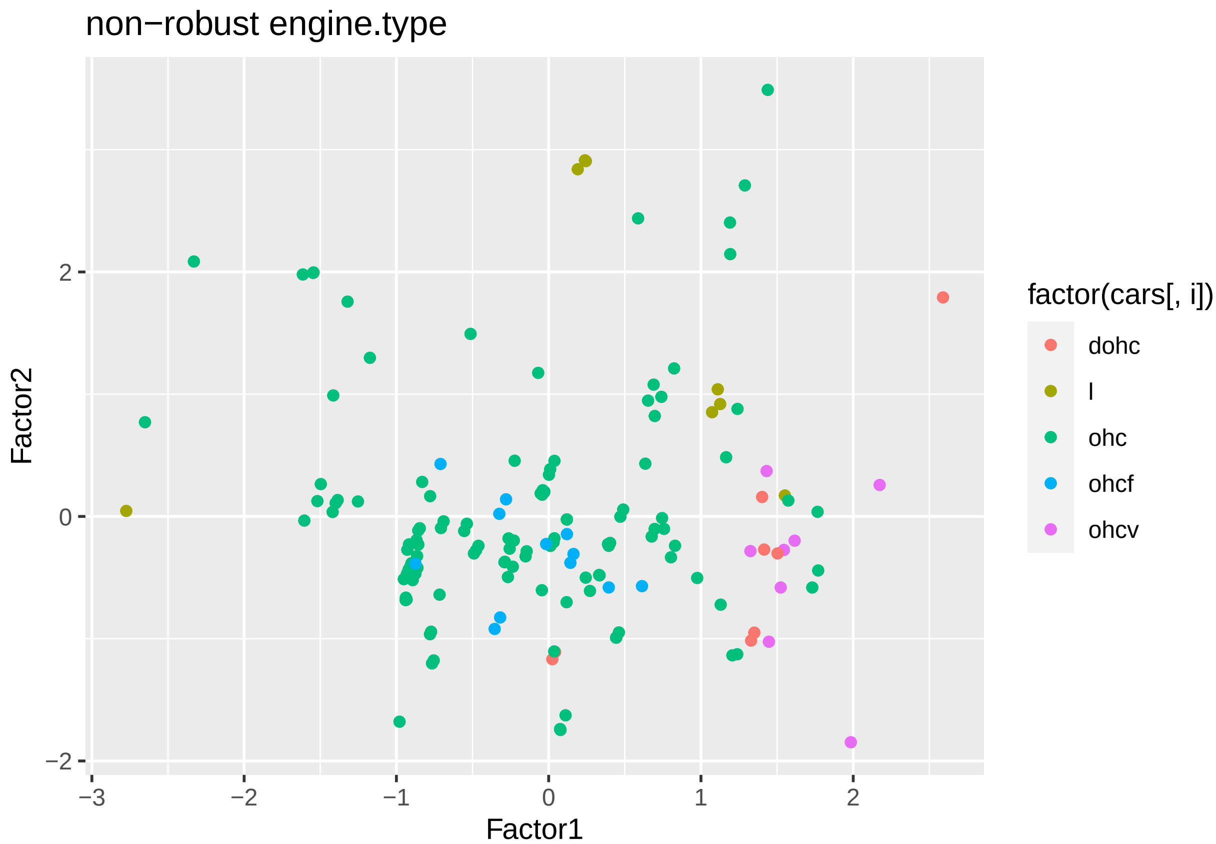


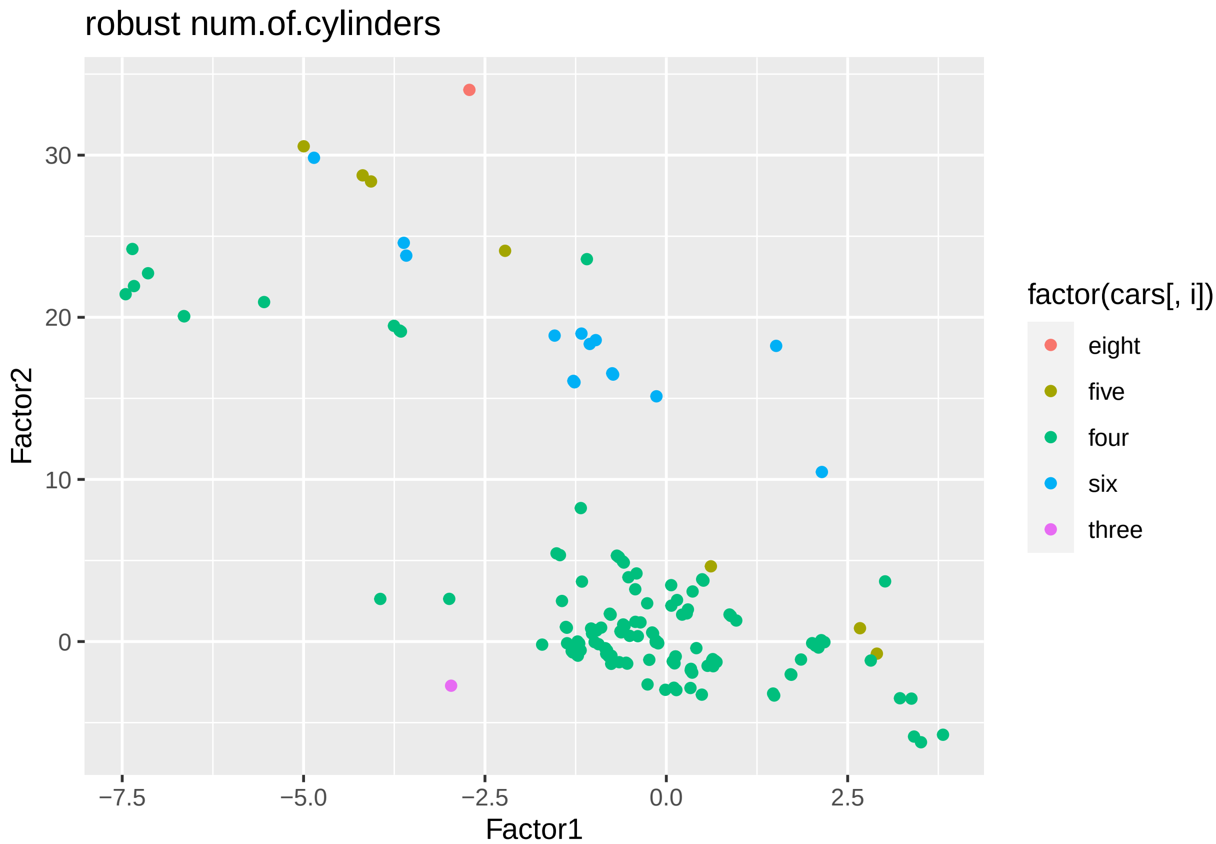
robust engine.location
Factor2
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
Factor1
factor(cars[, i])
front

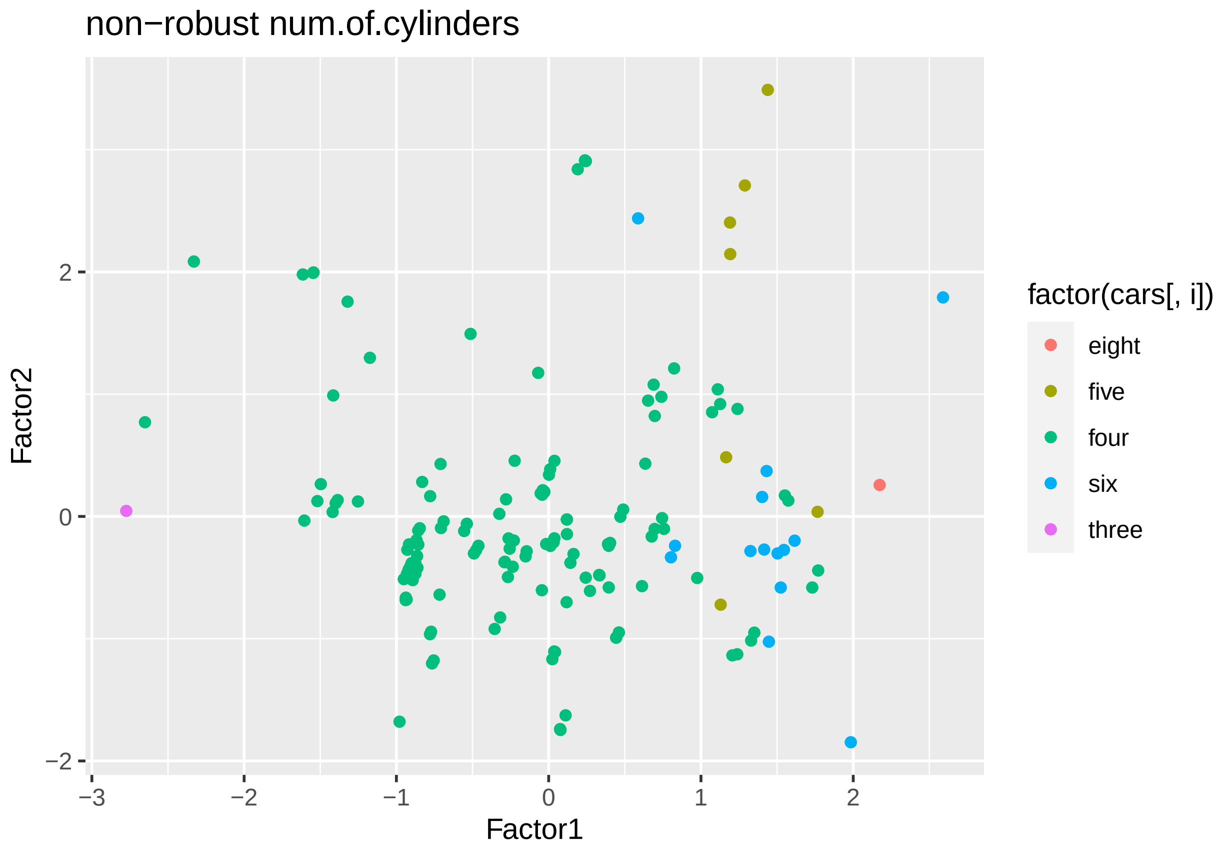

non−robust engine.location
Factor2
2
0
−2
−3
−2
−1
0
1
2
Factor1
factor(cars[, i])
front

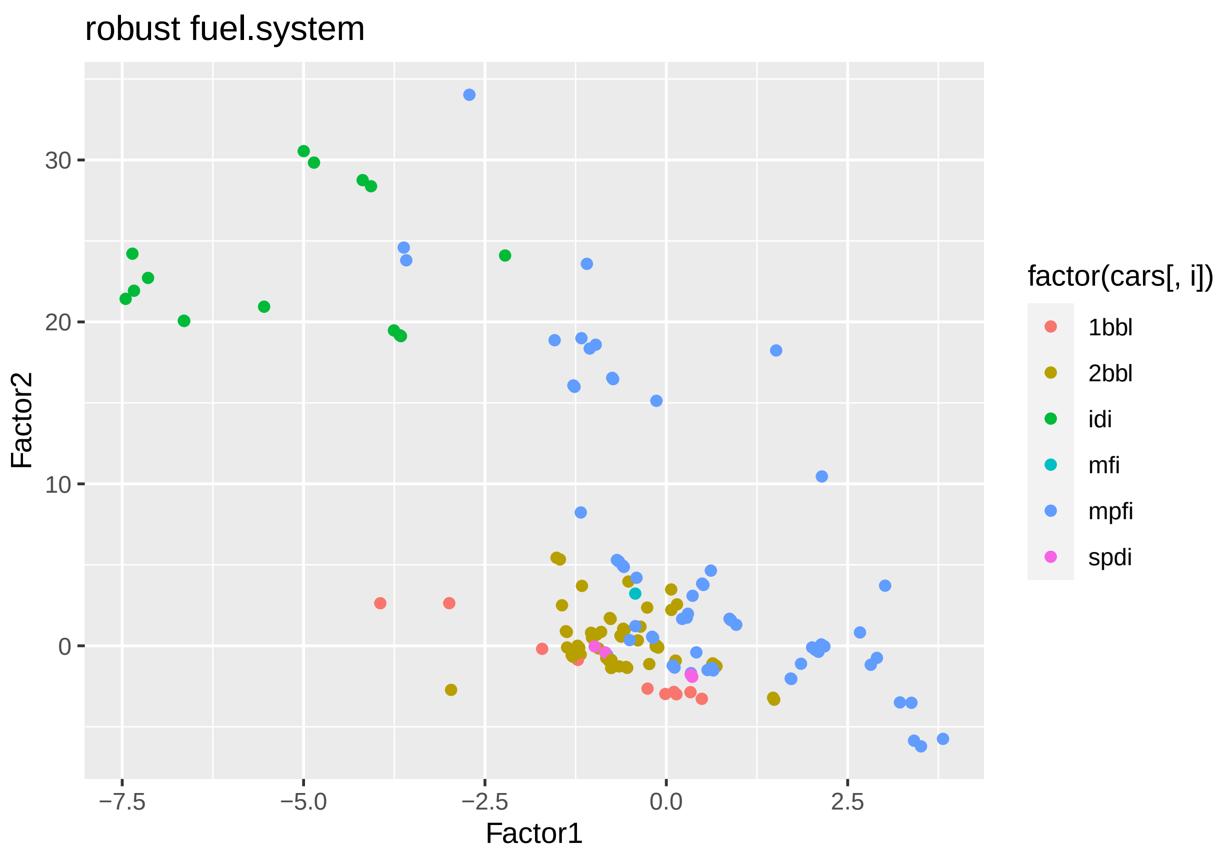


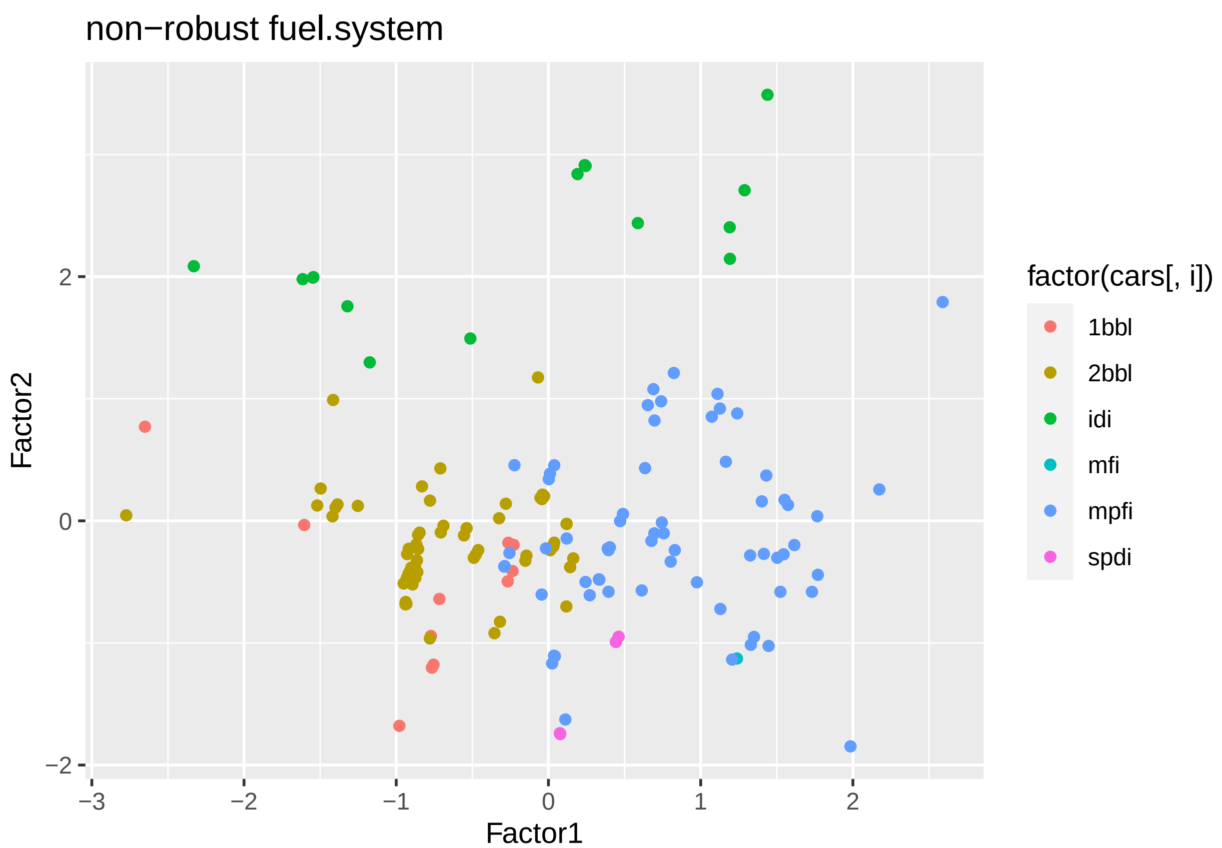
robust engine.type
Factor2
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
Factor1
factor(cars[, i])
dohc
l
ohc
ohcf
ohcv


non−robust engine.type
Factor2
Factor1
−3
−2
−1
0
1
2
−2
0
2
factor(cars[, i])
dohc
l
ohc
ohcf
ohcv


robust num.of.cylinders
Factor2
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
Factor1
factor(cars[, i])
eight
five
four
six
three


non−robust num.of.cylinders
Factor2
Factor1
2
0
−2
−3
−2
−1
0
1
2
factor(cars[, i])
eight
five
four
six
three


robust fuel.system
Factor2
Factor1
30
20
10
0
−7.5
−5.0
−2.5
0.0
2.5
factor(cars[, i])
1bbl
2bbl
idi
mfi
mpfi
spdi



The one outlier on the top left is the only one where the variable *body.style* is "convertible". Also The *num.of.cylinders* is eight for this observation only. In the non- robust method this is not detectable.

4

With print() applied to the factor analysis output object you can see the variance proportions of the factors. How are these values computed?

```
print(robust_pfa)
```

```
##
## Call:
## pfa(x = df_robscaled, factors = 2, covmat = covmat, scores = "regression")
##
## Uniquenesses:
##        wheel.base            length             width            height
##             0.123             0.111             0.154             0.486
##       curb.weight       engine.size              bore            stroke
##             0.049             0.098             0.245             0.892
## compression.ratio        horsepower          peak.rpm          city.mpg
##             0.714             0.106             0.792             0.115
##       highway.mpg             price
##             0.081             0.211
##
## Loadings:
##                   Factor1 Factor2
## wheel.base         0.936
```

```
## length             0.939
## width              0.911   0.124
## height             0.668  -0.261
## curb.weight        0.921   0.320
## engine.size        0.926   0.211
## bore               0.867
## stroke                     0.327
## compression.ratio         -0.533
## horsepower         0.626   0.708
## peak.rpm          -0.292   0.351
## city.mpg          -0.644  -0.686
## highway.mpg       -0.680  -0.676
## price              0.826   0.327
##
##                Factor1 Factor2
## SS loadings      7.530   2.293
## Proportion Var   0.538   0.164
## Cumulative Var   0.538   0.702
##
## The degrees of freedom for the model is 64 and the fit was NA
```

5

Compute the robust principal components based on the MCD estimator, and focus on the first two components. Rotate the components according to the varimax criterion (which is also the default for pfa()). This can be done by using the function varimax() from the package GPArotation.

```r
library(GPArotation)
```

```
## Warning: package 'GPArotation' was built under R version 4.3.2
```

```r
robust_pca = prcomp(~., data=data.frame(df_robscaled), covmat=covmat)
```

```
## Warning: In prcomp.default(x, ...) :
##  extra argument 'covmat' will be disregarded
```

```r
summary(robust_pca)
```

```
## Importance of components:
##                          PC1    PC2     PC3    PC4     PC5     PC6     PC7
## Standard deviation     6.635 3.3721 1.46392 1.1215 0.99795 0.70272 0.61711
## Proportion of Variance 0.714 0.1844 0.03476 0.0204 0.01615 0.00801 0.00618
## Cumulative Proportion  0.714 0.8984 0.93317 0.9536 0.96972 0.97773 0.98390
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.52822 0.45348 0.42848 0.36615 0.33932 0.22296 0.15896
## Proportion of Variance 0.00453 0.00334 0.00298 0.00217 0.00187 0.00081 0.00041
## Cumulative Proportion  0.98843 0.99176 0.99474 0.99692 0.99878 0.99959 1.00000
```

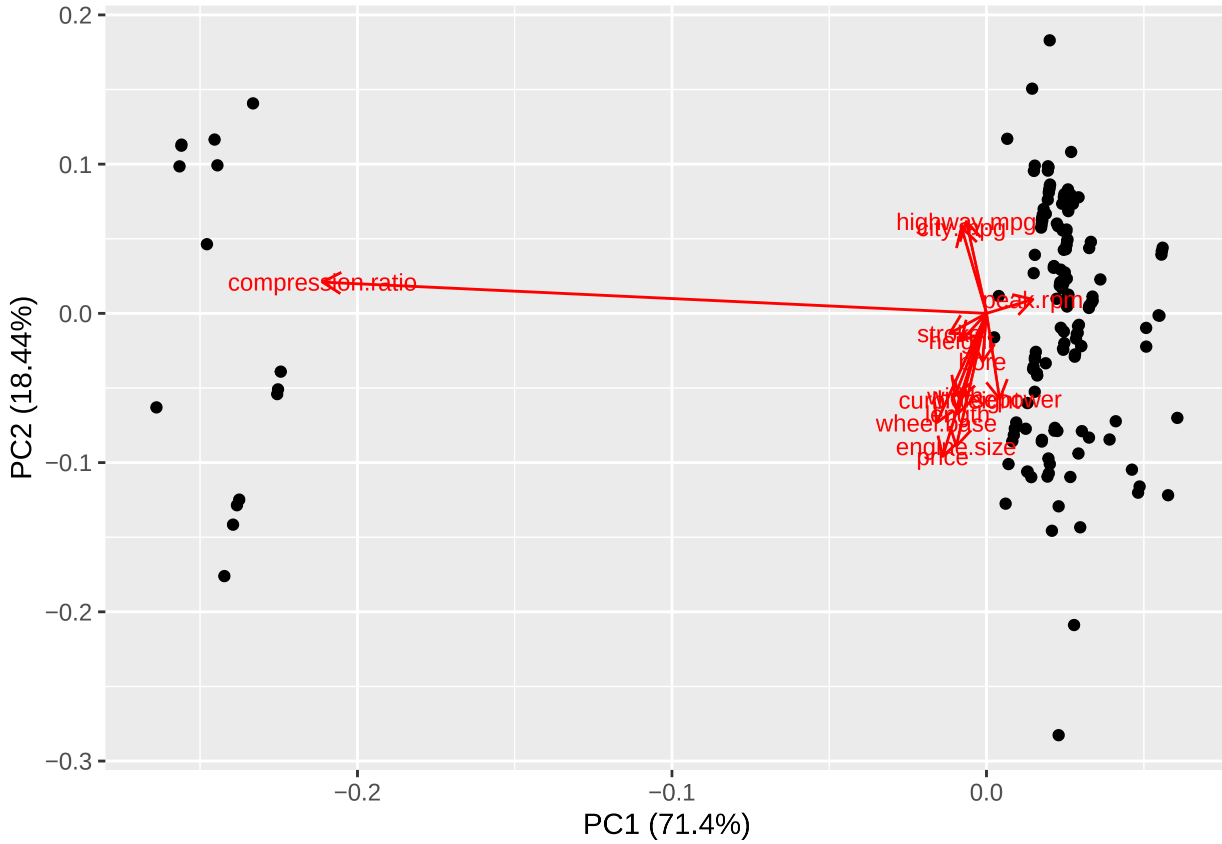
6

Compute also the scores to the rotated principal components, and present loadings and scores in a biplot. What are major differences to the robust factor analysis solution?

```r
varimax_loadings = varimax(robust_pca$rotation)$loadings
varimax_loadings
```

```
##
## Loadings:
##                   PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13 PC14
## wheel.base                    -1
## length                                                   1
## width                                                         -1
## height                                    -1
## curb.weight                                                              1
## engine.size           -1
## bore                                                 1
## stroke                    1
## compression.ratio -1
## horsepower                                                          1
## peak.rpm                          -1
## city.mpg                                                                      1
## highway.mpg                                   -1
## price                                 -1
##
##                  PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10
## SS loadings    1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## Proportion Var 0.071 0.071 0.071 0.071 0.071 0.071 0.071 0.071 0.071 0.071
## Cumulative Var 0.071 0.143 0.214 0.286 0.357 0.429 0.500 0.571 0.643 0.714
##                 PC11  PC12  PC13  PC14
## SS loadings    1.000 1.000 1.000 1.000
## Proportion Var 0.071 0.071 0.071 0.071
## Cumulative Var 0.786 0.857 0.929 1.000
```

```r
scores = df_robscaled %*% t(solve(varimax_loadings))
autoplot(robust_pca, data=df_robscaled, loadings = TRUE, loadings.colour = 'red',
         loadings.label = TRUE, loadings.label.size = 3)
```



Here compression ratio makes up most of the first component.