TOTAL

```
knitr::opts_chunk$set(echo = TRUE)
```

This notebook in an implementation of simple linear regression

```
#Loading data
trainG1<-read.csv('Train_G1.csv',sep = ';',header = TRUE)
trainG2<-read.csv('Train_G2.csv',sep = ';',header = TRUE)
testG1<-read.csv('Test_G1.csv',sep = ';',header = TRUE)
testG2<-read.csv('Test_G2.csv',sep = ';',header = TRUE)
trainG1[is.na(trainG1)] <- 0
trainG2[is.na(trainG2)] <- 0
```

Model for Oil

```
oilPredG1<-trainG1[,-which(names(trainG1) %in% c("GasCum360","API"))]
oilModelG1 <- lm(OilCum360 ~ ., oilPredG1)
#
oilPredG2<-trainG2[, -which(names(trainG2) %in% c("GasCum360","API"))]
oilModelG2 <- lm(OilCum360 ~ ., oilPredG2)
```

```
testOilG1 <- predict(oilModelG1, newdata = testG1, interval = 'pre')
testOilG2 <- predict(oilModelG2, newdata = testG2, interval = 'pre')
```

Model for Gas

```
gasPredG1<-trainG1[, -which(names(trainG1) %in% c("OilCum360","API"))]
gasModelG1 <- lm(GasCum360 ~ ., gasPredG1)
#
gasPredG2<-trainG2[, -which(names(trainG2) %in% c("OilCum360","API"))]
gasModelG2 <- lm(GasCum360 ~ ., gasPredG2)
```

```
testGasG1 <- predict(gasModelG1, newdata = testG1, interval = 'pre')
testGasG2 <- predict(gasModelG2, newdata = testG2, interval = 'pre')
```

```
test_m<-read.csv('test_merged.csv',sep = ',',header = TRUE)
```

```
test_m<-test_m[ , !(names(test_m) %in% c('Zone'))]
G1<-testG1[c('API','Zone')]
G2<-testG2[c('API','Zone')]
G<-rbind(G1,G2)
test_m2<-merge(test_m,G,by = 'API')
write.table(test_m2, "final_test.csv", sep = ",",
quote = FALSE, row.names = FALSE)
```

Résultats G1

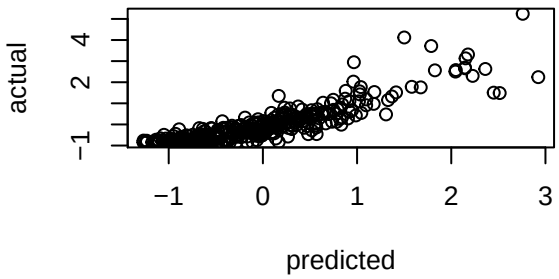
```
resultsG1 <- data.frame(ID = testG1$API,
CUM360_INF = testOilG1[,2],
CUM360_SUP = testOilG1[,3],
GAS360_INF = testGasG1[,2],
GAS360_SUP = testGasG1[,3])
```

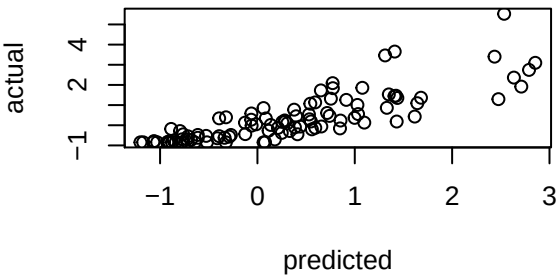
```
resultsG2 <- data.frame(ID = testG2$API,
CUM360_INF = testOilG2[,2],
CUM360_SUP = testOilG2[,3],
GAS360_INF = testGasG2[,2],
GAS360_SUP = testGasG2[,3])
```

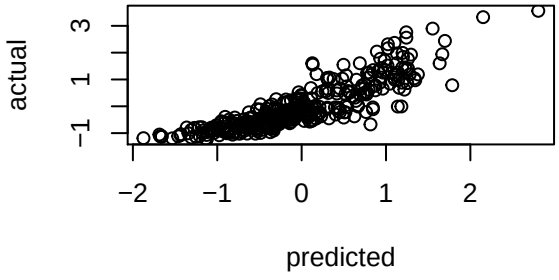
Writing result

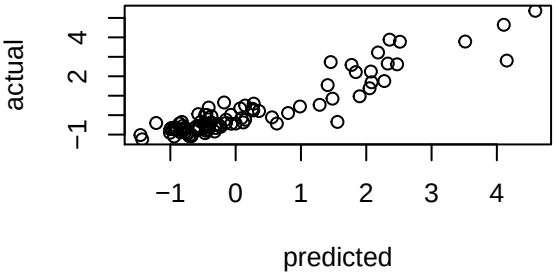
```
results<-rbind(resultsG1,resultsG2)
write.table(results, "submit.csv", sep = ";",
quote = FALSE, row.names = FALSE)
```

```
par(mfrow=c(2,2))
plot(predict(oilModelG1),trainG1$OilCum360,xlab="predicted",ylab="actual")
plot(predict(oilModelG2),trainG2$OilCum360,xlab="predicted",ylab="actual")
#
plot(predict(gasModelG1),trainG1$GasCum360,xlab="predicted",ylab="actual")
plot(predict(gasModelG2),trainG2$GasCum360,xlab="predicted",ylab="actual")
```










```
#RMSE for the simple model linéaire :
library(ModelMetrics)
mO1<-rmse(predicted = (oilModelG1$fitted.values), actual = trainG1$OilCum360)
print(mO1)
```

```
## [1] 0.4155829
```

```
mO2<-rmse(predicted = (oilModelG2$fitted.values), actual = trainG2$OilCum360)
print(mO2)
```

```
## [1] 0.6798284
```

```
mG1<-rmse(predicted = (gasModelG1$fitted.values), actual = trainG1$GasCum360)
print(mG1)
```

```
## [1] 0.4414425
```

```
mG2<-rmse(predicted = (gasModelG2$fitted.values), actual = trainG2$GasCum360)
print(mG2)
```

```
## [1] 0.519629
```

Implement PCR : PRINCIPAL COMPONENT REGRESSION (Feature selection for the linear regression) :

```
require(pls)
```

```
## Loading required package: pls
```

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

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(100)
oilModel1 <- pcr(OilCum360 ~ ., data=oilPredG1, scale = TRUE, validation = "CV")
oilModel2 <- pcr(OilCum360 ~ ., data=oilPredG2, scale = TRUE, validation = "CV")
gasModel1 <- pcr(GasCum360 ~ ., data=gasPredG1, scale = TRUE, validation = "CV")
gasModel2 <- pcr(GasCum360 ~ ., data=gasPredG2, scale = TRUE, validation = "CV")
```

```
summary(oilModel1)
```

```
## Data:    X dimension: 362 43
##  Y dimension: 362 1
## Fit method: svdpc
## Number of components considered: 43
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8999   0.8624   0.7321   0.6768   0.6601   0.6157   0.6073
## adjCV       0.8999   0.8625   0.7283   0.6726   0.6596   0.6143   0.6056
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.6067   0.5528   0.5468    0.5622    0.5652    0.5636    0.5601
## adjCV   0.6063   0.5492   0.5451    0.5603    0.5638    0.5570    0.5553
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.5452    0.5431    0.5644    0.5601    0.5484    0.5481
## adjCV    0.5418    0.5393    0.5590    0.5561    0.5443    0.5427
##        20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## CV       0.5220    0.5109    0.5005    0.5006    0.4987    0.5083
## adjCV    0.5201    0.5080    0.4990    0.4999    0.4971    0.5077
##        26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## CV       0.4918    0.4943    0.4983    0.4808    0.4835    0.4841
## adjCV    0.4899    0.4931    0.4954    0.4790    0.4814    0.4821
##        32 comps  33 comps  34 comps  35 comps  36 comps  37 comps
## CV       0.4899    0.4976    0.4917    0.9983     1.146     1.058
## adjCV    0.4874    0.4947    0.4881    0.9583     1.097     1.014
##        38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## CV       1.0134    1.0338    0.9827    0.8694    0.8888    0.7593
## adjCV    0.9723    0.9916    0.9435    0.8375    0.8556    0.7343
##
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
```

```
## X           18.919    32.80    44.08    50.56    56.46    61.11    65.36
## OilCum360    9.325    39.86    49.57    49.59    56.02    57.24    57.29
##            8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X            68.99    72.26     75.26     78.00     80.41     82.59
## OilCum360    64.44    64.83     64.84     65.06     68.62     68.62
##            14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X             84.65     86.39     87.98     89.43     90.77     92.02
## OilCum360     68.92     69.37     69.53     69.77     70.50     71.43
##            20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## X             93.14     94.13     94.96     95.68     96.34     96.90
## OilCum360     71.61     72.52     72.57     72.59     72.96     73.14
##            26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## X             97.45     97.92     98.32     98.70     98.95     99.17
## OilCum360     73.79     73.94     74.94     75.57     75.67     75.79
##            32 comps  33 comps  34 comps  35 comps  36 comps  37 comps
## X             99.35     99.51     99.65     99.77     99.84     99.89
## OilCum360     75.93     75.93     76.42     76.63     76.64     76.90
##            38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X             99.93     99.96     99.97     99.99    100.00    100.00
## OilCum360     77.02     77.02     77.18     77.30     77.32     78.56
```

```
summary(oilModel2)
```

```
## Data:    X dimension: 98 23
##  Y dimension: 98 1
## Fit method: svdpc
## Number of components considered: 23
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.211    1.214    1.214    1.012   0.9776   0.9852   0.9372
## adjCV        1.211    1.213    1.212    1.011   0.9762   0.9846   0.9201
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.9215   0.9725   0.9860    0.9855     1.148     1.167     1.071
## adjCV   0.9129   0.9679   0.9812    0.9821     1.134     1.153     1.049
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       1.0119    1.0162     1.032     1.059     1.055     1.026
## adjCV    0.9947    0.9981     1.013     1.038     1.034     1.006
##        20 comps  21 comps  22 comps  23 comps
## CV        1.037     1.069     1.079    0.9610
## adjCV     1.016     1.047     1.059    0.9431
##
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X          34.7104   60.130    75.62    82.83    86.54    89.21    91.70
## OilCum360   0.9648    3.211    31.20    35.21    35.54    44.50    47.14
##            8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X            93.97    95.40     96.58     97.58     98.43     99.08
## OilCum360    47.41    47.48     47.62     48.84     49.89     60.49
##            14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X             99.46     99.67     99.79     99.88     99.94     99.97
## OilCum360     61.29     61.97     62.05     62.07     62.86     63.69
##            20 comps  21 comps  22 comps  23 comps
## X             99.98     99.99       100    100.00
```

```
## OilCum360     63.94     64.00        64     67.85
```

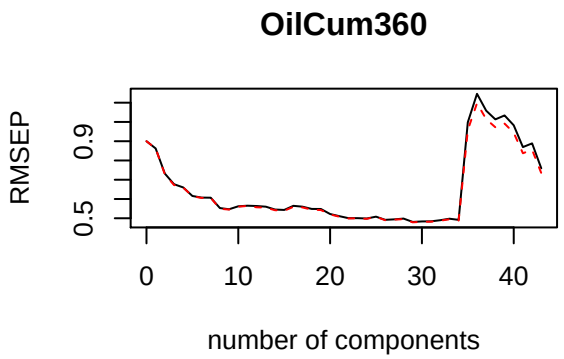
```
summary(gasModel1)
```

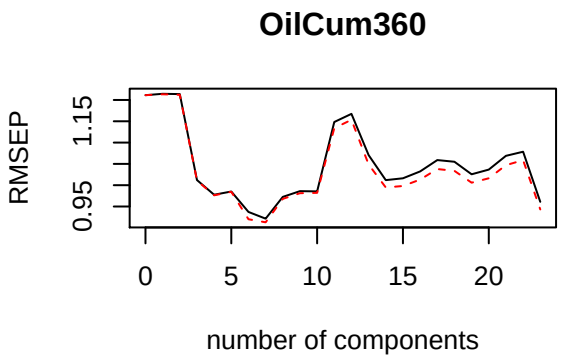
```
## Data:    X dimension: 362 43
##  Y dimension: 362 1
## Fit method: svdpc
## Number of components considered: 43
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8797   0.7797   0.6923   0.6791   0.6848   0.6770   0.6477
## adjCV       0.8797   0.7757   0.6907   0.6783   0.6850   0.6709   0.6456
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.6531   0.6527   0.5985    0.5766    0.5740    0.5731    0.5695
## adjCV   0.6516   0.6544   0.5919    0.5753    0.5718    0.5716    0.5686
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.5638    0.5424    0.5422    0.5458    0.5513    0.5407
## adjCV    0.5653    0.5394    0.5393    0.5442    0.5533    0.5381
##        20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## CV       0.5384    0.5279    0.5358    0.5404    0.5210    0.5170
## adjCV    0.5339    0.5251    0.5320    0.5360    0.5183    0.5132
##        26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## CV       0.5194    0.5296    0.5307    0.5315    0.5423    0.5346
## adjCV    0.5166    0.5263    0.5281    0.5270    0.5370    0.5297
##        32 comps  33 comps  34 comps  35 comps  36 comps  37 comps
## CV       0.5363    0.5563    0.5436    0.5462    0.5730    0.5348
## adjCV    0.5315    0.5500    0.5380    0.5406    0.5651    0.5327
##        38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## CV       0.5181    0.5188    0.5225    0.5175    0.5164    0.5238
## adjCV    0.5133    0.5144    0.5179    0.5133    0.5119    0.5187
##
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X            18.92    32.80    44.08    50.56    56.46    61.11    65.36
## GasCum360    27.00    39.73    41.88    41.89    48.92    49.29    49.45
##            8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X            68.99    72.26     75.26     78.00     80.41     82.59
## GasCum360    50.49    58.52     60.38     61.53     61.54     61.77
##            14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X             84.65     86.39     87.98     89.43     90.77     92.02
## GasCum360     62.01     65.35     65.76     65.79     65.93     68.70
##            20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## X             93.14     94.13     94.96     95.68     96.34     96.90
## GasCum360     69.52     69.76     70.15     70.39     70.68     70.98
##            26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## X             97.45     97.92     98.32     98.70     98.95     99.17
## GasCum360     70.98     70.98     71.14     71.88     71.88     72.06
##            32 comps  33 comps  34 comps  35 comps  36 comps  37 comps
## X             99.35     99.51     99.65     99.77     99.84     99.89
## GasCum360     72.06     72.09     72.30     72.40     72.42     72.70
##            38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X             99.93     99.96     99.97     99.99    100.00    100.00
## GasCum360     74.00     74.01     74.01     74.28     74.64     74.68
```

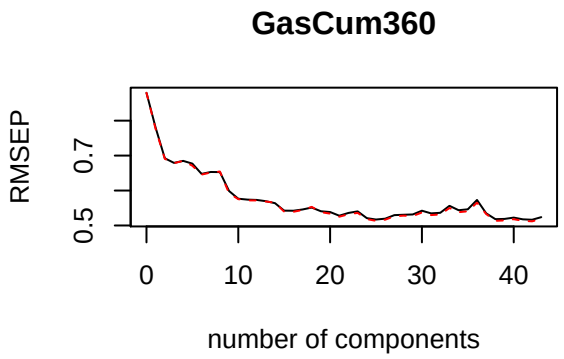
```r
summary(gasModel2)
```

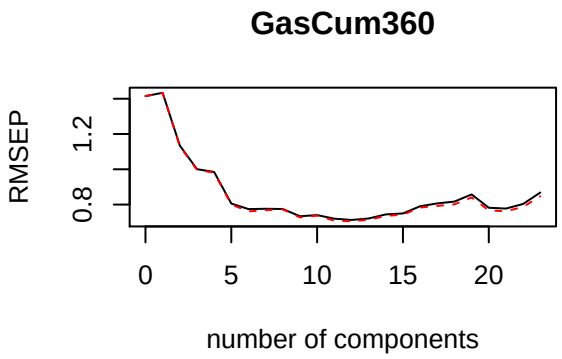
```
## Data:    X dimension: 98 23
##  Y dimension: 98 1
## Fit method: svdpc
## Number of components considered: 23
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.415    1.434    1.135   1.0006   0.9849   0.8059   0.7742
## adjCV        1.415    1.433    1.131   0.9969   0.9810   0.8003   0.7615
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.7760   0.7747   0.7338    0.7403    0.7201    0.7127    0.7211
## adjCV   0.7677   0.7729   0.7271    0.7373    0.7090    0.7051    0.7136
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.7441    0.7496    0.7905    0.8066    0.8170    0.8574
## adjCV    0.7357    0.7427    0.7832    0.7924    0.8013    0.8402
##        20 comps  21 comps  22 comps  23 comps
## CV       0.7820    0.7771    0.8035    0.8682
## adjCV    0.7668    0.7619    0.7867    0.8468
##
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           34.710    60.13    75.62    82.83    86.54    89.21    91.70
## GasCum360    1.614    39.98    55.46    58.00    73.51    75.83    76.33
##            8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X            93.97    95.40     96.58     97.58     98.43     99.08
## GasCum360    76.59    80.07     80.27     82.46     82.48     82.48
##            14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X             99.46     99.67     99.79     99.88     99.94     99.97
## GasCum360     82.48     82.61     83.00     84.44     84.63     84.70
##            20 comps  21 comps  22 comps  23 comps
## X             99.98     99.99    100.00    100.00
## GasCum360     85.83     86.22     86.23     86.23
```

```r
#Plot the ROOT MEAN SQUARE ERROR
par(mfrow=c(2,2))
validationplot(oilModel1)
validationplot(oilModel2)
validationplot(gasModel1)
validationplot(gasModel2)
```

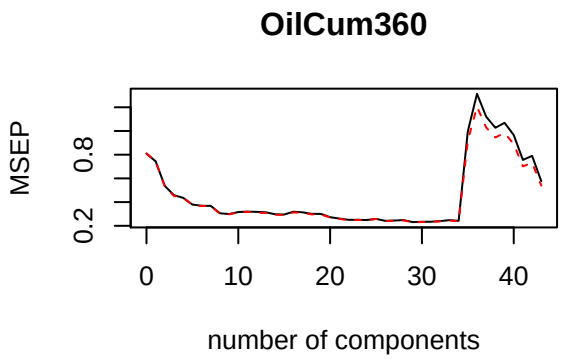


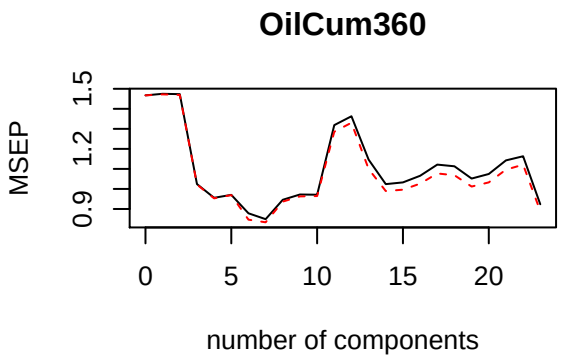


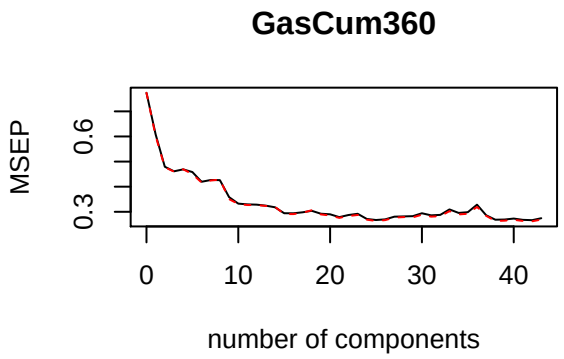


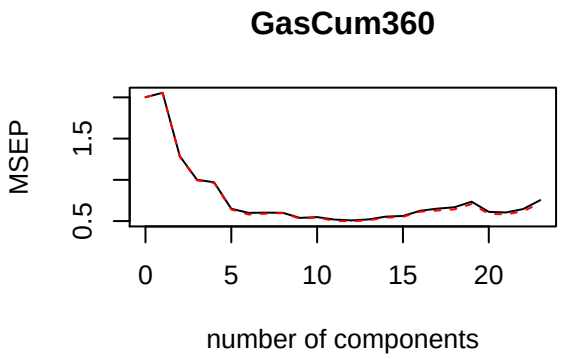



```
#Plot the ROOT MEAN SQUARE ERROR
par(mfrow=c(2,2))
validationplot(oilModel1,val.type = "MSEP")
validationplot(oilModel2,val.type = "MSEP")
validationplot(gasModel1,val.type = "MSEP")
validationplot(gasModel2,val.type = "MSEP")
```

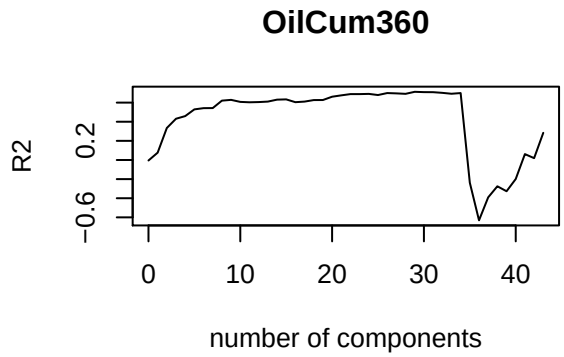


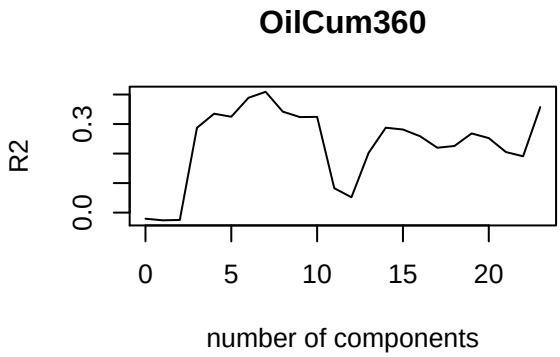


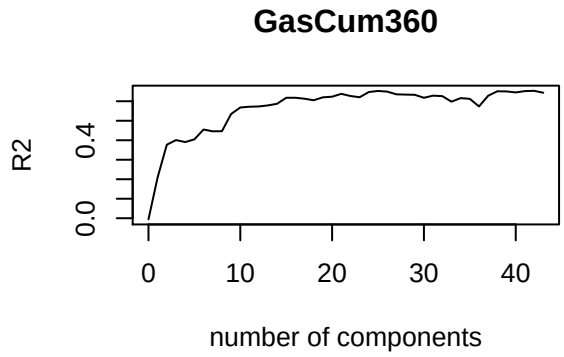


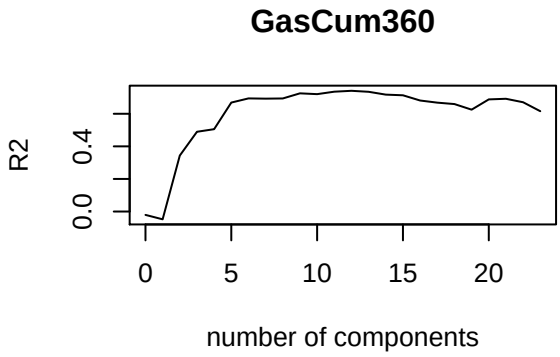



```
#Plot the ROOT MEAN SQUARE ERROR
par(mfrow=c(2,2))
validationplot(oilModel1,val.type = "R2")
validationplot(oilModel2,val.type = "R2")
validationplot(gasModel1,val.type = "R2")
validationplot(gasModel2,val.type = "R2")
```









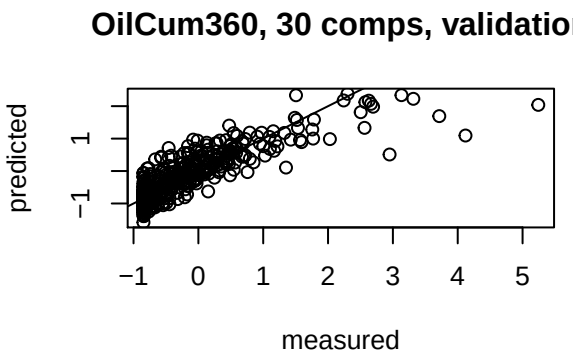
What we want is a low cross validation error with a lower number of components than the number of variables in your dataset

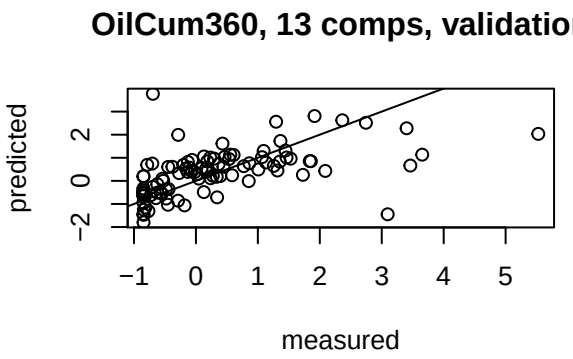
- for oilModel1 : 9cps (0.55,72%) - 30cps (0.49,99%)
- for oilModel2 : 13cps (0.87,99%)
- for gasModel1 : 21cps (0.52,94%) - 29cps (0.524,98%)
- for gasModel2 : 13cps (0.685,99%)

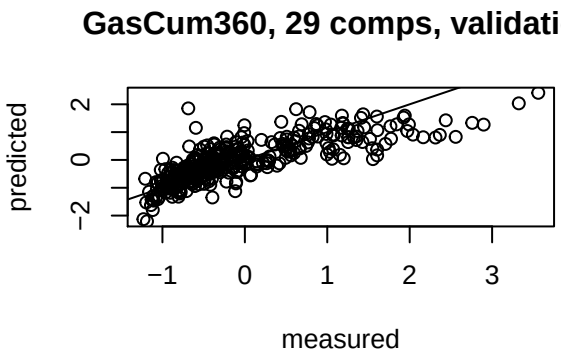
```
set.seed(100)
oilModel1 <- pcr(OilCum360 ~ ., data=oilPredG1, scale = TRUE, validation = "CV")
oilModel2 <- pcr(OilCum360 ~ ., data=oilPredG2, scale = TRUE, validation = "CV")
gasModel1 <- pcr(GasCum360 ~ ., data=gasPredG1, scale = TRUE, validation = "CV")
gasModel2 <- pcr(GasCum360 ~ ., data=gasPredG2, scale = TRUE, validation = "CV")
```

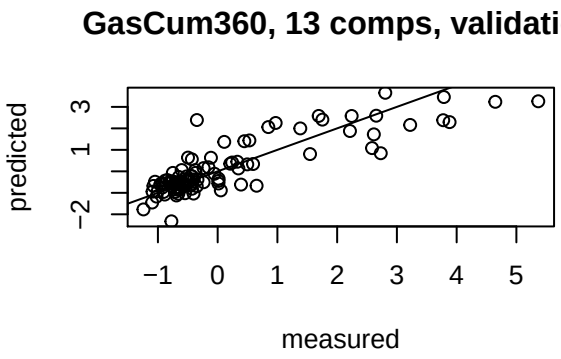
```
testOil1 <- predict(oilModel1, newdata = testG1,ncomp = 30 , interval = 'prediction')
testOil2 <- predict(oilModel2, newdata = testG2,ncomp = 13 , interval = 'prediction')
testGas1 <- predict(gasModel1, newdata = testG1,ncomp = 29, interval = 'prediction')
testGas2 <- predict(gasModel2, newdata = testG2,ncomp = 13, interval = 'prediction')
```

```
par(mfrow=c(2,2))
plot(oilModel1, ncomp=30, line=TRUE)
plot(oilModel2, ncomp=13, line=TRUE)
plot(gasModel1, ncomp=29, line=TRUE)
plot(gasModel2, ncomp=13, line=TRUE)
```










```
#RMSE
library(ModelMetrics)
mO_1<-rmse(predicted = (oilModel1$fitted.values[,,30]), actual = trainG1$OilCum360)
print(mO_1)
```

```
## [1] 0.4426886
```

```
mO_2<-rmse(predicted = (oilModel2$fitted.values[,,13]), actual = trainG2$OilCum360)
print(mO_2)
```

```
## [1] 0.7535584
```

```
mG_1<-rmse(predicted = (gasModel1$fitted.values[,,29]), actual = trainG1$GasCum360)
print(mG_1)
```

```
## [1] 0.4652408
```

```
mG_2<-rmse(predicted = (gasModel2$fitted.values[,,13]), actual = trainG2$GasCum360)
print(mG_2)
```

```
## [1] 0.5862397
```

[1] 0.4155829 [1] 0.6798284 [1] 0.4414425 [1] 0.519629 ## Linear model By zone :

```
partitions<-read.csv("partitionned_data2.csv",sep="," ,header = TRUE)[,c(3,49)]
```

```
trainG1<-(merge(x=trainG1, y=partitions, by.x = "API", by.y = "API"))
trainG2<-(merge(x=trainG2, y=partitions, by.x = "API", by.y = "API"))
```

```
###SPLITTING TO GROUPS
#group 0:(low gas): Zones 1,2,4, part of zone 0
#group 1:(intermediate) part of zone 0
#group 2:(low oil): Zones 5,6,7,8,9 part of zone 0
trainG1_1<-trainG1[trainG1$group==1,-which(names(trainG1) %in% c("group"))]
trainG1_2<-trainG1[trainG1$group==2,-which(names(trainG1) %in% c("group"))]
trainG1_3<-trainG1[trainG1$group==3,-which(names(trainG1) %in% c("group"))]
#
trainG2_1<-trainG2[trainG2$group==1,-which(names(trainG2) %in% c("group"))]
trainG2_2<-trainG2[trainG2$group==2,-which(names(trainG2) %in% c("group"))]
trainG2_3<-trainG2[trainG2$group==3,-which(names(trainG2) %in% c("group"))]
```

```
nrow(trainG1_1)
```

```
## [1] 67
```

```
nrow(trainG1_2)
```

```
## [1] 135
```

```
nrow(trainG1_3)
```

```
## [1] 160
```

```
nrow(trainG2_1)
```

```
## [1] 18
```

```
nrow(trainG2_2)
```

```
## [1] 50
```

```
nrow(trainG2_3)
```

```
## [1] 30
```

Model for Oil

```
#one model by partition (G1)
oilPredG1_1<-trainG1_1[,-which(names(trainG1_1) %in% c("GasCum360","API"))]
oilModelG1_1 <- lm(OilCum360 ~ ., oilPredG1_1)
#
oilPredG1_2<-trainG1_2[,-which(names(trainG1_2) %in% c("GasCum360","API"))]
oilModelG1_2 <- lm(OilCum360 ~ ., oilPredG1_2)
#
oilPredG1_3<-trainG1_3[,-which(names(trainG1_3) %in% c("GasCum360","API"))]
oilModelG1_3 <- lm(OilCum360 ~ ., oilPredG1_3)

#one model by partition (G2)
oilPredG2_1<-trainG2_1[, -which(names(trainG2_1) %in% c("GasCum360","API"))]
oilModelG2_1 <- lm(OilCum360 ~ ., oilPredG2_1)
#
oilPredG2_2<-trainG2_2[, -which(names(trainG2_2) %in% c("GasCum360","API"))]
oilModelG2_2 <- lm(OilCum360 ~ ., oilPredG2_2)
#
```

```
oilPredG2_3<-trainG2_3[, -which(names(trainG2_3) %in% c("GasCum360","API"))]
oilModelG2_3 <- lm(OilCum360 ~ ., oilPredG2_3)
```

Model for Gas

```
#one model by partition (G1)
gasPredG1_1<-trainG1_1[,-which(names(trainG1_1) %in% c("OilCum360","API"))]
gasModelG1_1 <- lm(GasCum360 ~ ., gasPredG1_1)
#
gasPredG1_2<-trainG1_2[,-which(names(trainG1_2) %in% c("OilCum360","API"))]
gasModelG1_2 <- lm(GasCum360 ~ ., gasPredG1_2)
#
gasPredG1_3<-trainG1_3[,-which(names(trainG1_3) %in% c("OilCum360","API"))]
gasModelG1_3 <- lm(GasCum360 ~ ., gasPredG1_3)

#one model by partition (G2)
gasPredG2_1<-trainG2_1[, -which(names(trainG2_1) %in% c("OilCum360","API"))]
gasModelG2_1 <- lm(GasCum360 ~ ., gasPredG2_1)
#
gasPredG2_2<-trainG2_2[, -which(names(trainG2_2) %in% c("OilCum360","API"))]
gasModelG2_2 <- lm(GasCum360 ~ ., gasPredG2_2)
#
gasPredG2_3<-trainG2_3[, -which(names(trainG2_3) %in% c("OilCum360","API"))]
gasModelG2_3 <- lm(GasCum360 ~ ., gasPredG2_3)
```

Regression for group1_OIL

```
par(mfrow=c(2,2))
plot(predict(oilModelG1_1),trainG1_1$OilCum360,xlab="predicted",ylab="actual",line=TRUE)
```

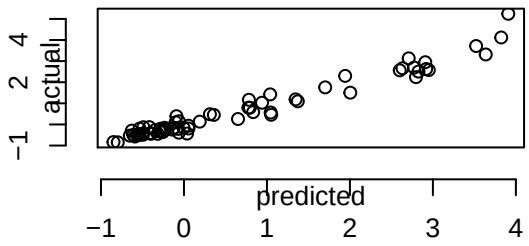
```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
## Warning in box(...): "line" n'est pas un paramètre graphique
```

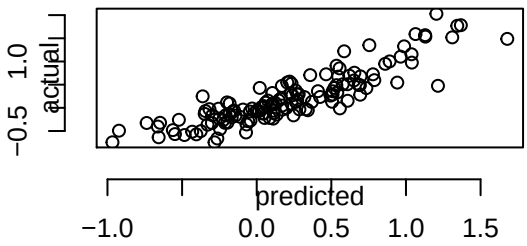
```
#
plot(predict(oilModelG1_2),trainG1_2$OilCum360,xlab="predicted",ylab="actual",line=TRUE)
```

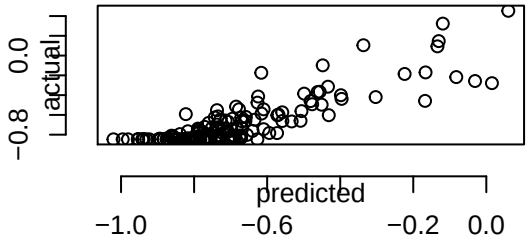
```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
## Warning in box(...): "line" n'est pas un paramètre graphique
```

```
#
plot(predict(oilModelG1_3),trainG1_3$OilCum360,xlab="predicted",ylab="actual",line=TRUE)
```

```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
## Warning in box(...): "line" n'est pas un paramètre graphique
```








```
#Errors
mO1_1<-rmse(predicted = predict(oilModelG1_1), actual = trainG1_1$OilCum360)
print(mO1_1)
```

```
## [1] 0.2940253
```

```
mO1_2<-rmse(predicted = predict(oilModelG1_2), actual = trainG1_2$OilCum360)
print(mO1_2)
```

```
## [1] 0.2651414
```

```
mO1_3<-rmse(predicted = predict(oilModelG1_3), actual = trainG1_3$OilCum360)
print(mO1_3)
```

```
## [1] 0.1229135
```

Regression for group2_OIL

```
par(mfrow=c(2,2))
plot(predict(oilModelG2_1),trainG2_1$OilCum360,xlab="predicted",ylab="actual",line=TRUE)
```

```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
```

```
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
```

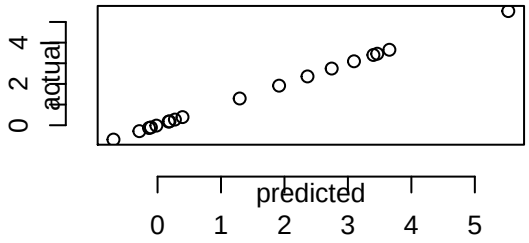
```
## Warning in box(...): "line" n'est pas un paramètre graphique
```

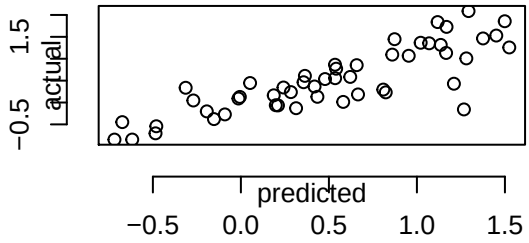
```
#
plot(predict(oilModelG2_2),trainG2_2$OilCum360,xlab="predicted",ylab="actual",line=TRUE)
```

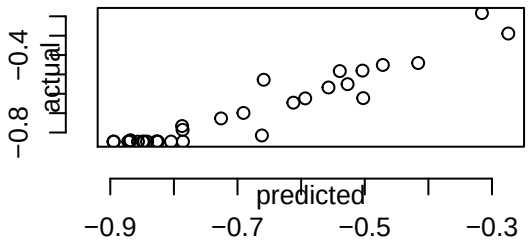
```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
## Warning in box(...): "line" n'est pas un paramètre graphique
```

```
#
plot(predict(oilModelG2_3),trainG2_3$OilCum360,xlab="predicted",ylab="actual",line=TRUE)
```

```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
## Warning in box(...): "line" n'est pas un paramètre graphique
```








```
#Errors
mO2_1<-rmse(predicted = predict(oilModelG2_1), actual = trainG2_1$OilCum360)
print(mO2_1)
```

```
## [1] 9.486223e-13
```

```
mO2_2<-rmse(predicted = predict(oilModelG2_2), actual = trainG2_2$OilCum360)
print(mO2_2)
```

```
## [1] 0.3968312
```

```
mO2_3<-rmse(predicted = predict(oilModelG2_3), actual = trainG2_3$OilCum360)
print(mO2_3)
```

```
## [1] 0.05596098
```

Regression for group1_GAS

```
par(mfrow=c(2,2))
plot(predict(gasModelG1_1),trainG1_1$GasCum360,xlab="predicted",ylab="actual",line=TRUE)
```

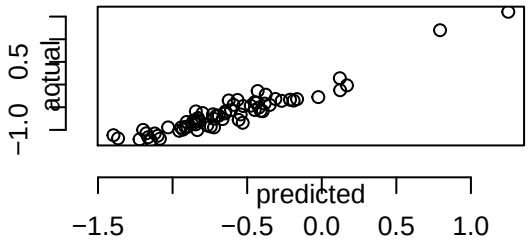
```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
## Warning in box(...): "line" n'est pas un paramètre graphique
```

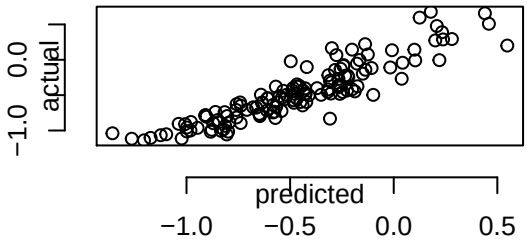
```
#
plot(predict(gasModelG1_2),trainG1_2$GasCum360,xlab="predicted",ylab="actual",line=TRUE)
```

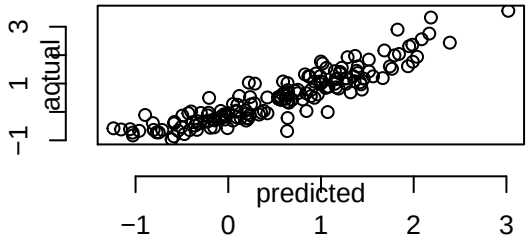
```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
## Warning in box(...): "line" n'est pas un paramètre graphique
```

```
#
plot(predict(gasModelG1_3),trainG1_3$GasCum360,xlab="predicted",ylab="actual",line=TRUE)
```

```
## Warning in plot.window(...): "line" n'est pas un paramètre graphique
## Warning in plot.xy(xy, type, ...): "line" n'est pas un paramètre graphique
## Warning in box(...): "line" n'est pas un paramètre graphique
```








```
#Errors
mG1_1<-rmse(predicted = predict(gasModelG1_1), actual = trainG1_1$GasCum360)
print(mG1_1)
```

```
## [1] 0.150233
```

```
mG1_2<-rmse(predicted = predict(gasModelG1_2), actual = trainG1_2$GasCum360)
print(mG1_2)
```

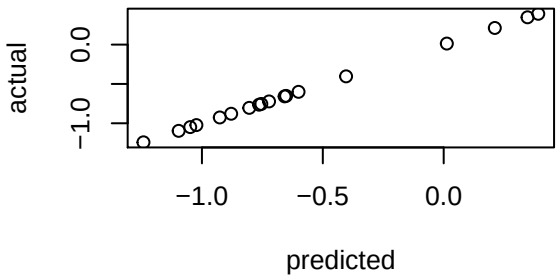
```
## [1] 0.1699014
```

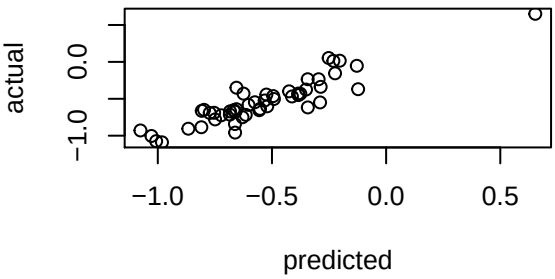
```
mG1_3<-rmse(predicted = predict(gasModelG1_3), actual = trainG1_3$GasCum360)
print(mG1_3)
```

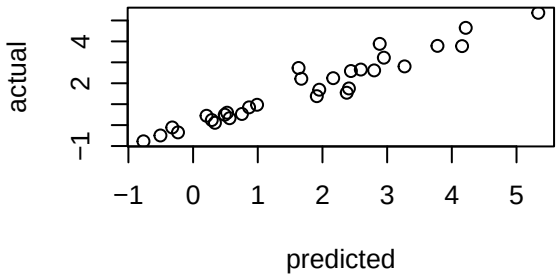
```
## [1] 0.3782107
```

Regression for group2_GAS

```
par(mfrow=c(2,2))
plot(predict(gasModelG2_1),trainG2_1$GasCum360,xlab="predicted",ylab="actual")
#
plot(predict(gasModelG2_2),trainG2_2$GasCum360,xlab="predicted",ylab="actual")
#
plot(predict(gasModelG2_3),trainG2_3$GasCum360,xlab="predicted",ylab="actual")
```








```
#eRRORS
mG2_1<-rmse(predicted = predict(gasModelG2_1), actual = trainG2_1$GasCum360)
print(mG2_1)
```

```
## [1] 2.950129e-13
```

```
mG2_2<-rmse(predicted = predict(gasModelG2_2), actual = trainG2_2$GasCum360)
print(mG2_2)
```

```
## [1] 0.1309738
```

```
mG2_3<-rmse(predicted = predict(gasModelG2_3), actual = trainG2_3$GasCum360)
print(mG2_3)
```

```
## [1] 0.4051946
```

```
#Table Resuming Results
dd<-data.frame(taille=c(nrow(trainG1_1),nrow(trainG1_2),nrow(trainG1_3),nrow(trainG1_1),nrow(trainG1_2)
dd
```

```
##                taille         rmse
## G1_oil_group1      67 2.940253e-01
## G1_oil_group2     135 2.651414e-01
## G1_oil_group3     160 1.229135e-01
## G1_gas_group1      67 1.502330e-01
## G1_gas_group2     135 1.699014e-01
## G1_gas_group3     160 3.782107e-01
## G2_oil_group1      18 9.486223e-13
## G2_oil_group2      50 3.968312e-01
```

```
## G2_oil_group3     30 5.596098e-02
## G2_gas_group1     18 2.950129e-13
## G2_gas_group2     50 1.309738e-01
## G2_gas_group3     30 4.051946e-01
```

Errors(means for groups):

[1] 0.2273601 [1] 0.1509307 [1] 0.2327817 [1] 0.1787228

Errors in the simple general model

[1] 0.4155829 [1] 0.6798284 [1] 0.4414425 [1] 0.519629