

Regression HD sur les données gasoline

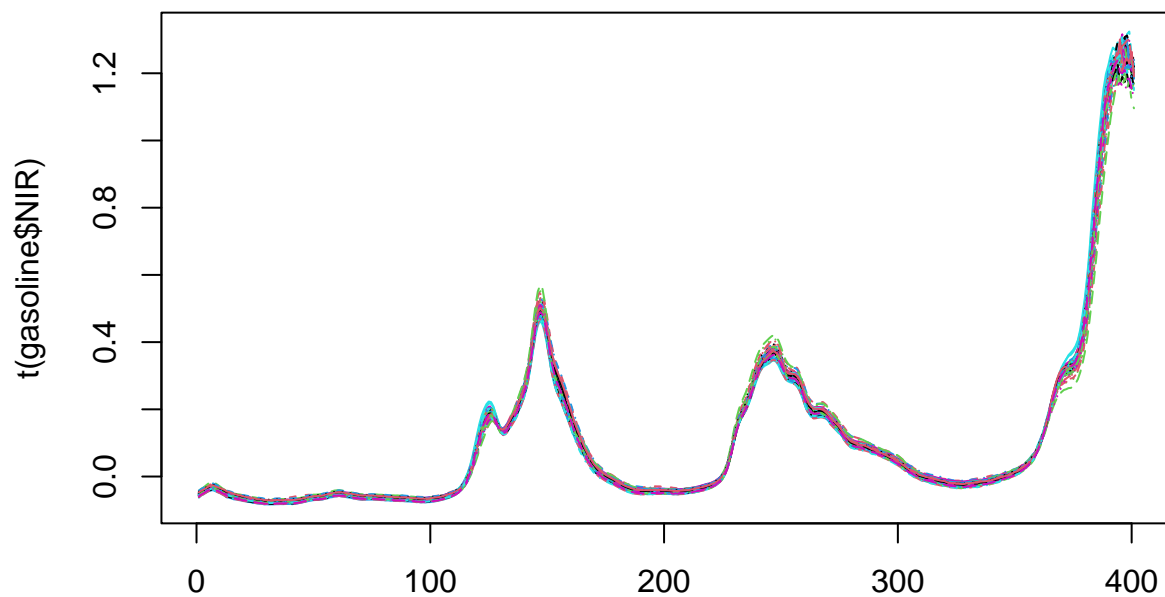
Julien JACQUES

09/10/2018

Chargeons les données gasolines (spectres NIR de 60 essences)

```
library(pls)
```

```
library(pls)
data("gasoline")
matplot(t(gasoline$NIR), type='l')
```



Régression linéaire

```
model1=lm(octane~NIR, data=gasoline)
print(anova(model1))
```

```
## Warning in anova.lm(model1): Les tests F d'ANOVA sur un ajustement pratiquement
## parfait ne sont pas fiables
```

```
## Analysis of Variance Table
```

```
##
```

```
## Response: octane
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
```

```
## NIR         59 138.13   2.3411    NaN    NaN
```

```
## Residuals    0   0.00     NaN
```

Plus de variables que d'individus, la régression moindres carrés est impossible.

Régression linéaire avec selection forward

C'est un peu technique à mettre en place quand il y a plus de variables que d'individus

```
x=cbind(gasoline$NIR,gasoline$octane)
data=data.frame(octane=x[,402],x[,1:401])
min.model <- lm(octane ~ 1, data=data)
tmp=""
for (n in seq(900,1700,2)) tmp=paste(tmp, ' + X',n, '.nm',sep='')
model1b <- step(min.model, direction = "forward", scope = (tmp),steps=50,trace = F)
summary(model1b)
```

```
##
## Call:
## lm(formula = octane ~ X1208.nm + X1196.nm + X976.nm + X1692.nm +
##      X970.nm + X1206.nm + X1056.nm + X1074.nm + X1098.nm + X1686.nm +
##      X1700.nm + X1544.nm + X1546.nm + X1094.nm + X1552.nm + X1274.nm +
##      X926.nm + X1342.nm + X940.nm + X918.nm + X1026.nm + X1064.nm +
##      X1600.nm + X1280.nm + X1278.nm + X960.nm + X1104.nm + X1320.nm +
##      X1588.nm + X900.nm + X1084.nm + X1080.nm + X1622.nm + X1076.nm +
##      X1318.nm + X1590.nm + X1550.nm + X1680.nm + X1072.nm + X1628.nm +
##      X1570.nm + X932.nm + X1562.nm + X1240.nm + X1328.nm + X1338.nm +
##      X938.nm + X912.nm + X1304.nm + X914.nm, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.403e-03 -6.654e-04 -2.852e-05  5.119e-04  2.370e-03
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   93.95791    0.41417  226.858  < 2e-16 ***
## X1208.nm      58.68357    2.70155   21.722 4.38e-09 ***
## X1196.nm      43.82832    0.70867   61.846 3.81e-13 ***
## X976.nm      282.47025    5.95384   47.443 4.11e-12 ***
## X1692.nm      -6.35868    0.04774 -133.201 3.85e-16 ***
## X970.nm     -145.20891   12.06119  -12.039 7.49e-07 ***
## X1206.nm     -112.98103    2.58944  -43.632 8.72e-12 ***
## X1056.nm     -147.86160    8.71903  -16.958 3.87e-08 ***
## X1074.nm      188.69706    8.22859   22.932 2.71e-09 ***
## X1098.nm      608.35982    9.63038   63.171 3.15e-13 ***
## X1686.nm      -9.57912    0.09472 -101.134 4.58e-15 ***
## X1700.nm       9.54631    0.11672   81.791 3.09e-14 ***
## X1544.nm     -67.11382    5.52940  -12.138 6.99e-07 ***
## X1546.nm      378.07592    5.24191   72.126 9.57e-14 ***
## X1094.nm     -616.08611    8.23124  -74.847 6.86e-14 ***
## X1552.nm     -262.06898    5.00999  -52.309 1.71e-12 ***
## X1274.nm     -282.47975    4.85978  -58.126 6.65e-13 ***
## X926.nm      -187.76911    4.16957  -45.033 6.56e-12 ***
## X1342.nm      273.65088    7.37360   37.112 3.71e-11 ***
## X940.nm       448.88799    8.05103   55.755 9.66e-13 ***
## X918.nm     -278.77900    2.45958 -113.344 1.64e-15 ***
## X1026.nm      195.02443    5.76319   33.840 8.47e-11 ***
## X1064.nm      496.40355    8.11150   61.198 4.19e-13 ***
## X1600.nm     -236.19192    2.87998  -82.012 3.02e-14 ***
## X1280.nm      132.07732    6.18331   21.360 5.08e-09 ***
```

```

## X1278.nm      -58.43611      5.26078     -11.108  1.48e-06 ***
## X960.nm       -262.43222      6.08237     -43.146  9.63e-12 ***
## X1104.nm      -270.86228     10.56058     -25.648  1.00e-09 ***
## X1320.nm      -339.27261      6.40133     -53.000  1.52e-12 ***
## X1588.nm       84.75666      4.49798      18.843  1.53e-08 ***
## X900.nm       -29.28231      1.92112     -15.242  9.81e-08 ***
## X1084.nm      -133.30216     13.11668     -10.163  3.13e-06 ***
## X1080.nm       233.95072      6.78697      34.471  7.19e-11 ***
## X1622.nm       93.24525      1.87888      49.628  2.75e-12 ***
## X1076.nm      -519.53341     14.91602     -34.831  6.55e-11 ***
## X1318.nm       249.64750      7.82555      31.902  1.44e-10 ***
## X1590.nm       105.94648      4.65838      22.743  2.91e-09 ***
## X1550.nm      -87.44534      6.33800     -13.797  2.33e-07 ***
## X1680.nm       -1.27874      0.07273     -17.583  2.82e-08 ***
## X1072.nm       176.73360      6.80680      25.964  8.99e-10 ***
## X1628.nm      -31.18749      0.75996     -41.038  1.51e-11 ***
## X1570.nm      -60.26094      2.22769     -27.051  6.24e-10 ***
## X932.nm        -44.19893      7.69392      -5.745  0.000278 ***
## X1562.nm       63.49092      4.97996      12.749  4.59e-07 ***
## X1240.nm      -32.14533      2.11160     -15.223  9.92e-08 ***
## X1328.nm       86.97972      5.65207      15.389  9.03e-08 ***
## X1338.nm      -65.47727      4.90213     -13.357  3.08e-07 ***
## X938.nm        23.74105      5.77676       4.110  0.002638 **
## X912.nm        33.19357      3.40430       9.750  4.41e-06 ***
## X1304.nm       25.86886      6.14606       4.209  0.002276 **
## X914.nm       -14.72095      4.00124      -3.679  0.005083 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.002377 on 9 degrees of freedom
## Multiple R-squared:  1, Adjusted R-squared:  1
## F-statistic: 4.891e+05 on 50 and 9 DF, p-value: < 2.2e-16

```

On autorise au maximum 50 étapes (donc 50 variables dans le modèle) étant donné qu'on n'a que 60 individus.

Régression PCR

```

library(pls)
model2=pcr(octane~NIR,data=gasoline,validation='L00')
summary(model2)

## Data:      X dimension: 60 401
## Y dimension: 60 1
## Fit method: svdpc
## Number of components considered: 58
##
## VALIDATION: RMSEP
## Cross-validated using 60 leave-one-out segments.
##      (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.543   1.447   1.474   1.255   0.2501   0.2503   0.2578
## adjCV         1.543   1.446   1.474   1.255   0.2496   0.2500   0.2575
##      7 comps  8 comps  9 comps 10 comps 11 comps 12 comps 13 comps
## CV      0.2646  0.2724  0.2474  0.2508  0.2340  0.2255  0.2293
## adjCV    0.2643  0.2733  0.2471  0.2508  0.2336  0.2244  0.2286

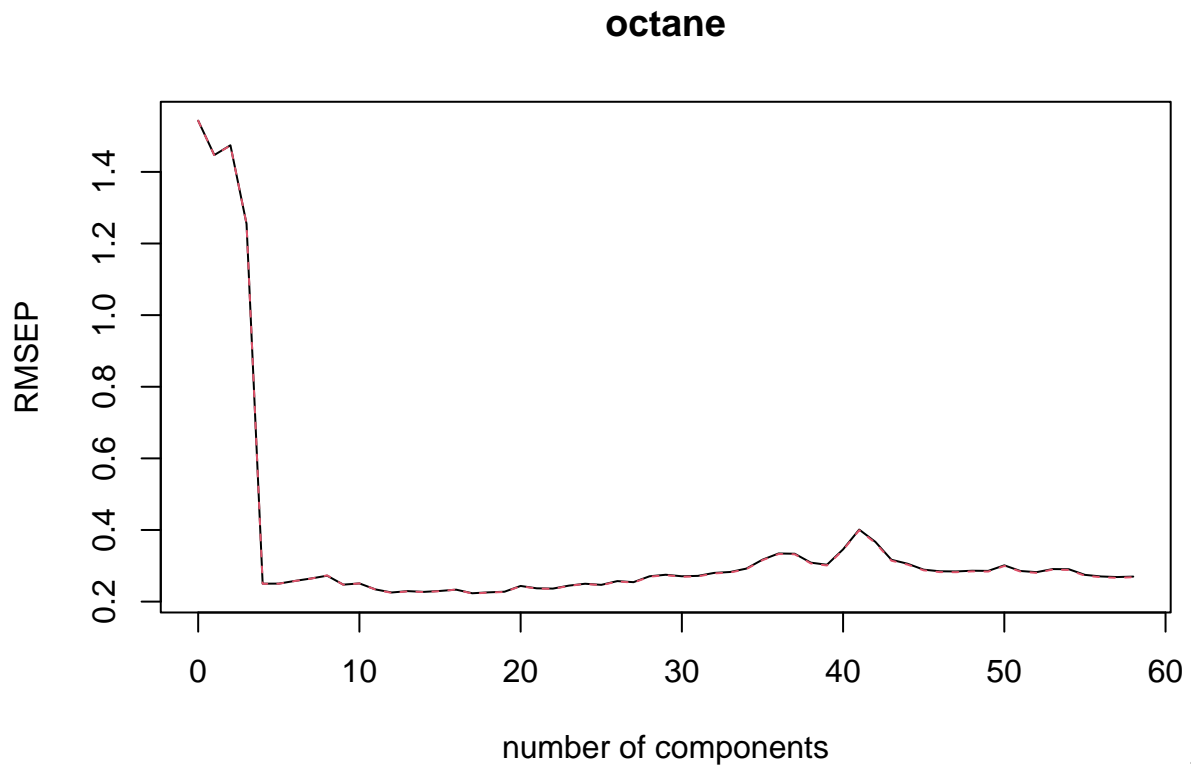
```

```

##      14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV      0.2272   0.2298   0.2334   0.2233   0.2259   0.2276   0.2437
## adjCV    0.2266   0.2292   0.2337   0.2225   0.2252   0.2270   0.2430
##      21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV      0.2371   0.2364   0.2447   0.2499   0.2468   0.2573   0.2543
## adjCV    0.2363   0.2356   0.2439   0.2495   0.2457   0.2565   0.2531
##      28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV      0.2707   0.2750   0.2707   0.2719   0.2802   0.2828   0.2922
## adjCV    0.2696   0.2738   0.2693   0.2704   0.2787   0.2813   0.2908
##      35 comps  36 comps  37 comps  38 comps  39 comps  40 comps  41 comps
## CV      0.3171   0.3345   0.3337   0.3088   0.3029   0.3459   0.4011
## adjCV    0.3157   0.3329   0.3322   0.3066   0.3009   0.3441   0.3994
##      42 comps  43 comps  44 comps  45 comps  46 comps  47 comps  48 comps
## CV      0.3670   0.3169   0.3058   0.2892   0.2847   0.2842   0.2863
## adjCV    0.3639   0.3143   0.3033   0.2873   0.2824   0.2822   0.2844
##      49 comps  50 comps  51 comps  52 comps  53 comps  54 comps  55 comps
## CV      0.2858   0.3013   0.2855   0.2823   0.2908   0.2904   0.2747
## adjCV    0.2839   0.2995   0.2840   0.2801   0.2896   0.2883   0.2724
##      56 comps  57 comps  58 comps
## CV      0.2702   0.2686   0.2699
## adjCV    0.2681   0.2664   0.2682
##
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      72.57   83.90   90.86   95.46   96.70   97.66   98.16   98.52
## octane  18.99   19.62   46.50   97.69   97.78   97.79   97.79   97.79
##      9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X      98.85   99.09   99.29   99.40   99.51   99.60   99.68
## octane  98.33   98.38   98.72   98.86   98.87   98.89   98.93
##      16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X      99.73   99.79   99.84   99.86   99.89   99.90   99.92
## octane  98.93   99.03   99.03   99.03   99.05   99.08   99.10
##      23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X      99.93   99.94   99.95   99.96   99.96   99.97   99.97
## octane  99.12   99.13   99.22   99.24   99.31   99.31   99.34
##      30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X      99.98   99.98   99.98   99.98   99.99   99.99   99.99
## octane  99.40   99.41   99.41   99.42   99.42   99.43   99.47
##      37 comps  38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X      99.99   99.99   99.99   99.99   99.99   99.99   100.00
## octane  99.53   99.61   99.63   99.63   99.66   99.81   99.83
##      44 comps  45 comps  46 comps  47 comps  48 comps  49 comps  50 comps
## X      100.00   100.00   100.00   100.00   100.00   100.00   100.00
## octane  99.84   99.85   99.87   99.87   99.87   99.88   99.88
##      51 comps  52 comps  53 comps  54 comps  55 comps  56 comps  57 comps
## X      100.00   100.00   100.00   100.00   100.00   100.00   100.00
## octane  99.91   99.93   99.94   99.97   99.98   99.98   99.99
##      58 comps
## X      100.00
## octane  99.99

```

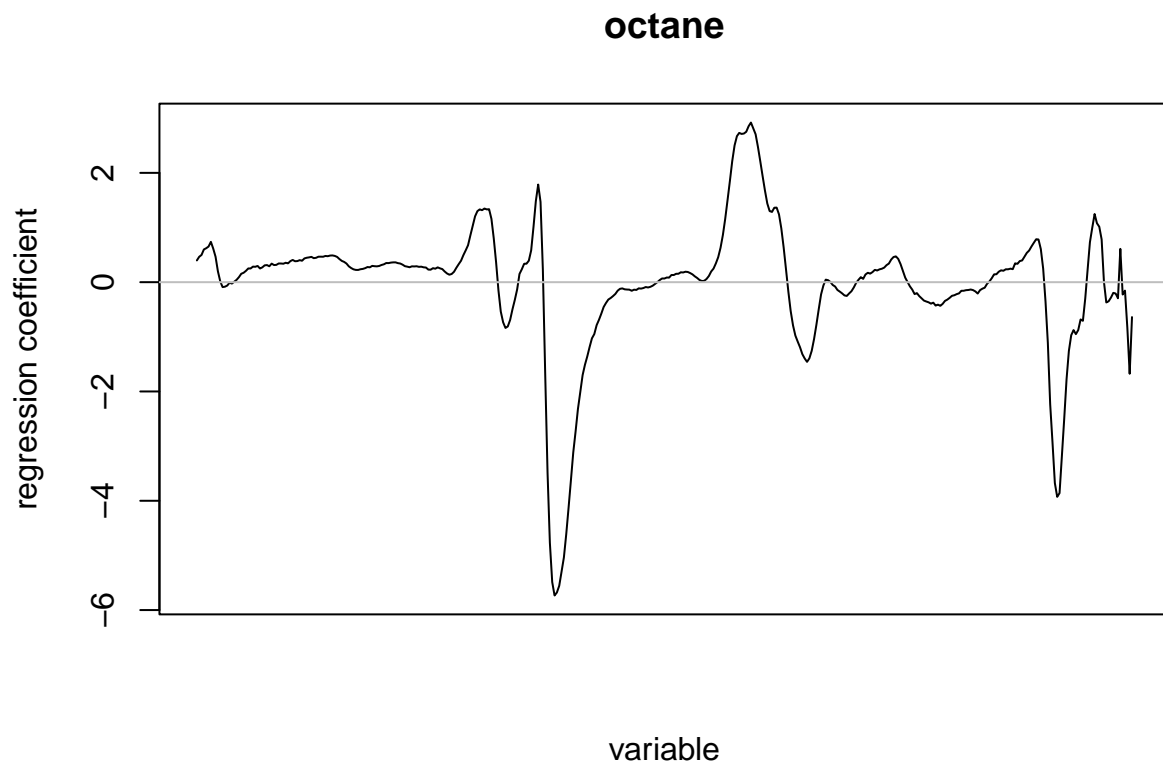
```
plot(RMSEP(model2))
```



posantes semblent le plus efficaces.

On peut tracer les coefficients obtenus avec 4 composantes

```
plot(model2, plottype = "coef", ncomp=4,xaxt='n')
```



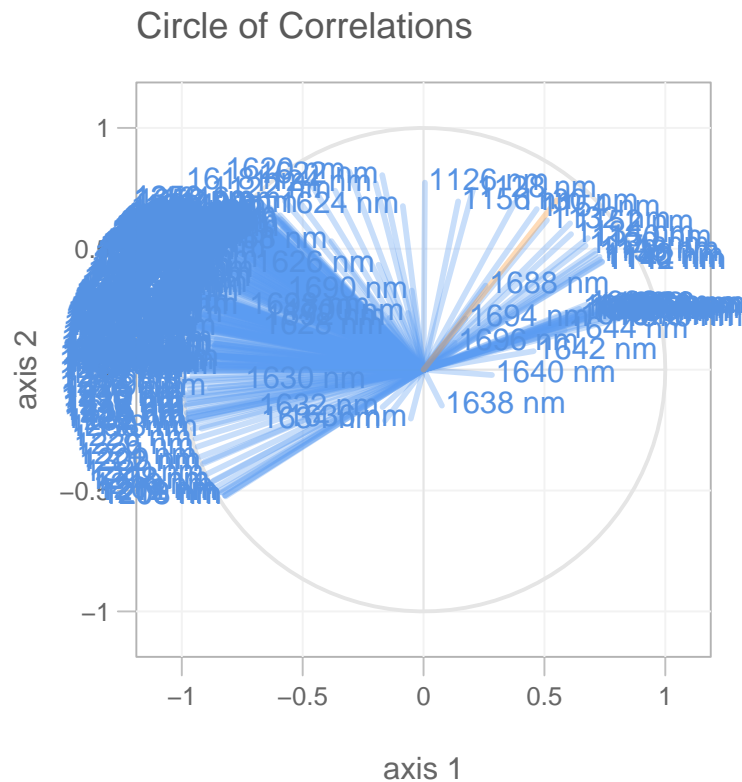
Régression PLS

```
library(plsdepot)
x=cbind(gasoline$NIR,gasoline$octane)
tmp=plsreg1(x[,1:401],x[,402,drop=F],comps=10,crosval=T)
print(tmp$Q2)
```

##	PRESS	RSS	Q2	LimQ2	Q2cum
## 1	42.3575468	59.0000000	0.28207548	0.0975	0.2820755
## 2	13.9025768	40.9797905	0.66074554	0.0975	0.7564409
## 3	1.4166532	11.9217690	0.88117089	0.0975	0.9710581
## 4	1.1395100	1.3381513	0.14844456	0.0975	0.9753544
## 5	0.8361958	1.0226792	0.18234792	0.0975	0.9798484
## 6	0.8397470	0.7828962	-0.07261601	0.0975	0.9783851
## 7	0.7819730	0.6485442	-0.20573591	0.0975	0.9739382
## 8	0.8045042	0.5629184	-0.42916674	0.0975	0.9627533
## 9	0.5170719	0.4180883	-0.23675269	0.0975	0.9539350
## 10	0.3387020	0.3327009	-0.01803728	0.0975	0.9531041

On va conserver 5 composantes PLS, ré-estimons le modèle

```
model2=plsreg1(x[,1:401],x[,402,drop=F],comps=5,crosval=T)
plot(model2,what='variables',comps=c(1,2))
```



```
VIP=matrix(0,401,5)
for (j in 1:401){
  for (h in 1:5){
    VIP[j,h]=sqrt(401/sum(model2$R2[1:h])*sum(model2$R2[1:h]*(model2$raw.wgs[j,1:h]^2)))
  }
}
rownames(VIP)=colnames(x)[1:401]
```

Un $VIP > .8$ est généralement signe que la variable a une importance sur la prediction

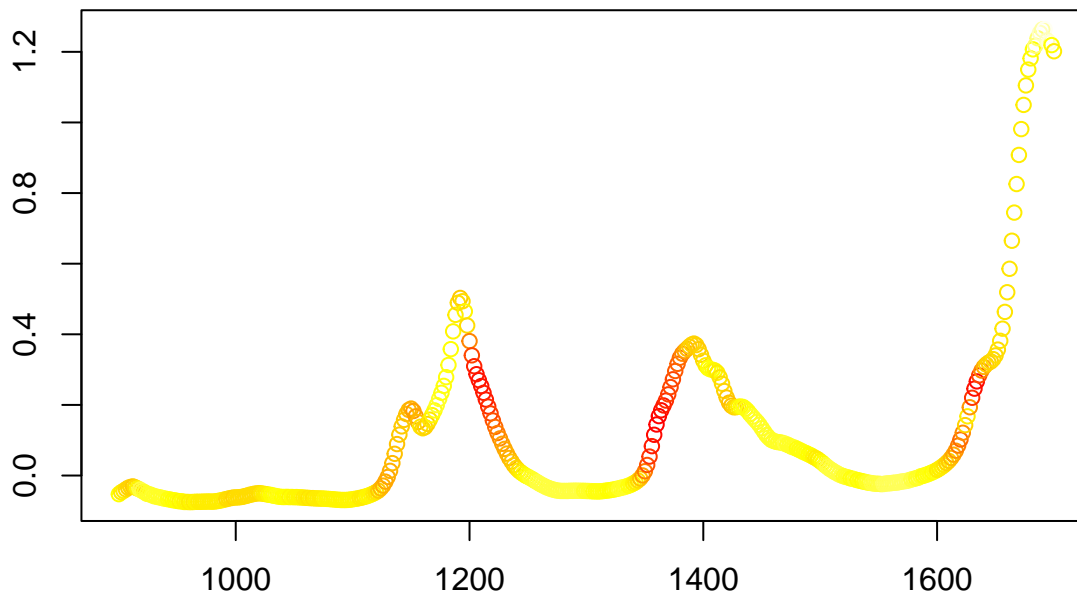
```
vip5=VIP[,5]
print(sort(vip5,decreasing = T)[1:50])
```

```
## 1634 nm 1636 nm 1362 nm 1360 nm 1358 nm 1364 nm 1638 nm 1208 nm
## 1.983240 1.977585 1.932420 1.928871 1.900198 1.894388 1.881106 1.865221
## 1632 nm 1206 nm 1210 nm 1212 nm 1356 nm 1366 nm 1214 nm 1204 nm
## 1.863573 1.858243 1.852577 1.812422 1.804902 1.803425 1.796586 1.791321
## 1216 nm 1368 nm 1354 nm 1630 nm 1218 nm 1370 nm 1202 nm 1372 nm
## 1.736321 1.723010 1.721043 1.699718 1.670232 1.662528 1.639609 1.631660
## 1640 nm 1374 nm 1220 nm 1352 nm 1376 nm 1378 nm 1222 nm 1620 nm
## 1.622554 1.613363 1.587454 1.576908 1.566747 1.522408 1.488407 1.473901
## 1380 nm 1350 nm 1224 nm 1618 nm 1642 nm 1200 nm 1382 nm 1226 nm
## 1.462477 1.421538 1.419367 1.393152 1.363824 1.355776 1.355456 1.326368
## 1622 nm 1616 nm 1628 nm 1228 nm 1384 nm 1154 nm 1348 nm 1152 nm
## 1.324735 1.291240 1.272816 1.270583 1.251501 1.242518 1.239138 1.236150
## 1128 nm 1130 nm
## 1.223943 1.216564
```

Beaucoup de variables ont des VIP importants.

Pour ce type de données spectrales, on peut par exemple représenter graphiquement la courbe moyenne en mettant en couleur plus ou moins chaude suivant si le VIP est important (rouge = variables importantes)

```
vip5_normalises=(vip5-min(vip5))/(max(vip5)-min(vip5))
plot(seq(900,1700,2),colMeans(gasoline$NIR),col=heat.colors(40)[40*(1-vip5_normalises)],xlab="",ylab="")
```

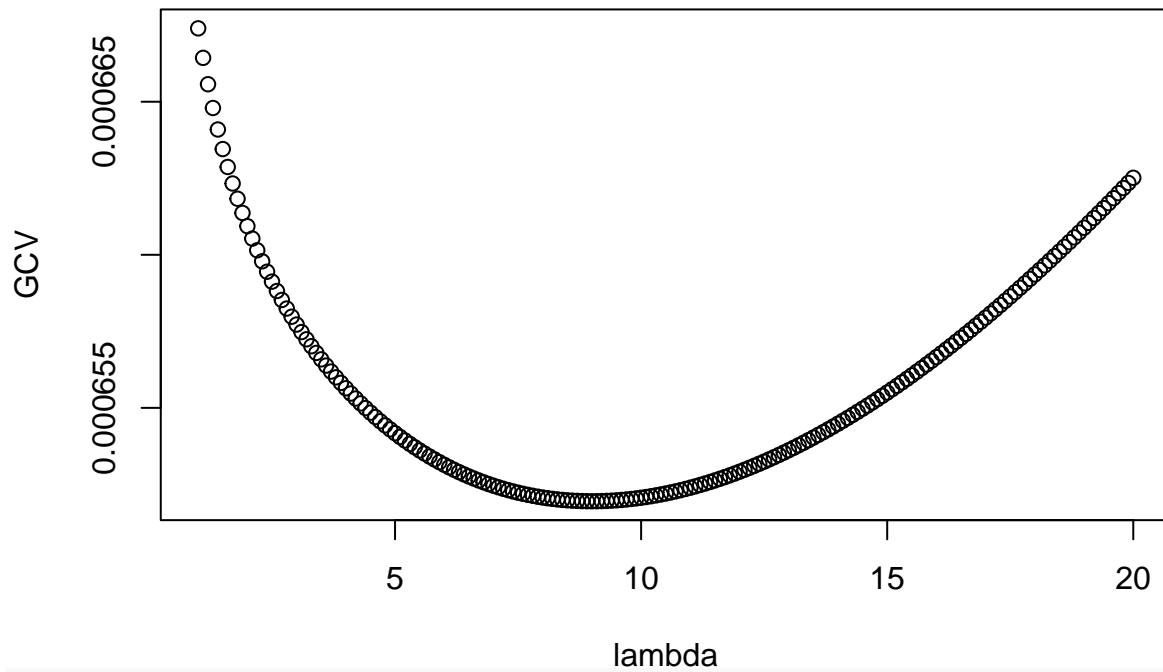


Régression ridge

Effectuons la régression ridge et cherchons le meilleur lambda.

```
model_ride <- lm.ride(octane~NIR,data=gasoline,lambda=seq(1,20,0.1))
plot(seq(1,20,0.1),model_ride$GCV,xlab='lambda',ylab='GCV',main='GCV')
```

GCV



```
print(model_ride$lambda[which.min(model_ride$GCV)])
```

```
## [1] 9
print(min(model_ride$GCV))
```

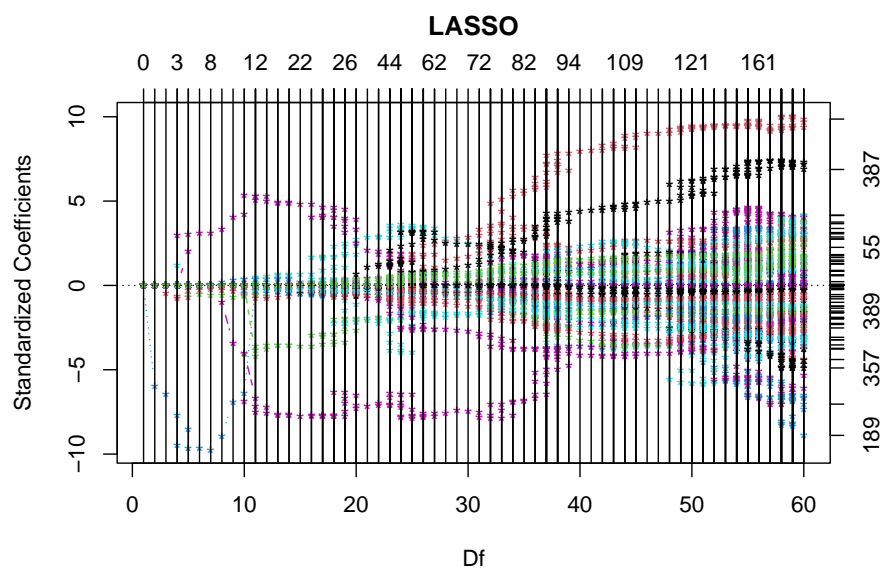
```
## [1] 0.0006519503
```

D'après l'indice de validation croisée généralisée (GCV), le lambda optimal est 9. Ré-estimons alors le modèle.

```
model_ride <- lm.ride(octane~NIR,data=gasoline,lambda=9)
```

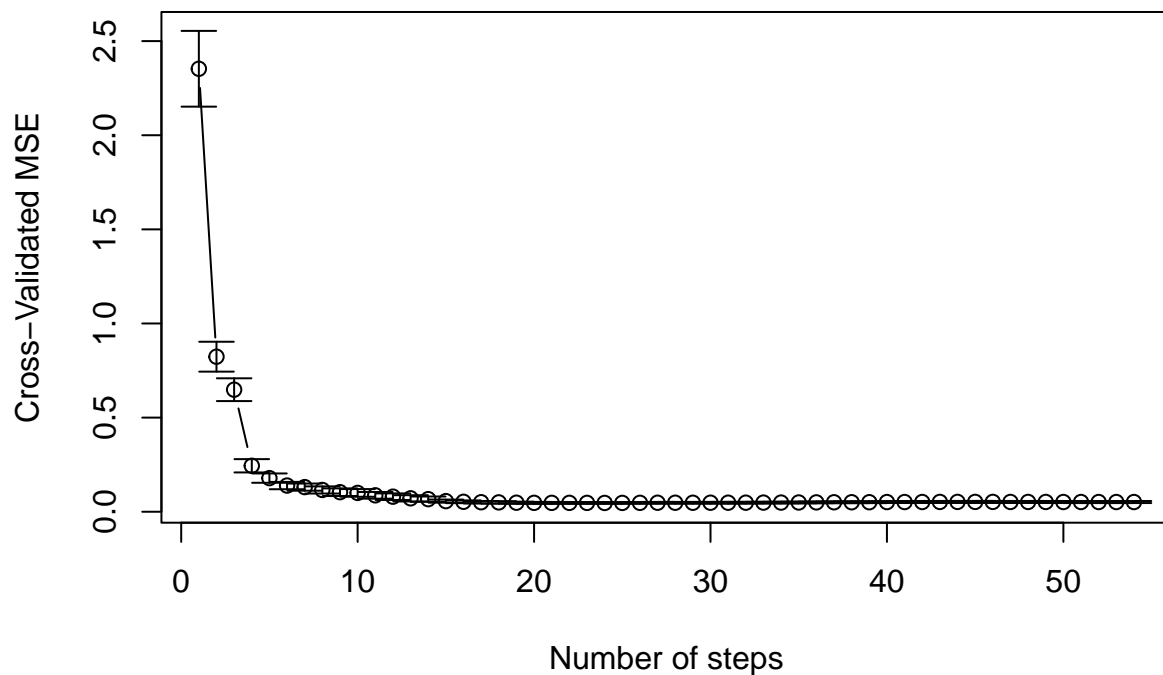
Régression lasso

```
model_lasso <- lars(gasoline$NIR,gasoline$octane,type="lasso",trace=F,normalize=TRUE)
plot(model_lasso,xvar="df", plottype='coeff')
```



Cherchons le lambda optimal


```
cv=cv.lars(gasoline$NIR,gasoline$octane, K = 10,trace = FALSE, plot.it = TRUE, se = TRUE,type = c("lasso"),mode='step')
```



On peut afficher le lambda optimal d'après le CVMSE

```
print(model_lasso$lambda[which.min(cv$cv)])
```

```
## [1] 0.08226631
```

où encore prendre le lambda qui conduit à la solution la plus parcimonieuse avec une erreur inférieur à l'erreur minimal + l'erreur d'estimation de cette erreur :

```
tmp=which.min(cv$cv)
tmp2=min(which(cv$cv < cv$cv[tmp]+cv$cv.error[tmp]))
print(model_lasso$lambda[tmp2])
```

```
## [1] 0.116896
```

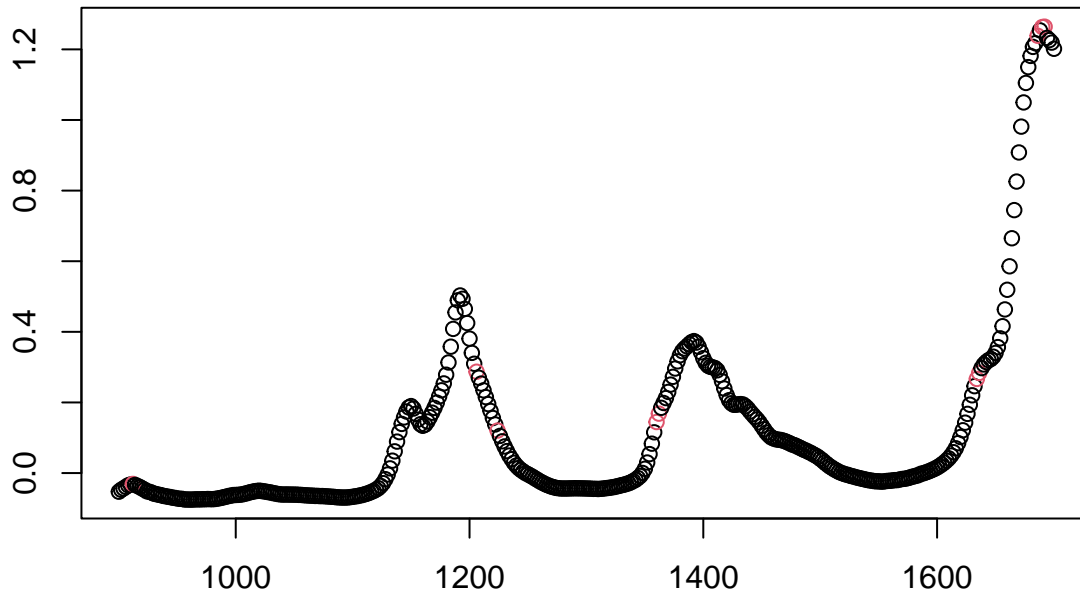
```
lambda_opt=model_lasso$lambda[tmp2]
print(model_lasso$beta[tmp2,])
```

```
##      900 nm      902 nm      904 nm      906 nm      908 nm      910 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      912 nm      914 nm      916 nm      918 nm      920 nm      922 nm
## 10.9480466 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      924 nm      926 nm      928 nm      930 nm      932 nm      934 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      936 nm      938 nm      940 nm      942 nm      944 nm      946 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      948 nm      950 nm      952 nm      954 nm      956 nm      958 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      960 nm      962 nm      964 nm      966 nm      968 nm      970 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      972 nm      974 nm      976 nm      978 nm      980 nm      982 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      984 nm      986 nm      988 nm      990 nm      992 nm      994 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      996 nm      998 nm      1000 nm      1002 nm      1004 nm      1006 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1008 nm      1010 nm      1012 nm      1014 nm      1016 nm      1018 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1020 nm      1022 nm      1024 nm      1026 nm      1028 nm      1030 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1032 nm      1034 nm      1036 nm      1038 nm      1040 nm      1042 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1044 nm      1046 nm      1048 nm      1050 nm      1052 nm      1054 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1056 nm      1058 nm      1060 nm      1062 nm      1064 nm      1066 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1068 nm      1070 nm      1072 nm      1074 nm      1076 nm      1078 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1080 nm      1082 nm      1084 nm      1086 nm      1088 nm      1090 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1092 nm      1094 nm      1096 nm      1098 nm      1100 nm      1102 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1104 nm      1106 nm      1108 nm      1110 nm      1112 nm      1114 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1116 nm      1118 nm      1120 nm      1122 nm      1124 nm      1126 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1128 nm      1130 nm      1132 nm      1134 nm      1136 nm      1138 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1140 nm      1142 nm      1144 nm      1146 nm      1148 nm      1150 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##      1152 nm      1154 nm      1156 nm      1158 nm      1160 nm      1162 nm
```

```
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1164 nm 1166 nm 1168 nm 1170 nm 1172 nm 1174 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1176 nm 1178 nm 1180 nm 1182 nm 1184 nm 1186 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1188 nm 1190 nm 1192 nm 1194 nm 1196 nm 1198 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1200 nm 1202 nm 1204 nm 1206 nm 1208 nm 1210 nm
## 0.0000000 0.0000000 0.0000000 -22.6751777 0.0000000 0.0000000
## 1212 nm 1214 nm 1216 nm 1218 nm 1220 nm 1222 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1224 nm 1226 nm 1228 nm 1230 nm 1232 nm 1234 nm
## -69.9167620 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1236 nm 1238 nm 1240 nm 1242 nm 1244 nm 1246 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1248 nm 1250 nm 1252 nm 1254 nm 1256 nm 1258 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1260 nm 1262 nm 1264 nm 1266 nm 1268 nm 1270 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1272 nm 1274 nm 1276 nm 1278 nm 1280 nm 1282 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1284 nm 1286 nm 1288 nm 1290 nm 1292 nm 1294 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1296 nm 1298 nm 1300 nm 1302 nm 1304 nm 1306 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1308 nm 1310 nm 1312 nm 1314 nm 1316 nm 1318 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1320 nm 1322 nm 1324 nm 1326 nm 1328 nm 1330 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1332 nm 1334 nm 1336 nm 1338 nm 1340 nm 1342 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1344 nm 1346 nm 1348 nm 1350 nm 1352 nm 1354 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1356 nm 1358 nm 1360 nm 1362 nm 1364 nm 1366 nm
## 0.0000000 0.0000000 6.9772124 69.6021056 0.0000000 0.0000000
## 1368 nm 1370 nm 1372 nm 1374 nm 1376 nm 1378 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1380 nm 1382 nm 1384 nm 1386 nm 1388 nm 1390 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1392 nm 1394 nm 1396 nm 1398 nm 1400 nm 1402 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1404 nm 1406 nm 1408 nm 1410 nm 1412 nm 1414 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1416 nm 1418 nm 1420 nm 1422 nm 1424 nm 1426 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1428 nm 1430 nm 1432 nm 1434 nm 1436 nm 1438 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1440 nm 1442 nm 1444 nm 1446 nm 1448 nm 1450 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1452 nm 1454 nm 1456 nm 1458 nm 1460 nm 1462 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1464 nm 1466 nm 1468 nm 1470 nm 1472 nm 1474 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1476 nm 1478 nm 1480 nm 1482 nm 1484 nm 1486 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1488 nm 1490 nm 1492 nm 1494 nm 1496 nm 1498 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1500 nm 1502 nm 1504 nm 1506 nm 1508 nm 1510 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1512 nm 1514 nm 1516 nm 1518 nm 1520 nm 1522 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1524 nm 1526 nm 1528 nm 1530 nm 1532 nm 1534 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1536 nm 1538 nm 1540 nm 1542 nm 1544 nm 1546 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1548 nm 1550 nm 1552 nm 1554 nm 1556 nm 1558 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1560 nm 1562 nm 1564 nm 1566 nm 1568 nm 1570 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1572 nm 1574 nm 1576 nm 1578 nm 1580 nm 1582 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1584 nm 1586 nm 1588 nm 1590 nm 1592 nm 1594 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1596 nm 1598 nm 1600 nm 1602 nm 1604 nm 1606 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1608 nm 1610 nm 1612 nm 1614 nm 1616 nm 1618 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1620 nm 1622 nm 1624 nm 1626 nm 1628 nm 1630 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1632 nm 1634 nm 1636 nm 1638 nm 1640 nm 1642 nm
## 0.0000000 -8.4203830 -3.4615592 0.0000000 0.0000000 0.0000000
## 1644 nm 1646 nm 1648 nm 1650 nm 1652 nm 1654 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1656 nm 1658 nm 1660 nm 1662 nm 1664 nm 1666 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1668 nm 1670 nm 1672 nm 1674 nm 1676 nm 1678 nm
## 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## 1680 nm 1682 nm 1684 nm 1686 nm 1688 nm 1690 nm
## 0.0000000 0.0000000 0.0000000 -0.6290696 0.0000000 -0.9336469
## 1692 nm 1694 nm 1696 nm 1698 nm 1700 nm
## -0.6872938 0.0000000 0.0000000 0.0000000 0.0000000
```

On peut afficher les longueurs d'ondes sélectionnées

```
plot(seq(900,1700,2),colMeans(gasoline$NIR),col=(abs(model_lasso$beta[tmp2,])>0)+1,xlab="",ylab="")
```



Comparaison des méthodes

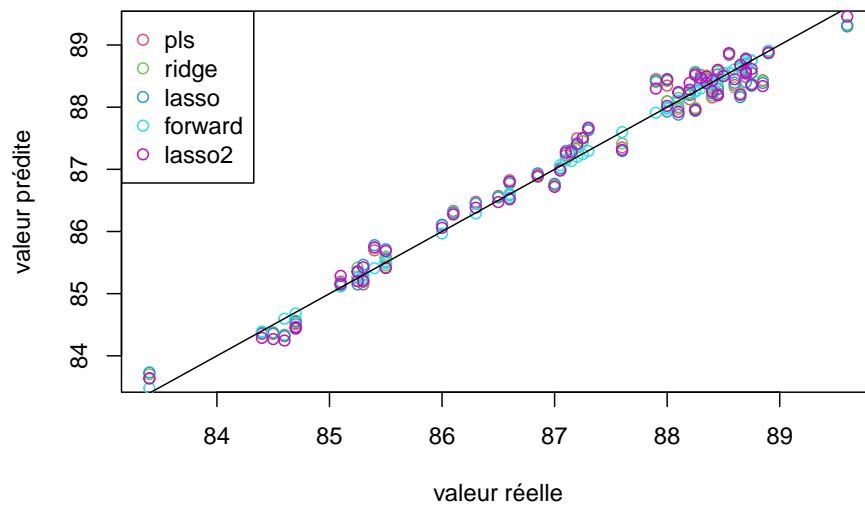
Comme les erreurs GCV et CV-MSE ne sont pas comparables directement, on va implémenter à la main une 10-fold CV pour comparer les 3 modèles. Pour la PLS, nous changeons de fonction pour la fonction *pls* du package *pls* qui comporte une fonction *predict* associé contrairement à la fonction que l'on a utilisé jusqu'à maintenant

```
library('pls')
x=cbind(gasoline$NIR,gasoline$octane)
fold=sample(1:10,nrow(gasoline),replace=T)
pred=matrix(0,nrow(gasoline),5)
for (i in 1:10){
  # pls
  model_pls=plsr(octane~NIR,data=data.frame(gasoline[-which(fold==i),]),scale=TRUE,ncomp=5)
  tmp=predict(model_pls,newdata=data.frame(gasoline[which(fold==i),]))
  pred[which(fold==i),1]=tmp[,5]
  # ridge
  model_ridge <- lm.ridge(octane~NIR,data=data.frame(gasoline[-which(fold==i),]),lambda=9)
  tmp= scale(gasoline$NIR[which(fold==i),], center = model_ridge$xm, scale = model_ridge$scales) %*% model_ridge$coef + model_ridge$ym
  pred[which(fold==i),2]=tmp
  # lasso
  model_lasso=lars(gasoline$NIR[-which(fold==i),],gasoline$octane[-which(fold==i)],type="lasso",trace=F,normalize=TRUE)
  tmp=predict(model_lasso,gasoline$NIR[which(fold==i),],s=lambda_opt,mode="lambda")
  pred[which(fold==i),3]=tmp$fit
  # lasso 2
  varindex=which((abs(model_lasso$beta[tmp2,])>0))
  data2=data.frame(octane=gasoline$octane[-which(fold==i)],NIRs=gasoline$NIR[-which(fold==i),varindex])
  newdata2=data.frame(octane=gasoline$octane[which(fold==i)],NIRs=gasoline$NIR[which(fold==i),varindex])
  model_lasso2=lm(octane~NIRs,data=data2)
  tmp=predict(model_lasso2,newdata2)
  pred[which(fold==i),5]=tmp
  # forward
  x=cbind(gasoline$NIR,gasoline$octane)
  data=data.frame(octane=x[,402],x[,1:401])
  model1b=lm(octane ~ X1208.nm + X1196.nm + X976.nm + X1692.nm +
    X970.nm + X1206.nm + X1056.nm + X1074.nm + X1098.nm + X1686.nm +
    X1700.nm + X1544.nm + X1546.nm + X1094.nm + X1552.nm + X1274.nm +
    X926.nm + X1342.nm + X940.nm + X918.nm + X1026.nm + X1064.nm +
    X1600.nm + X1280.nm + X1278.nm + X960.nm + X1104.nm + X1320.nm +
    X1588.nm + X900.nm + X1084.nm + X1080.nm + X1622.nm + X1076.nm +
    X1318.nm + X1590.nm + X1550.nm + X1680.nm + X1072.nm + X1628.nm +
    X1570.nm + X932.nm + X1562.nm + X1240.nm + X1328.nm + X1338.nm +
    X938.nm + X912.nm + X1304.nm + X914.nm,data=data[-which(fold==i),])
  tmp=predict(model1b,data[which(fold==i),])
  pred[which(fold==i),4]=tmp
}
```

Attention : les hyper-paramètres utilisés (ncomp pour PLS, lambda pour Ridge et LASSO, variables sélectionnées pour forward) ont été sélectionnés au préalable sur l'ensemble des données. Il faudrait idéalement le faire au sein de chaque itérations de la validation croisée.

On peut représenter graphiquement les prédictions en fonctions des valeurs réelles, toutes sont très bonnes

```
plot(gasoline$octane,pred[,1],col=2,xlab="valeur réelle",ylab="valeur prédite")
points(gasoline$octane,pred[,2],col=3)
points(gasoline$octane,pred[,3],col=4)
points(gasoline$octane,pred[,4],col=5)
points(gasoline$octane,pred[,5],col=6)
abline(coef = c(0,1))
legend("topleft",legend=c('pls','ridge','lasso','forward','lasso2'),col=2:6,pch=1)
```



On peut calculer l'erreur quadratique moyenne, mais les trois méthodes sont très proches

```
colnames(pred)=c('pls','ridge','lasso','forward','lasso2')
colMeans((pred-gasoline$octane)^2)
```

```
##      pls      ridge      lasso      forward      lasso2
## 0.04379013 0.04272772 0.04904911 74.93589869 0.04697540
```

La sélection forward marche étonnamment bien. C'est peut-être parce que la sélection de variables a été faite au préalable et non pas au sein de chaque itération de validation croisée, cette méthode tendant à faire du sur-apprentissage sur un si grand nombre de variables.