TP Krigeage - Challenge

```
#1. SROUR Mathieu
#2. ALLOUCH Omar
#3. LAMNAOIR Imane
```

Dans ce compte rendu, nous présentons les différentes méthodes testées. On teste chaque méthode par la crosse validation afin d'évaluer la performance de la manière la plus pertinante possible puisqu'on a remarqué que le changement de la décomposition train/test peut changer largement les résultats trouvés. La méthode retenue est celle de la partie 8.

Les prédiction

1. Dévision en ensemble d'apprentissage et ensemble test

On commence d'abord par déviser notre base de données à une base d'apprentissage et une base test afin de voir le meilleur modèle en calculant RMSE sur les données test.

Notre base de données contient 700 observations avec 7 variables. La base d'apprentissage contiendra 550 observations et la base de test contiendra 150. Cela nous permet d'avoir les paramètres optimaux de chaque modèle tout en évitant le surpprentissage puisque le RMSE sera calculé sur l'ensemble de test.

```
set.seed(12345)
Observations = read.csv("defi_observations.csv", header = TRUE)
p<-7
n<-700
X=Observations[,1:p]
Y=Observations[,(p+1)]
n_app<-550
n_test<-150
y_app<-Observations[1:n_app,8]
X_app<-Observations[1:n_app,1:p]
y_test<-Observations[(n_app+1):n,8]
X_test<-Observations[(n_app+1):n,1:p]
mu.y<-mean(y_app)
# prédiction de Y par sa moyenne sur les données d'apprentissage
RMSE.ref<-sqrt(sum((y_test-mu.y)^2)/n_test)
print(RMSE.ref)
```

```
## [1] 6.556478
```

On remarque que La valeur RMSE en utilisant la moyenne est très élevée.

1.a Dévision en ensemble d'apprentissage et ensemble test et moyenne avec Cross Validation

```
#mélanger les données aléatoirement
Observations<-Observations[sample(nrow(Observations)),]
#Creation de 10 plis de taille egale
folds <- cut(seq(1,nrow(Observations)),breaks=5,labels=FALSE)
lR<-seq(1,5)
# 10 plis validation croisée
for(i in 1:5){
  testIndexes <- which(folds==i,arr.ind=TRUE)
  y_app_cv<-Observations[-testIndexes,8]
  X_app_cv<-Observations[-testIndexes,1:p]
  y_test_cv<-Observations[testIndexes,8]
  X_test_cv<-Observations[testIndexes,1:p]
  mucv.y<-mean(y_app_cv)
  # prédiction de Y par sa moyenne sur les données d'apprentissage
  RMSE.ref<-sqrt(sum((y_test_cv-mucv.y)^2)/n_test)
  lR[i] <- RMSE.ref
  #print(RMSE.ref)
}
print(mean(lR))
```

```
## [1] 6.085994
```

2. La fonction pour calculer RMSE

On implémente une fonction qui calcule la valeur de RMSE et qui en même temps trace le graphe des résidus et le graphe des observations en fonction des prédictions.

```
rmse<-function(y, ypred, trace=FALSE){

  rmse=y-ypred

  if( trace){
  plot( ypred, y, xlab="Ypred", ylab="Ytrue")
  abline(a=0,b=1, col="blue")
  title(" Valeurs observées en fonctions des prédictions")

  plot(rmse, main="Résidus")
  }

  return( sqrt( mean( rmse**2)  ) )
}
```

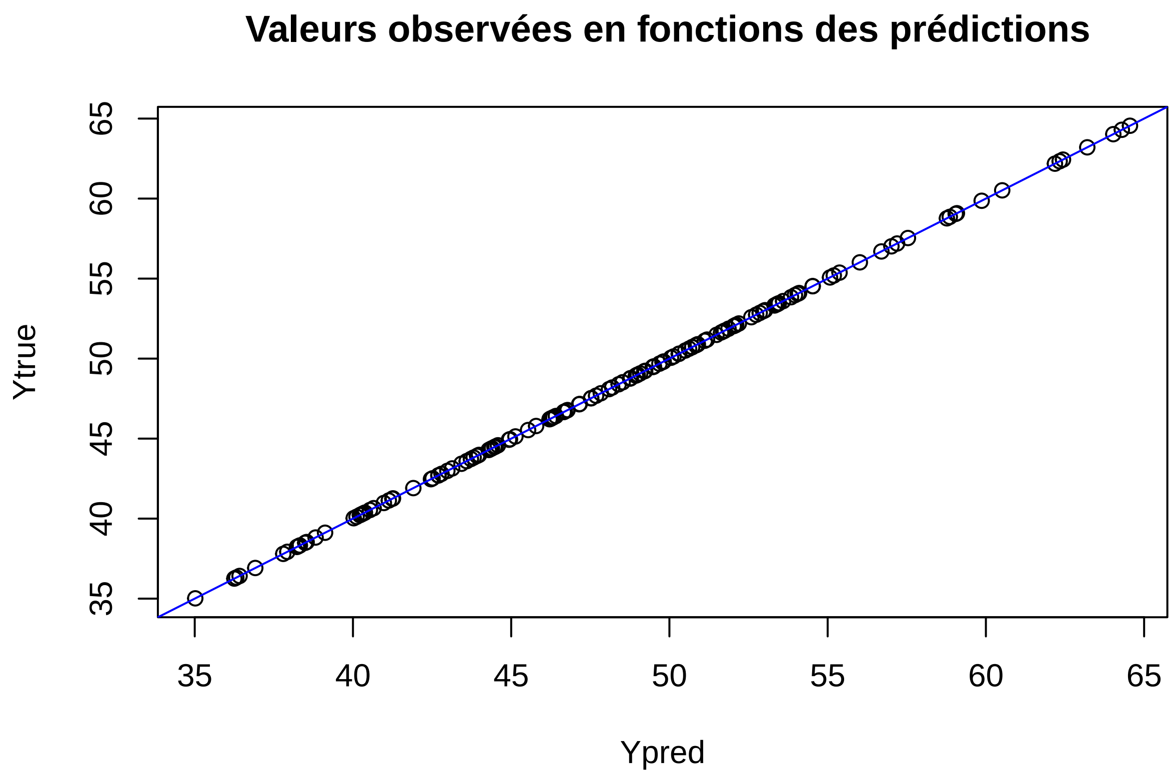
3. Krigeage Simple

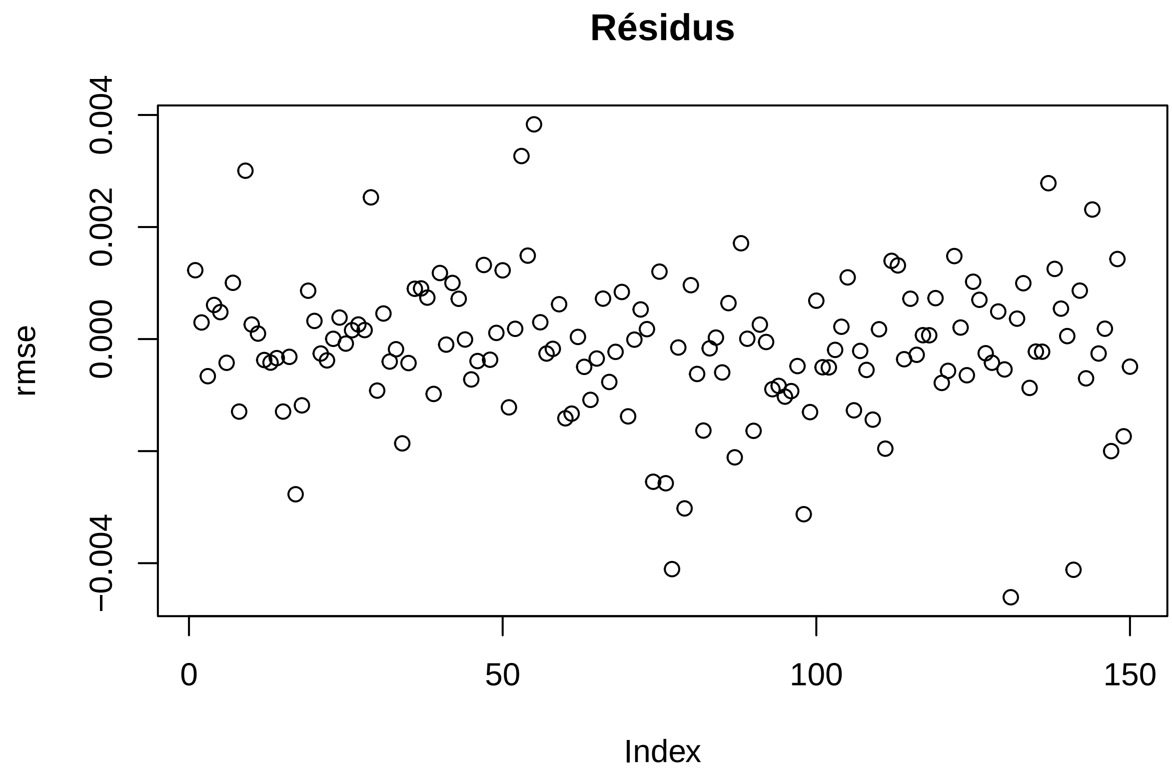
Nous commençant par la méthode de krigeage simple, et on varie la valeur de theta afin d'avoir la valeur RMSE la plus petite.

```r
library(DiceKriging)
```

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

```r
theta=rep(9.4,7)
Msimple<-km(formula=~1,design=X_app,response=y_app,coef.trend=0,coef.cov=theta,coef.var = 1)
predsimple<-predict(object =Msimple,newdata=X_test,type="SK",ckeckNames=False )
Meansimple<-predsimple$mean
rmse(y_test, Meansimple, trace=TRUE)
```






```
## [1] 0.001278394
```

3.a Krigeage Simple avec CrossValidation

```
#Randomly shuffle the data
Observations<-Observations[sample(nrow(Observations)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(Observations)),breaks=5,labels=FALSE)
lR<-seq(1,5)
#Perform 10 fold cross validation
for(i in 1:5){
  #Segement your data by fold using the which() function
  testIndexes <- which(folds==i,arr.ind=TRUE)
  y_app_cv<-Observations[-testIndexes,8]
  X_app_cv<-Observations[-testIndexes,1:p]
  y_test_cv<-Observations[testIndexes,8]
  X_test_cv<-Observations[testIndexes,1:p]

  Msimplecv<-km(formula=~1,design=X_app_cv,response=y_app_cv,coef.trend=0,coef.cov=theta,coef.var = 1)
  predsimplecv<-predict(object =Msimplecv,newdata=X_test_cv,type="SK",ckeckNames=False )
  Meansimplecv<-predsimplecv$mean
  lR[i] <- rmse(y_test_cv, Meansimplecv)
  #print(rmse(y_test_cv, Meansimplecv))
```

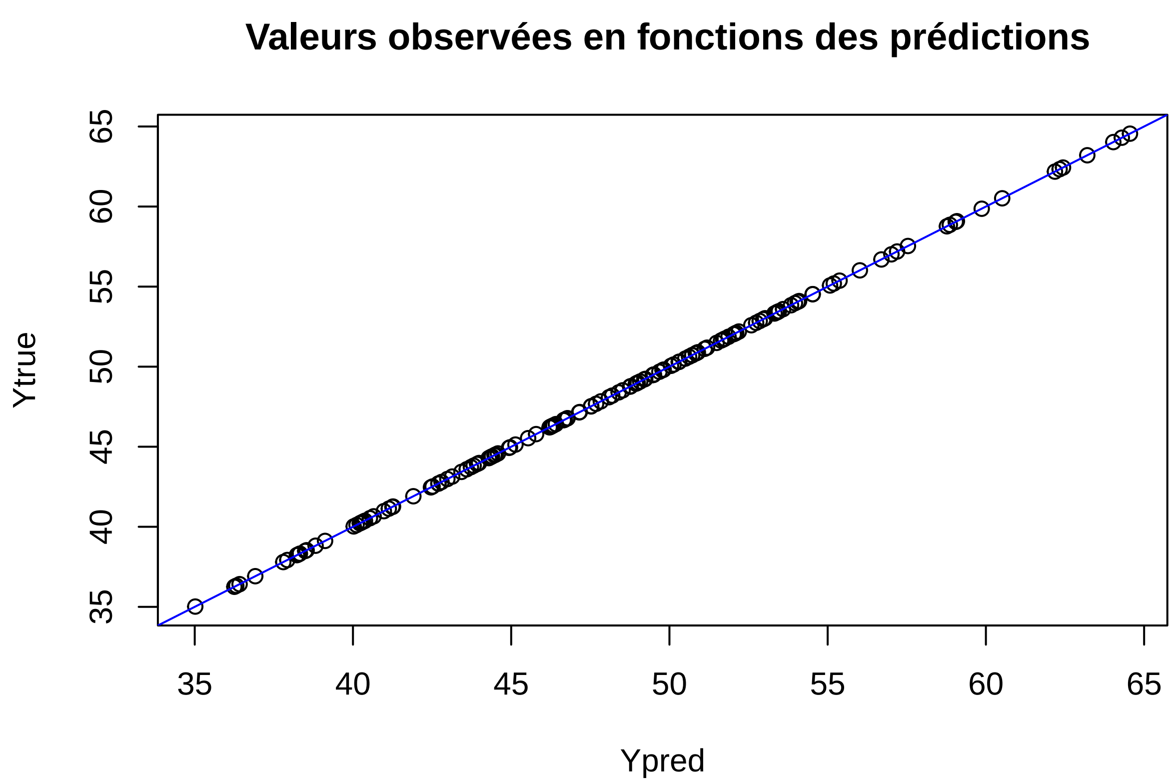
```
}
print(mean(lR))
```

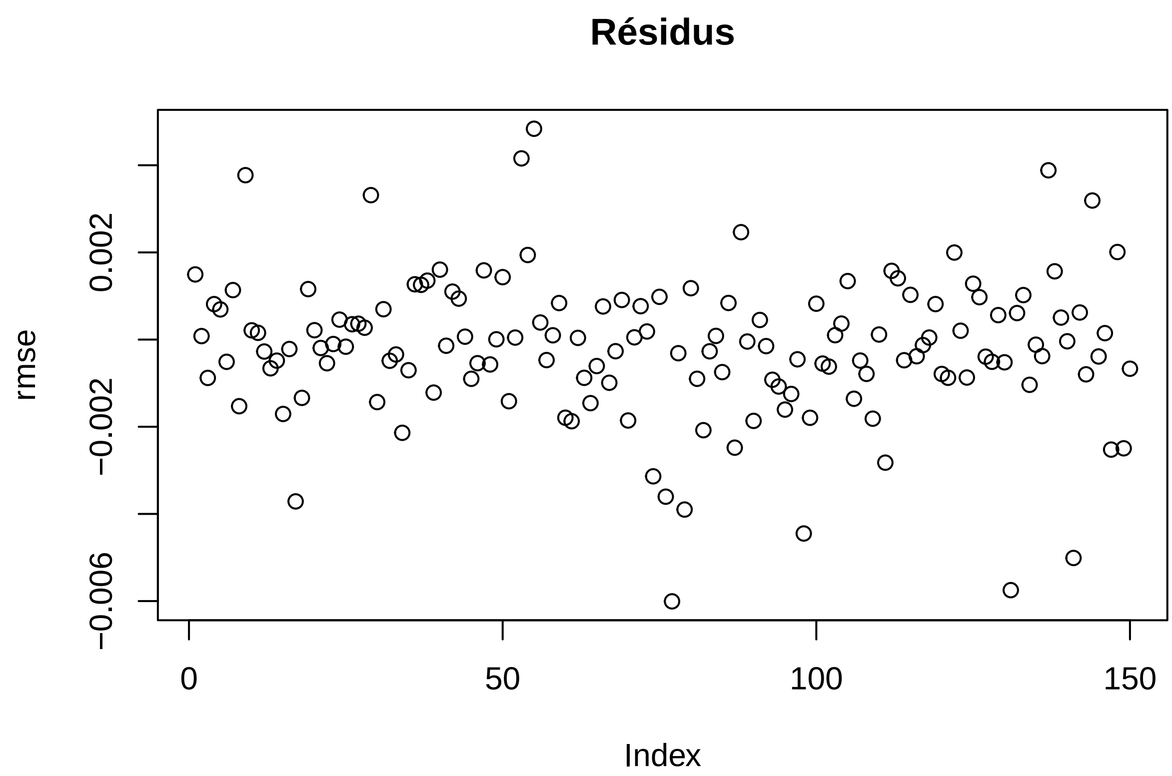
```
## [1] 0.001203203
```

4. Krigeage Ordinaire

On teste après la méthode de krigeage ordinaire

```
theta=rep(8,7)
Mordinaire<-km(formula=~1,design=X_app,response=y_app,coef.trend=NULL,coef.cov=theta,coef.var = 1)
predordinaire<-predict(object =Mordinaire,newdata=X_test,type="UK",ckeckNames=False )
Meanordinaire<-predordinaire$mean
rmse(y_test, Meanordinaire, trace=TRUE)
```






```
## [1] 0.001659609
```

4. Krigeage Ordinaire avec CrossValidation

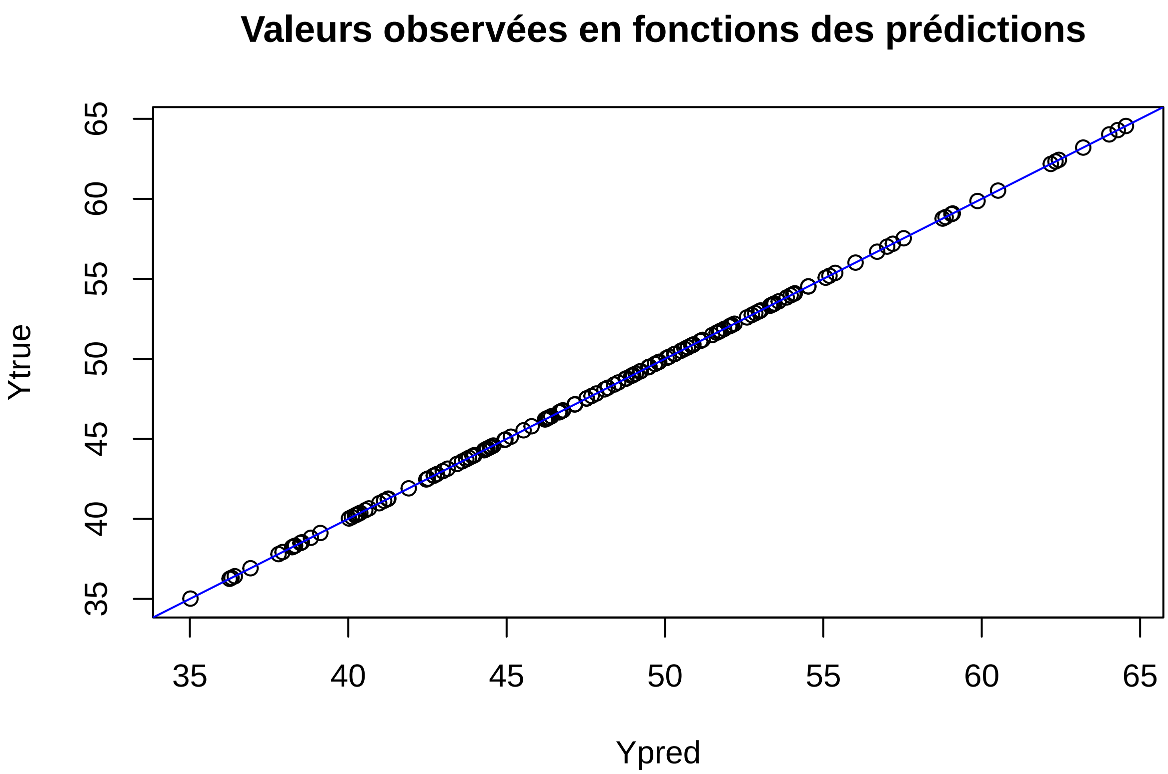
```
Observations<-Observations[sample(nrow(Observations)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(Observations)),breaks=5,labels=FALSE)
lR<-seq(1,5)
#Perform 10 fold cross validation
for(i in 1:5){
  #Segement your data by fold using the which() function
  testIndexes <- which(folds==i,arr.ind=TRUE)
  y_app_cv<-Observations[-testIndexes,8]
  X_app_cv<-Observations[-testIndexes,1:p]
  y_test_cv<-Observations[testIndexes,8]
  X_test_cv<-Observations[testIndexes,1:p]

  Mordinaire<-km(formula=~1,design=X_app_cv,response=y_app_cv,coef.trend=NULL,coef.cov=theta,coef.var =
  predordinaire<-predict(object =Mordinaire,newdata=X_test_cv,type="UK",ckeckNames=False )
  Meanordinaire<-predordinaire$mean
  lR[i] <- rmse(y_test_cv, Meanordinaire)
}
print(mean(lR))
```

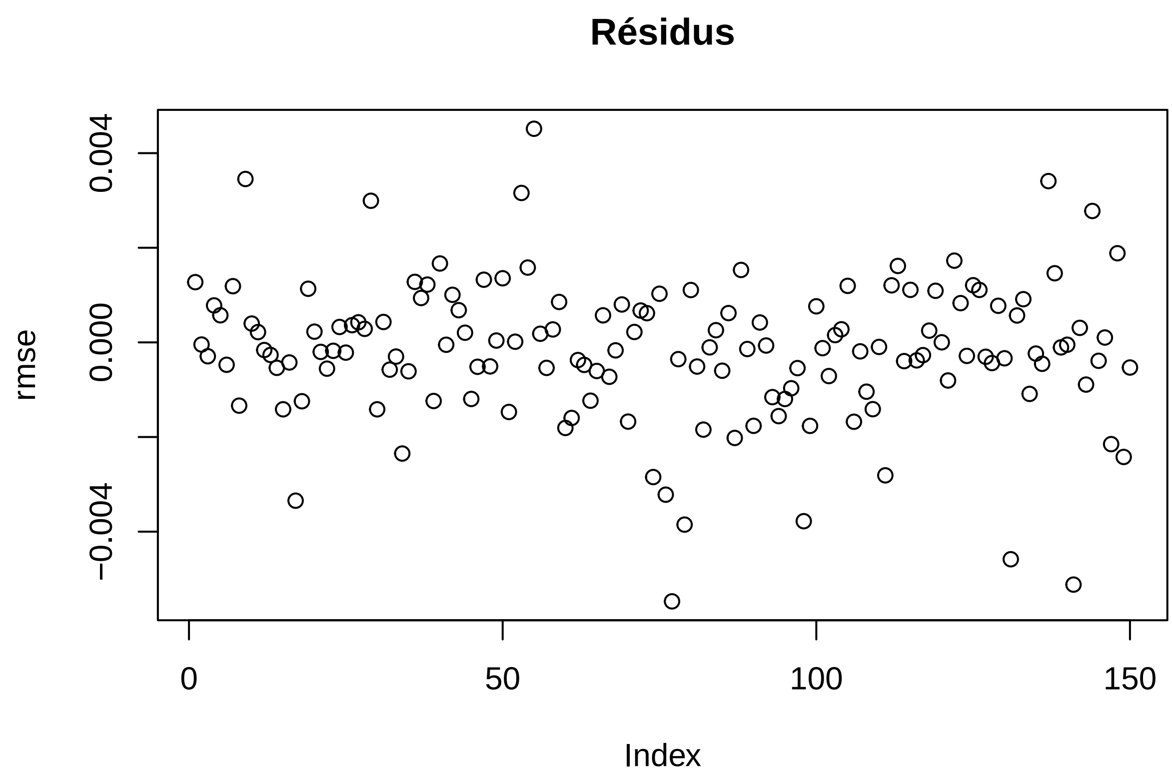
```
## [1] 0.001389223
```

5. Krigeage universal

Pour la mèthode de krigeage universel, on teste les deux formule "y~1+X" et "y~1+X+X^2" qui nous donne presque le même résultat

```
theta=rep(8.4,7)
Muniversel <- km(formula = ~y_app~1+.+.^2, coef.trend = NULL, design = X_app, response = y_app,coef.cov
preduniversel <- predict(object = Muniversel, newdata = X_test, type = "UK")
meanuniversel <- preduniversel$mean
rmse(y_test, meanuniversel, trace=TRUE)
```






```
## [1] 0.001505418
```

5.a Krigeage universal avec CrossValidation

```
Observations<-Observations[sample(nrow(Observations)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(Observations)),breaks=5,labels=FALSE)
lR<-seq(1,5)
#Perform 10 fold cross validation
for(i in 1:5){
  #Segement your data by fold using the which() function
  testIndexes <- which(folds==i,arr.ind=TRUE)
  y_app_cv<-Observations[-testIndexes,8]
  X_app_cv<-Observations[-testIndexes,1:p]
  y_test_cv<-Observations[testIndexes,8]
  X_test_cv<-Observations[testIndexes,1:p]

  Muniversel <- km(formula = ~y_app_cv~1+.+.^2, coef.trend = NULL, design = X_app_cv, response = y_app_c
  preduniversel <- predict(object = Muniversel, newdata = X_test_cv, type = "UK")
  meanuniversel <- preduniversel$mean

  lR[i] <- rmse(y_test_cv, meanuniversel)
```

```
}
print(mean(lR))
```

```
## [1] 0.001265007
```

6. Optimisation par LOO

Dans les dernières méthodes, l'optimisation de la portée n'est pas simple puisque on parle d'un vecteur de 7 composantes et pas d'un simple réel, pour cette raison on utilise la méthode de leave-one-out et la méthode du maximum de vraisemblance.

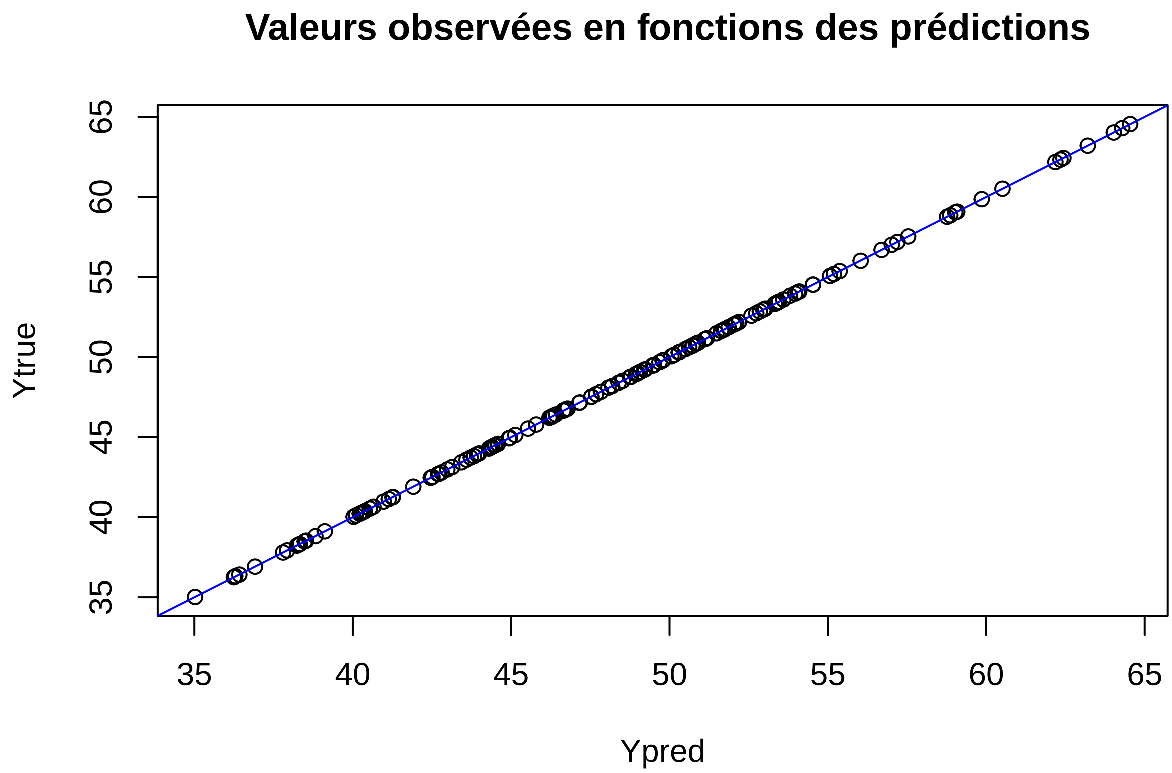
Dans l'implémentation de ces deux méthodes, on varie à chaque fois la structure de covariance "covtype" entre "gauss", "matern5_2", "matern3_2", "exp" ou "powexp" et dans tous les cas la covariance gaussienne nous donne les meilleur résultats.

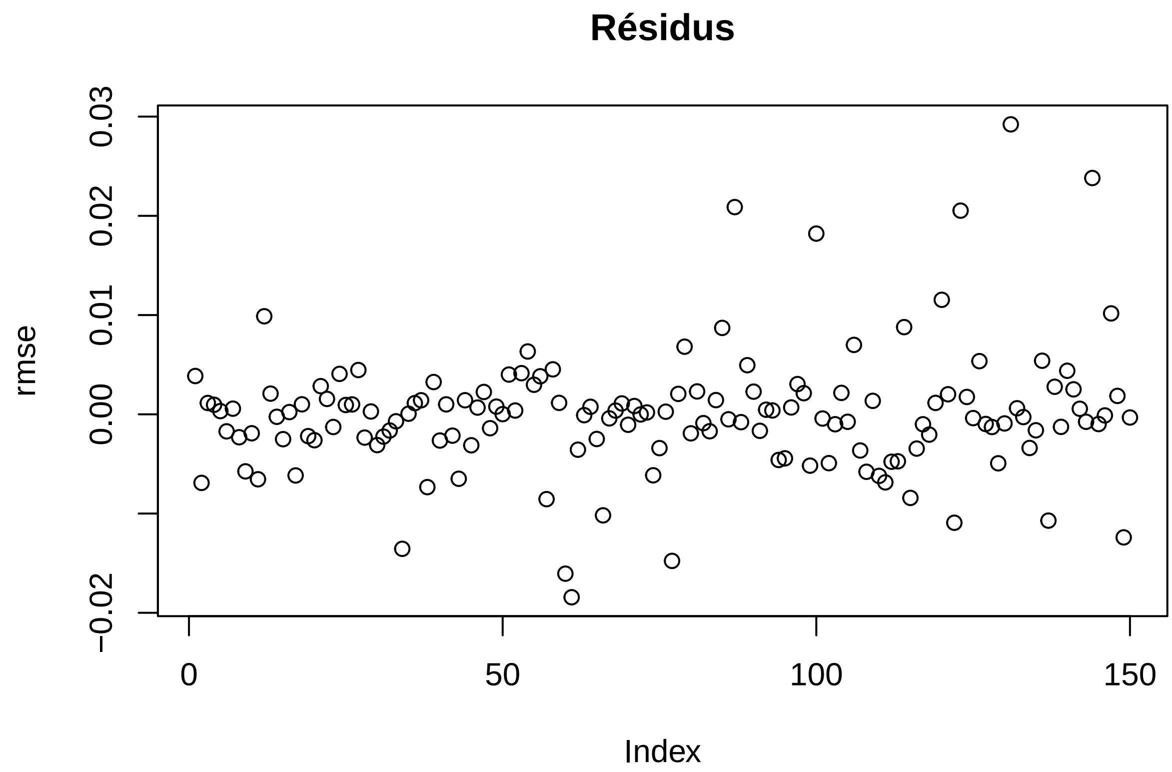
```
famille="matern5_2"
Mloo<-km(formula=~.^2,design=X_app,response=y_app,covtype="gauss",coef.trend=NULL,estim.method = "LOO")
```

```
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.987558 1.997227 1.996341 1.992466 1.993762
##   - best initial criterion value(s) :  0.002371201
##
## N = 7, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=    0.0023712  |proj g|=     0.022418
## At iterate     1  f =    0.0019121  |proj g|=      0.015991
## At iterate     2  f =    0.0012673  |proj g|=     0.0067028
## At iterate     3  f =    0.0010391  |proj g|=     0.0037562
## At iterate     4  f =   0.00085974  |proj g|=     0.0021381
## At iterate     5  f =   0.00067381  |proj g|=     0.0014966
## At iterate     6  f =   0.00050924  |proj g|=      0.002225
## At iterate     7  f =   0.00039184  |proj g|=    0.00075765
## At iterate     8  f =   0.00031826  |proj g|=    0.00045745
## At iterate     9  f =   0.00021033  |proj g|=    0.00084691
## At iterate    10  f =    0.0001872  |proj g|=    0.00037111
## At iterate    11  f =   0.00011452  |proj g|=    0.00020296
## At iterate    12  f =   7.4145e-05  |proj g|=    8.6717e-05
## At iterate    13  f =    5.619e-05  |proj g|=    6.8228e-05
```

```
## At iterate    14  f =   4.0166e-05  |proj g|=    9.8244e-05
## At iterate    15  f =   3.8851e-05  |proj g|=    5.5958e-05
## At iterate    16  f =   3.8519e-05  |proj g|=    1.6897e-05
## At iterate    17  f =   3.8423e-05  |proj g|=    1.6904e-05
## At iterate    18  f =   3.7277e-05  |proj g|=    1.7874e-05
## At iterate    19  f =   3.5767e-05  |proj g|=    2.0386e-05
## At iterate    20  f =   3.5586e-05  |proj g|=    1.0585e-05
## At iterate    21  f =   3.5505e-05  |proj g|=    1.1551e-06
## At iterate    22  f =   3.5505e-05  |proj g|=    3.0987e-06
##
## iterations 22
## function evaluations 24
## segments explored during Cauchy searches 23
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 3.09875e-06
## final function value 3.55045e-05
##
## F = 3.55045e-05
## final  value 0.000036
## converged
```

```
bestThetaLOO = coef(Mloo, "range")
sd2LOO = coef(Mloo, "sd2")
predLOO<-predict(object=Mloo,newdata=X_test,type="UK",ckeckNames=FALSE )
MeanLOO<-predLOO$mean
rmse(y_test, MeanLOO, trace=TRUE)
```


Valeurs observées en fonctions des prédictions
Ytrue
Ypred
35
40
45
50
55
60
65




```
## [1] 0.006303565
```

```r
message("DiceKriging: par optimisation MLE, theta= ", bestThetaLOO)
```

```
## DiceKriging: par optimisation MLE, theta= 0.3938690598057541.998102292418481.987557558808481.9972273
```

6.a Optimisation par LOO CV

```r
Observations<-Observations[sample(nrow(Observations)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(Observations)),breaks=5,labels=FALSE)
lR<-seq(1,5)
#Perform 10 fold cross validation
for(i in 1:5){
  #Segement your data by fold using the which() function
  testIndexes <- which(folds==i,arr.ind=TRUE)
  y_app_cv<-Observations[-testIndexes,8]
  X_app_cv<-Observations[-testIndexes,1:p]
  y_test_cv<-Observations[testIndexes,8]
  X_test_cv<-Observations[testIndexes,1:p]

  Mloo<-km(formula=~.^2,design=X_app_cv,response=y_app_cv,covtype="gauss",coef.trend=NULL,estim.method =
```

```r
  bestThetaLOO = coef(Mloo, "range")
  sd2LOO = coef(Mloo, "sd2")
  predLOO<-predict(object=Mloo,newdata=X_test_cv,type="UK",ckeckNames=FALSE )
  MeanLOO<-predLOO$mean

  lR[i] <- rmse(y_test_cv, MeanLOO)
}
```

```
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.982769 1.998337 1.996341 1.996626 1.991921
##   - best initial criterion value(s) :  0.001477276
##
## N = 7, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=    0.0014773  |proj g|=    0.0067133
## At iterate     1  f =    0.0014247  |proj g|=     0.0065746
## At iterate     2  f =   0.00092759  |proj g|=    0.00087924
## At iterate     3  f =   0.00063567  |proj g|=    0.00087569
## At iterate     4  f =   0.00032165  |proj g|=     0.0026527
## At iterate     5  f =   0.00013515  |proj g|=    0.00074148
## At iterate     6  f =   0.00010375  |proj g|=    0.00021058
## At iterate     7  f =   7.5594e-05  |proj g|=    9.7586e-05
## At iterate     8  f =    4.452e-05  |proj g|=    0.00012336
## At iterate     9  f =   4.1504e-05  |proj g|=    9.1811e-05
## At iterate    10  f =   3.8098e-05  |proj g|=    4.3778e-05
## At iterate    11  f =   3.7297e-05  |proj g|=    2.4199e-05
## At iterate    12  f =   3.6409e-05  |proj g|=    1.8073e-05
## At iterate    13  f =   3.4204e-05  |proj g|=    1.6962e-05
## At iterate    14  f =    3.231e-05  |proj g|=    1.5142e-05
## At iterate    15  f =   3.1108e-05  |proj g|=    3.4583e-05
## At iterate    16  f =   3.0946e-05  |proj g|=    1.0029e-05
## At iterate    17  f =   3.0917e-05  |proj g|=    9.4372e-07
## At iterate    18  f =   3.0916e-05  |proj g|=    8.7536e-07
##
## iterations 18
## function evaluations 21
## segments explored during Cauchy searches 19
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 8.75356e-07
```

```
## final function value 3.09157e-05
##
## F = 3.09157e-05
## final  value 0.000031
## converged
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.988234 1.998102 1.987558 1.990277 1.994731 1.996244 1.987477
##   - best initial criterion value(s) :  0.002717226
##
## N = 7, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=    0.0027172  |proj g|=    0.0097358
## At iterate     1  f =    0.0025977  |proj g|=     0.0093399
## At iterate     2  f =   0.00099668  |proj g|=    0.00099835
## At iterate     3  f =   0.00088434  |proj g|=    0.00079969
## At iterate     4  f =   0.00063578  |proj g|=     0.0010631
## At iterate     5  f =   0.00020537  |proj g|=      0.001024
## At iterate     6  f =   0.00016776  |proj g|=     0.0023293
## At iterate     7  f =   7.3289e-05  |proj g|=    0.00069068
## At iterate     8  f =   5.7825e-05  |proj g|=    0.00044391
## At iterate     9  f =   4.3061e-05  |proj g|=    0.00020573
## At iterate    10  f =   3.7082e-05  |proj g|=    5.5601e-05
## At iterate    11  f =   3.5016e-05  |proj g|=    5.7728e-05
## At iterate    12  f =    3.207e-05  |proj g|=    4.0215e-05
## At iterate    13  f =   2.9838e-05  |proj g|=     1.622e-05
## At iterate    14  f =   2.9261e-05  |proj g|=    1.4073e-05
## At iterate    15  f =   2.8969e-05  |proj g|=    1.0117e-05
## At iterate    16  f =   2.7701e-05  |proj g|=    9.1691e-06
## At iterate    17  f =   2.7645e-05  |proj g|=     6.594e-06
## At iterate    18  f =   2.7627e-05  |proj g|=    7.0306e-06
## At iterate    19  f =   2.7617e-05  |proj g|=    2.2449e-07
## At iterate    20  f =   2.7617e-05  |proj g|=    1.3267e-09
##
## iterations 20
## function evaluations 21
## segments explored during Cauchy searches 20
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 1.32671e-09
## final function value 2.76173e-05
##
```

```
## F = 2.76173e-05
## final  value 0.000028
## converged
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.987558 1.998337 1.996341 1.996626 1.993762
##   - best initial criterion value(s) :  0.00535388
##
## N = 7, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=    0.0053539  |proj g|=     0.034338
## At iterate     1  f =    0.0038921  |proj g|=      0.016898
## At iterate     2  f =    0.0030867  |proj g|=       0.01077
## At iterate     3  f =    0.0023486  |proj g|=     0.0063883
## At iterate     4  f =    0.0016151  |proj g|=      0.003315
## At iterate     5  f =   0.00098939  |proj g|=     0.0017544
## At iterate     6  f =   0.00026112  |proj g|=    0.00050641
## At iterate     7  f =   0.00016384  |proj g|=    0.00090227
## At iterate     8  f =   0.00010298  |proj g|=    0.00015131
## At iterate     9  f =   8.8119e-05  |proj g|=    0.00014444
## At iterate    10  f =   7.7097e-05  |proj g|=    6.5739e-05
## At iterate    11  f =    5.416e-05  |proj g|=    0.00015439
## At iterate    12  f =   5.0826e-05  |proj g|=    0.00012659
## At iterate    13  f =    4.682e-05  |proj g|=    4.5259e-05
## At iterate    14  f =      3.9e-05  |proj g|=    0.00012641
## At iterate    15  f =   3.8034e-05  |proj g|=     6.219e-05
## At iterate    16  f =    3.702e-05  |proj g|=    2.2132e-05
## At iterate    17  f =   3.6411e-05  |proj g|=    5.8173e-05
## At iterate    18  f =     3.55e-05  |proj g|=    7.1271e-05
## At iterate    19  f =   3.4913e-05  |proj g|=    4.6208e-05
## At iterate    20  f =   3.3567e-05  |proj g|=    5.4087e-05
## At iterate    21  f =   3.3124e-05  |proj g|=    6.7728e-05
## At iterate    22  f =   3.2137e-05  |proj g|=    4.5975e-05
## At iterate    23  f =   3.1906e-05  |proj g|=    8.0565e-06
## At iterate    24  f =   3.1885e-05  |proj g|=    5.3058e-06
## At iterate    25  f =   3.1872e-05  |proj g|=    7.3458e-06
## At iterate    26  f =   3.1853e-05  |proj g|=    5.5428e-06
## At iterate    27  f =    3.185e-05  |proj g|=    4.3748e-07
## At iterate    28  f =    3.185e-05  |proj g|=     1.933e-08
##
## iterations 28
## function evaluations 29
```

```
## segments explored during Cauchy searches 28
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 1.93303e-08
## final function value 3.18495e-05
##
## F = 3.18495e-05
## final  value 0.000032
## converged
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.985485 1.981771 1.987558 1.997227 1.996341 1.996626 1.993762
##   - best initial criterion value(s) :  0.03737871
##
## N = 7, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=     0.037379  |proj g|=       0.2983
## At iterate     1  f =    0.0020855  |proj g|=      0.012149
## At iterate     2  f =    0.0019141  |proj g|=      0.010752
## At iterate     3  f =   0.00099103  |proj g|=     0.0042901
## At iterate     4  f =   0.00066222  |proj g|=     0.0022559
## At iterate     5  f =    0.0004514  |proj g|=     0.0011128
## At iterate     6  f =   0.00031571  |proj g|=    0.00062885
## At iterate     7  f =    0.0002214  |proj g|=     0.0003432
## At iterate     8  f =   0.00014568  |proj g|=    0.00017238
## At iterate     9  f =   9.7456e-05  |proj g|=    0.00038298
## At iterate    10  f =   6.2446e-05  |proj g|=    4.5998e-05
## At iterate    11  f =   4.3912e-05  |proj g|=    2.2325e-05
## At iterate    12  f =   3.8424e-05  |proj g|=    3.8351e-05
## At iterate    13  f =   3.5815e-05  |proj g|=    1.5359e-05
## At iterate    14  f =   3.5597e-05  |proj g|=    4.1006e-06
## At iterate    15  f =   3.5593e-05  |proj g|=    3.6111e-06
## At iterate    16  f =   3.5578e-05  |proj g|=    1.3142e-06
## At iterate    17  f =   3.5578e-05  |proj g|=    3.6571e-07
##
## iterations 17
## function evaluations 18
## segments explored during Cauchy searches 18
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 3.6571e-07
## final function value 3.55782e-05
```

```
##
## F = 3.55782e-05
## final  value 0.000036
## converged
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.978714 1.998337 1.994373 1.992466 1.993762
##   - best initial criterion value(s) :  0.01244225
##
## N = 7, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=     0.012442  |proj g|=     0.096037
## At iterate     1  f =    0.0055617  |proj g|=      0.041115
## At iterate     2  f =    0.0031205  |proj g|=      0.020799
## At iterate     3  f =    0.0019552  |proj g|=     0.0079473
## At iterate     4  f =    0.0015773  |proj g|=     0.0035791
## At iterate     5  f =     0.001268  |proj g|=     0.0019252
## At iterate     6  f =   0.00086485  |proj g|=      0.001247
## At iterate     7  f =   0.00050465  |proj g|=     0.0012969
## At iterate     8  f =   0.00029607  |proj g|=     0.0011661
## At iterate     9  f =   0.00020414  |proj g|=    0.00038552
## At iterate    10  f =   0.00013932  |proj g|=    0.00051184
## At iterate    11  f =   8.9135e-05  |proj g|=    9.3122e-05
## At iterate    12  f =   6.5885e-05  |proj g|=    4.9652e-05
## At iterate    13  f =   5.0252e-05  |proj g|=     4.085e-05
## At iterate    14  f =   4.2698e-05  |proj g|=    2.5909e-05
## At iterate    15  f =   3.9141e-05  |proj g|=    2.2724e-05
## At iterate    16  f =   3.8106e-05  |proj g|=    0.00010037
## At iterate    17  f =   3.6527e-05  |proj g|=    3.2423e-05
## At iterate    18  f =   3.5799e-05  |proj g|=    1.4282e-05
## At iterate    19  f =   3.4377e-05  |proj g|=     7.375e-06
## At iterate    20  f =   3.4358e-05  |proj g|=    3.0506e-06
## At iterate    21  f =   3.4344e-05  |proj g|=    2.6928e-06
## At iterate    22  f =   3.4343e-05  |proj g|=    1.5372e-06
##
## iterations 22
## function evaluations 24
## segments explored during Cauchy searches 23
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 1.53719e-06
## final function value 3.4343e-05
```

```
##
## F = 3.4343e-05
## final  value 0.000034
## converged
```

```
print(mean(lR))
```

```
## [1] 0.005742813
```

7. Optimisation par MLE

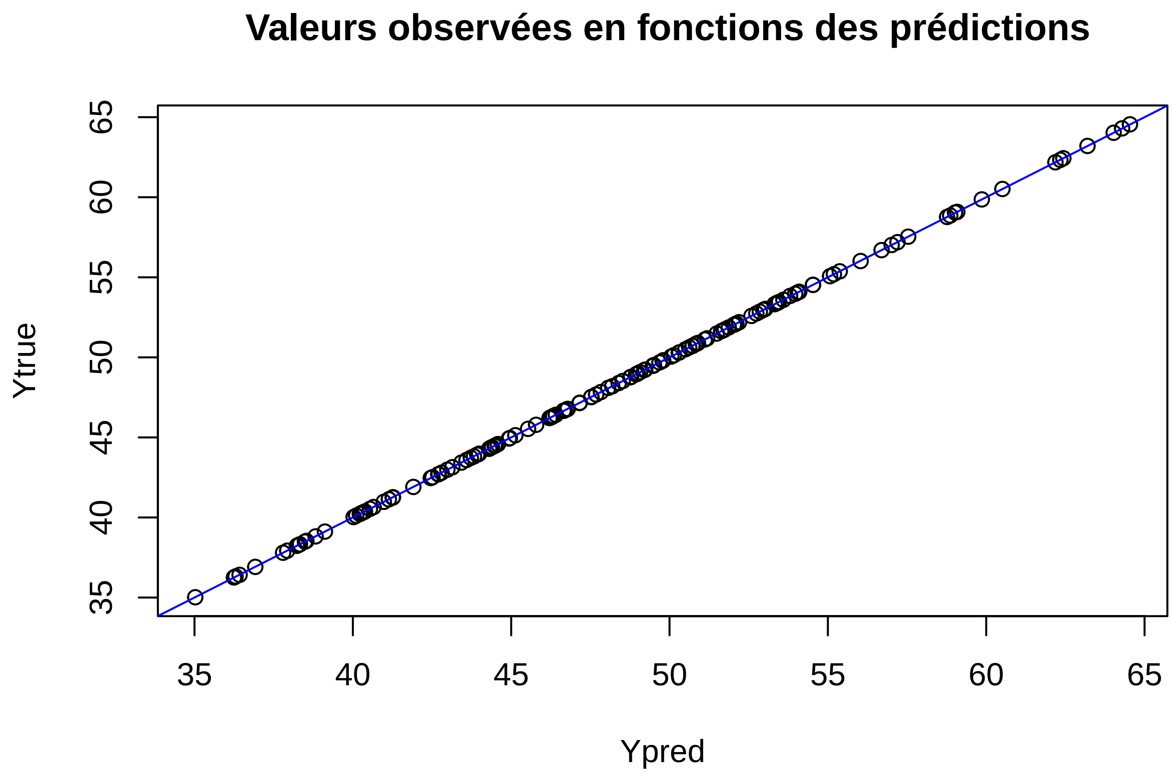
```
MMLE <- km(formula = ~.^2, design = X_app, response = y_app, covtype = "gauss", coef.trend = NULL,  esti
```

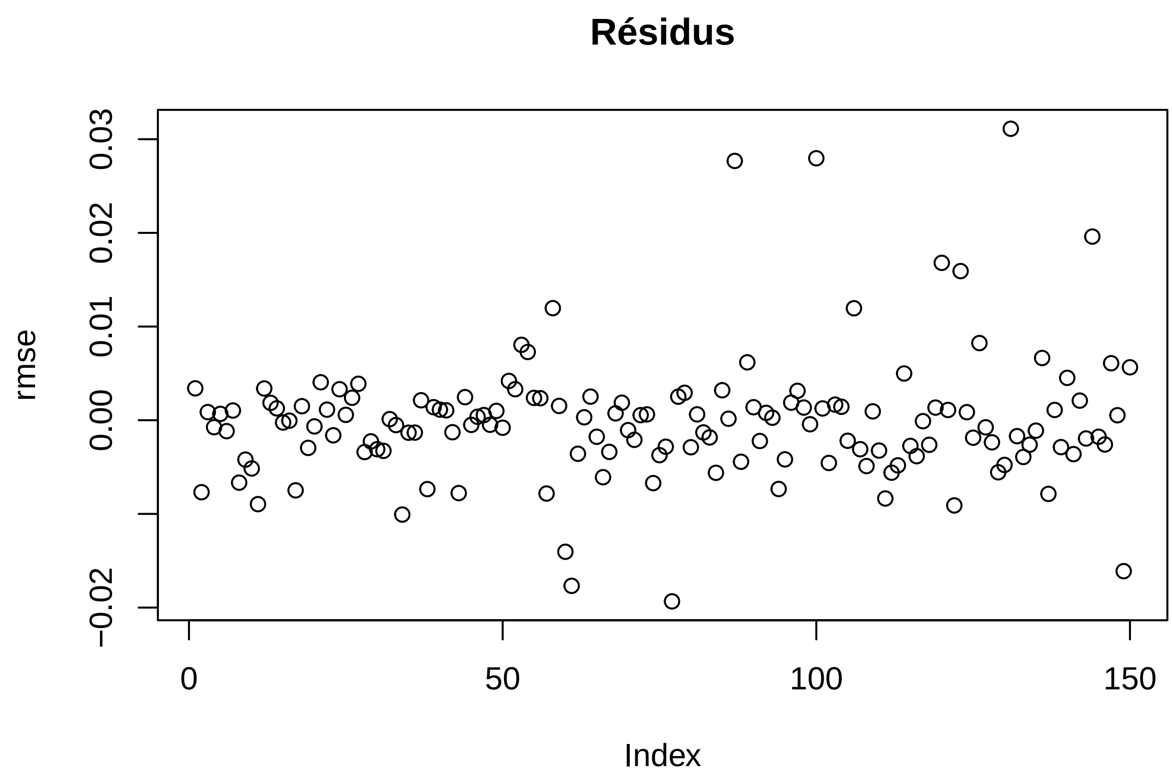
```
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : 1e-13
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.987558 1.997227 1.996341 1.992466 1.993762
##   - variance bounds :  0.06301832 6.547727
##   - best initial criterion value(s) :  -488.6816
##
## N = 8, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=       488.68  |proj g|=       5.6976
## At iterate     1  f =       476.86  |proj g|=        1.9379
## At iterate     2  f =       474.24  |proj g|=        1.9347
## At iterate     3  f =       464.16  |proj g|=        1.8939
## At iterate     4  f =       446.51  |proj g|=        1.9486
## At iterate     5  f =       444.57  |proj g|=        1.9699
## At iterate     6  f =       439.15  |proj g|=        1.9917
## At iterate     7  f =       438.93  |proj g|=        1.9917
## At iterate     8  f =       438.73  |proj g|=        1.9917
## At iterate     9  f =       438.18  |proj g|=        1.9917
## At iterate    10  f =       437.99  |proj g|=        1.9917
## At iterate    11  f =       437.83  |proj g|=        1.9917
## At iterate    12  f =        437.5  |proj g|=        1.9916
## At iterate    13  f =       436.45  |proj g|=        1.9908
## At iterate    14  f =       434.53  |proj g|=        1.9884
## At iterate    15  f =        432.9  |proj g|=        6.0438
## At iterate    16  f =       432.49  |proj g|=        6.0642
```

```
## At iterate    17  f =       432.39  |proj g|=        6.0725
## At iterate    18  f =       432.25  |proj g|=        6.0788
## At iterate    19  f =       432.03  |proj g|=        6.0824
## At iterate    20  f =       431.86  |proj g|=        6.0803
## At iterate    21  f =          431  |proj g|=        6.0739
## At iterate    22  f =       428.83  |proj g|=        1.9622
## At iterate    23  f =       428.35  |proj g|=        6.0683
## At iterate    24  f =       425.83  |proj g|=         1.966
## At iterate    25  f =       425.44  |proj g|=        6.0413
## At iterate    26  f =       425.21  |proj g|=        6.0343
## At iterate    27  f =       425.09  |proj g|=        4.4456
## At iterate    28  f =       425.01  |proj g|=        1.9084
## At iterate    29  f =       424.97  |proj g|=        1.9024
## At iterate    30  f =       424.94  |proj g|=          1.92
## At iterate    31  f =       424.92  |proj g|=        1.9654
## At iterate    32  f =       424.82  |proj g|=        1.9652
## At iterate    33  f =       424.64  |proj g|=        1.9651
## At iterate    34  f =       424.42  |proj g|=        1.9651
## At iterate    35  f =       424.21  |proj g|=        6.0045
## At iterate    36  f =       423.95  |proj g|=        6.0048
## At iterate    37  f =       423.67  |proj g|=        4.5255
## At iterate    38  f =       423.25  |proj g|=        6.0143
## At iterate    39  f =       422.75  |proj g|=        6.0106
## At iterate    40  f =       422.58  |proj g|=        6.0062
## At iterate    41  f =       422.17  |proj g|=        5.9956
## At iterate    42  f =       421.72  |proj g|=        4.6187
## At iterate    43  f =          421  |proj g|=        1.6038
## At iterate    44  f =       420.41  |proj g|=        1.4714
## At iterate    45  f =       420.28  |proj g|=        1.4664
## At iterate    46  f =       419.41  |proj g|=        1.3983
## At iterate    47  f =        418.2  |proj g|=         1.952
## At iterate    48  f =       416.84  |proj g|=        1.9528
## At iterate    49  f =       413.44  |proj g|=        1.3192
## At iterate    50  f =       406.55  |proj g|=        6.0079
## At iterate    51  f =       403.89  |proj g|=        1.8647
## At iterate    52  f =       402.34  |proj g|=        4.8798
## At iterate    53  f =       400.49  |proj g|=        1.9501
## At iterate    54  f =       398.38  |proj g|=        1.9558
## At iterate    55  f =       396.44  |proj g|=         1.963
## At iterate    56  f =       388.93  |proj g|=        1.9607
## At iterate    57  f =       369.14  |proj g|=         1.953
## At iterate    58  f =        176.4  |proj g|=        1.9545
## At iterate    59  f =       168.62  |proj g|=        1.9539
## At iterate    60  f =      -229.69  |proj g|=        1.9152
## At iterate    61  f =       -342.1  |proj g|=         1.868
## At iterate    62  f =      -420.94  |proj g|=        1.8479
## At iterate    63  f =      -465.88  |proj g|=        6.3964
## At iterate    64  f =       -669.1  |proj g|=        6.3389
## At iterate    65  f =      -981.33  |proj g|=        6.3271
## At iterate    66  f =      -1109.9  |proj g|=        6.2025
## At iterate    67  f =      -1216.1  |proj g|=        6.1998
## At iterate    68  f =      -1234.2  |proj g|=        6.1592
## At iterate    69  f =      -1287.6  |proj g|=        5.9928
## At iterate    70  f =      -1320.9  |proj g|=        5.9277
```

```
## At iterate    71  f =      -1326.9  |proj g|=       5.8896
## At iterate    72  f =      -1330.4  |proj g|=       5.8384
## At iterate    73  f =      -1333.6  |proj g|=         5.77
## At iterate    74  f =        -1336  |proj g|=       5.7242
## At iterate    75  f =      -1336.7  |proj g|=       5.7156
## At iterate    76  f =      -1337.3  |proj g|=       5.6904
## At iterate    77  f =      -1337.4  |proj g|=       5.6808
## At iterate    78  f =      -1337.8  |proj g|=       1.6116
## At iterate    79  f =      -1338.6  |proj g|=       1.6092
## At iterate    80  f =      -1339.4  |proj g|=       1.6044
## At iterate    81  f =      -1339.4  |proj g|=        1.603
## At iterate    82  f =      -1339.5  |proj g|=       1.6025
## At iterate    83  f =      -1339.6  |proj g|=       1.3802
## At iterate    84  f =      -1339.6  |proj g|=      0.20326
## At iterate    85  f =      -1339.6  |proj g|=     0.013945
## At iterate    86  f =      -1339.6  |proj g|=   0.00034213
##
## iterations 86
## function evaluations 119
## segments explored during Cauchy searches 94
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 0.00034213
## final function value -1339.62
##
## F = -1339.62
## final  value -1339.616662
## converged
```

```r
predMLE <- predict(object = MMLE, newdata = X_test , type="UK" , checkNames=FALSE, se.compute=TRUE)
MeanMLE<-predMLE$mean
rmse(y_test, MeanMLE, trace=TRUE)
```


Valeurs observées en fonctions des prédictions
Ytrue
35
40
45
50
55
60
65
Ypred
35
40
45
50
55
60
65




```
## [1] 0.006756957
```

```
bestThetaMLE = coef(MMLE, "range")
sd2MLE = coef(MMLE, "sd2")

message("DiceKriging: par optimisation MLE, theta= ", bestThetaMLE)
```

```
## DiceKriging: par optimisation MLE, theta= 0.3880583424710951.998102292418481.987557558808481.99722730
```

7.a Optimisation par MLE avec CV

```
Observations<-Observations[sample(nrow(Observations)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(Observations)),breaks=5,labels=FALSE)
lR<-seq(1,5)
#Perform 10 fold cross validation
for(i in 1:5){
  #Segement your data by fold using the which() function
  testIndexes <- which(folds==i,arr.ind=TRUE)
  y_app_cv<-Observations[-testIndexes,8]
  X_app_cv<-Observations[-testIndexes,1:p]
  y_test_cv<-Observations[testIndexes,8]
```

```
  X_test_cv<-Observations[testIndexes,1:p]

  MMLE <- km(formula = ~.^2, design = X_app_cv, response = y_app_cv, covtype = "gauss", coef.trend = NU
  predMLE <- predict(object = MMLE, newdata = X_test_cv , type="UK" , checkNames=FALSE, se.compute=TRUE
  MeanMLE<-predMLE$mean

  lR[i] <- rmse(y_test_cv, MeanMLE)
  #print(rmse(y_test_cv, Meanordinaire))
}
```

```
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : 1e-13
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.988534 1.987558 1.998337 1.996341 1.996244 1.991921
##   - variance bounds :  0.06654336 6.962068
##   - best initial criterion value(s) :  135.4504
##
## N = 8, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=      -135.45  |proj g|=       6.0087
## At iterate     1  f =      -716.69  |proj g|=        2.4388
## At iterate     2  f =      -873.48  |proj g|=        2.6765
## At iterate     3  f =      -896.44  |proj g|=        2.7661
## At iterate     4  f =      -941.69  |proj g|=        2.8959
## At iterate     5  f =      -1203.2  |proj g|=        3.6353
## At iterate     6  f =        -1219  |proj g|=        3.6482
## At iterate     7  f =      -1245.8  |proj g|=         3.659
## At iterate     8  f =      -1255.5  |proj g|=          3.63
## At iterate     9  f =      -1295.9  |proj g|=        3.3884
## At iterate    10  f =      -1310.8  |proj g|=        3.3031
## At iterate    11  f =      -1320.1  |proj g|=        3.3047
## At iterate    12  f =      -1325.1  |proj g|=        3.3046
## At iterate    13  f =        -1329  |proj g|=        3.2975
## At iterate    14  f =      -1329.3  |proj g|=        3.2922
## At iterate    15  f =      -1329.4  |proj g|=        3.2872
## At iterate    16  f =      -1329.6  |proj g|=        3.2717
## At iterate    17  f =      -1330.3  |proj g|=         3.229
## At iterate    18  f =      -1332.2  |proj g|=        3.1034
## At iterate    19  f =      -1337.1  |proj g|=        2.7682
## At iterate    20  f =      -1347.1  |proj g|=        1.9581
## At iterate    21  f =      -1360.3  |proj g|=        5.5046
## At iterate    22  f =      -1384.9  |proj g|=        6.1232
```

```
## At iterate    23  f =      -1387.3  |proj g|=       1.4214
## At iterate    24  f =      -1387.8  |proj g|=       1.4119
## At iterate    25  f =      -1388.2  |proj g|=       3.6333
## At iterate    26  f =      -1388.3  |proj g|=       1.6096
## At iterate    27  f =      -1388.3  |proj g|=      0.52043
## At iterate    28  f =      -1388.3  |proj g|=      0.38243
## At iterate    29  f =      -1388.3  |proj g|=      0.17303
## At iterate    30  f =      -1388.3  |proj g|=     0.026205
## At iterate    31  f =      -1388.3  |proj g|=    0.0030675
##
## iterations 31
## function evaluations 37
## segments explored during Cauchy searches 39
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 0.00306754
## final function value -1388.27
##
## F = -1388.27
## final  value -1388.266938
## converged
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : 1e-13
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.985485 1.998102 1.977007 1.998337 1.996341 1.996626 1.992937
##   - variance bounds :  0.06330323 6.759943
##   - best initial criterion value(s) :  113.7156
##
## N = 8, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=      -113.72  |proj g|=       5.8887
## At iterate     1  f =       -815.6  |proj g|=       5.2501
## At iterate     2  f =      -963.16  |proj g|=       5.2139
## At iterate     3  f =      -1083.7  |proj g|=       1.4226
## At iterate     4  f =      -1100.4  |proj g|=       1.4453
## At iterate     5  f =      -1255.3  |proj g|=       1.6163
## At iterate     6  f =      -1313.4  |proj g|=       1.5239
## At iterate     7  f =      -1329.5  |proj g|=       1.4961
## At iterate     8  f =      -1337.8  |proj g|=       1.6232
## At iterate     9  f =      -1344.5  |proj g|=       5.9964
## At iterate    10  f =      -1353.8  |proj g|=        6.024
## At iterate    11  f =      -1367.2  |proj g|=       6.0632
## At iterate    12  f =      -1374.2  |proj g|=       6.0607
```

```
## At iterate    13  f =      -1378.5  |proj g|=        6.052
## At iterate    14  f =      -1386.2  |proj g|=       3.3822
## At iterate    15  f =      -1393.7  |proj g|=       1.6142
## At iterate    16  f =      -1394.1  |proj g|=       5.6183
## At iterate    17  f =      -1394.2  |proj g|=       6.0376
## At iterate    18  f =      -1394.2  |proj g|=       6.0367
## At iterate    19  f =      -1394.2  |proj g|=       6.0355
## At iterate    20  f =      -1394.2  |proj g|=       6.0309
## At iterate    21  f =      -1394.3  |proj g|=       6.0195
## At iterate    22  f =      -1394.4  |proj g|=       5.9918
## At iterate    23  f =      -1394.6  |proj g|=       5.8746
## At iterate    24  f =      -1394.8  |proj g|=        1.608
## At iterate    25  f =        -1395  |proj g|=       1.4272
## At iterate    26  f =        -1395  |proj g|=       1.4249
## At iterate    27  f =        -1395  |proj g|=       1.0253
## At iterate    28  f =        -1395  |proj g|=        0.161
## At iterate    29  f =        -1395  |proj g|=     0.016185
##
## iterations 29
## function evaluations 34
## segments explored during Cauchy searches 36
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 0.0161854
## final function value -1395
##
## F = -1395
## final  value -1394.999973
## converged
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : 1e-13
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.987558 1.998337 1.996341 1.992466 1.993762
##   - variance bounds :  0.06078994 6.34027
##   - best initial criterion value(s) :  -197.2073
##
## N = 8, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=       197.21  |proj g|=        1.802
## At iterate     1  f =      -447.95  |proj g|=       5.7692
## At iterate     2  f =       -517.3  |proj g|=        5.777
## At iterate     3  f =      -584.47  |proj g|=       5.7779
## At iterate     4  f =      -841.11  |proj g|=        1.709
```

```
## At iterate     5  f =        -1060  |proj g|=       1.7037
## At iterate     6  f =      -1351.6  |proj g|=       5.8385
## At iterate     7  f =      -1371.9  |proj g|=       5.8258
## At iterate     8  f =      -1377.1  |proj g|=       5.8123
## At iterate     9  f =      -1381.1  |proj g|=       5.7986
## At iterate    10  f =      -1382.1  |proj g|=       5.7882
## At iterate    11  f =      -1383.5  |proj g|=       5.7679
## At iterate    12  f =      -1386.2  |proj g|=       5.7105
## At iterate    13  f =      -1391.1  |proj g|=       1.4796
## At iterate    14  f =      -1395.5  |proj g|=        1.607
## At iterate    15  f =      -1395.6  |proj g|=        1.402
## At iterate    16  f =      -1395.6  |proj g|=       1.6057
## At iterate    17  f =      -1395.7  |proj g|=       1.0774
## At iterate    18  f =      -1395.7  |proj g|=      0.69665
## At iterate    19  f =      -1395.7  |proj g|=      0.18507
## At iterate    20  f =      -1395.7  |proj g|=     0.089515
## At iterate    21  f =      -1395.7  |proj g|=        0.131
##
## iterations 21
## function evaluations 26
## segments explored during Cauchy searches 29
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 0.131005
## final function value -1395.67
##
## F = -1395.67
## final  value -1395.671838
## converged
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : 1e-13
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.989814 1.998102 1.987558 1.983115 1.990315 1.996626 1.993762
##   - variance bounds :  0.06330028 6.81084
##   - best initial criterion value(s) :  832.977
##
## N = 8, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=      -832.98  |proj g|=       1.8007
## At iterate     1  f =      -1239.9  |proj g|=       1.6513
## At iterate     2  f =      -1259.9  |proj g|=       6.3178
## At iterate     3  f =      -1271.7  |proj g|=       6.3122
## At iterate     4  f =      -1274.1  |proj g|=       6.3113
```

```
## At iterate     5  f =        -1309  |proj g|=        6.2763
## At iterate     6  f =      -1359.2  |proj g|=         6.206
## At iterate     7  f =      -1364.7  |proj g|=        6.1955
## At iterate     8  f =      -1368.1  |proj g|=        6.1876
## At iterate     9  f =      -1371.6  |proj g|=        6.1714
## At iterate    10  f =      -1375.6  |proj g|=        1.6252
## At iterate    11  f =      -1376.9  |proj g|=         6.153
## At iterate    12  f =        -1378  |proj g|=        6.1498
## At iterate    13  f =      -1378.2  |proj g|=        6.1487
## At iterate    14  f =      -1378.3  |proj g|=        6.1451
## At iterate    15  f =      -1378.7  |proj g|=        6.1332
## At iterate    16  f =      -1379.7  |proj g|=        6.0918
## At iterate    17  f =      -1381.8  |proj g|=        5.9895
## At iterate    18  f =      -1381.9  |proj g|=        5.9792
## At iterate    19  f =        -1384  |proj g|=        5.8123
## At iterate    20  f =        -1385  |proj g|=        5.7269
## At iterate    21  f =      -1385.1  |proj g|=        2.2969
## At iterate    22  f =      -1385.1  |proj g|=        1.6003
## At iterate    23  f =      -1385.1  |proj g|=       0.11052
## At iterate    24  f =      -1385.1  |proj g|=       0.41142
## At iterate    25  f =      -1385.1  |proj g|=      0.047925
##
## iterations 25
## function evaluations 31
## segments explored during Cauchy searches 32
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 0.0479251
## final function value -1385.14
##
## F = -1385.14
## final  value -1385.141365
## converged
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X1:X2 + X1:X3 + X1:X4 + X1:X5 +
##     X1:X6 + X1:X7 + X2:X3 + X2:X4 + X2:X5 + X2:X6 + X2:X7 + X3:X4 +
##     X3:X5 + X3:X6 + X3:X7 + X4:X5 + X4:X6 + X4:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : 1e-13
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.986204 1.98205 1.998337 1.996341 1.996626 1.993762
##   - variance bounds :  0.06527738 6.860711
##   - best initial criterion value(s) :  30.36991
##
## N = 8, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=       -30.37  |proj g|=       1.8271
```

```
## At iterate     1  f =      -636.25  |proj g|=       6.4848
## At iterate     2  f =      -890.98  |proj g|=        6.451
## At iterate     3  f =      -977.06  |proj g|=       6.4451
## At iterate     4  f =      -1176.9  |proj g|=        6.411
## At iterate     5  f =      -1316.2  |proj g|=       6.3328
## At iterate     6  f =      -1338.4  |proj g|=       6.3085
## At iterate     7  f =      -1366.5  |proj g|=       6.2809
## At iterate     8  f =      -1368.6  |proj g|=       6.2739
## At iterate     9  f =      -1375.9  |proj g|=       6.2499
## At iterate    10  f =      -1376.7  |proj g|=        6.247
## At iterate    11  f =      -1377.8  |proj g|=       6.2417
## At iterate    12  f =      -1378.7  |proj g|=        6.237
## At iterate    13  f =        -1379  |proj g|=       6.2361
## At iterate    14  f =      -1379.2  |proj g|=       6.2327
## At iterate    15  f =      -1379.5  |proj g|=       6.2261
## At iterate    16  f =      -1380.2  |proj g|=       6.1986
## At iterate    17  f =      -1381.7  |proj g|=       6.1348
## At iterate    18  f =      -1383.6  |proj g|=       5.9608
## At iterate    19  f =        -1384  |proj g|=       1.6218
## At iterate    20  f =      -1387.3  |proj g|=       3.8472
## At iterate    21  f =      -1387.9  |proj g|=       5.8113
## At iterate    22  f =      -1387.9  |proj g|=       1.7441
## At iterate    23  f =      -1387.9  |proj g|=       1.3915
## At iterate    24  f =      -1387.9  |proj g|=      0.62138
## At iterate    25  f =      -1387.9  |proj g|=     0.021347
## At iterate    26  f =      -1387.9  |proj g|=    0.0020919
##
## iterations 26
## function evaluations 31
## segments explored during Cauchy searches 33
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 5
## norm of the final projected gradient 0.00209189
## final function value -1387.94
##
## F = -1387.94
## final  value -1387.936669
## converged
```

```r
print(mean(lR))
```

```
## [1] 0.006683068
```

Dans toutes les méthodes précedentes, avec ou sans cross validation, on obtient une valeur de RMSE de l'ordre de 10^{-3} qui est déjà une valeur très petite par rapport à la moyenne calculée dans la première partie. Pourtant, on va essayer une autre méthode afin de diminuer de plus la valeur RMSE.

8. Régression linéaire

On fait une régression entre y et les 7 variables par une interaction de deuxième ordre afin de voir les variables les moins significatifs

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
# cross validation
train_control <- trainControl(method = "cv",
                              number = 5)
rlm=lm(y_app~1+.+.^2, data=X_app,trControl = train_control)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trControl' will be disregarded
```

```r
summary(rlm)
```

```
##
## Call:
## lm(formula = y_app ~ 1 + . + .^2, data = X_app, trControl = train_control)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1897 -0.5795 -0.1202  0.5229  2.1785
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  54.56261    0.50275 108.528  < 2e-16 ***
## X1           11.25255    0.52431  21.462  < 2e-16 ***
## X2          -13.07802    0.55493 -23.567  < 2e-16 ***
## X3            0.08913    0.54720   0.163   0.8707
## X4            0.18531    0.51606   0.359   0.7197
## X5            5.02738    0.53714   9.360  < 2e-16 ***
## X6          -12.68135    0.51783 -24.489  < 2e-16 ***
## X7           -0.44855    0.58679  -0.764   0.4450
## X1:X2        -4.44755    0.44926  -9.900  < 2e-16 ***
## X1:X3        -0.51295    0.43528  -1.178   0.2392
## X1:X4        -0.52687    0.43289  -1.217   0.2241
## X1:X5        -0.18644    0.44295  -0.421   0.6740
## X1:X6        -0.02597    0.41385  -0.063   0.9500
## X1:X7         0.41368    0.44917   0.921   0.3575
## X2:X3         0.11868    0.43086   0.275   0.7831
## X2:X4         0.37899    0.43207   0.877   0.3808
## X2:X5         0.96356    0.45311   2.127   0.0339 *
## X2:X6         0.96824    0.43251   2.239   0.0256 *
## X2:X7         0.73401    0.45124   1.627   0.1044
## X3:X4        -0.15315    0.42241  -0.363   0.7171
## X3:X5         0.03389    0.44756   0.076   0.9397
## X3:X6         0.42932    0.42944   1.000   0.3179
## X3:X7        -0.29994    0.44474  -0.674   0.5003
## X4:X5         0.08462    0.43857   0.193   0.8471
## X4:X6        -0.35441    0.42112  -0.842   0.4004
```

```
## X4:X7        -0.17403    0.44490  -0.391   0.6958
## X5:X6        -4.03578    0.41570  -9.708  < 2e-16 ***
## X5:X7        -0.18418    0.43565  -0.423   0.6726
## X6:X7         2.48691    0.43112   5.768 1.37e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8149 on 521 degrees of freedom
## Multiple R-squared:  0.9838, Adjusted R-squared:  0.9829
## F-statistic:  1129 on 28 and 521 DF,  p-value: < 2.2e-16
```

On voit clairement que les variables $X3$ et $X4$ sont très peu significatives étant avoir une p_valeur>0.05 et très peu influentes (beta petits par rapport aux autres paramètres). On chosisit du coup d'enlever ces variable et refaire le krigeage avec les 5 autres, en ajoutant les interaction significatifs.

```
X_app1=X_app[, c(-3, -4)]
X_test1=X_test[, c(-3, -4)]
```

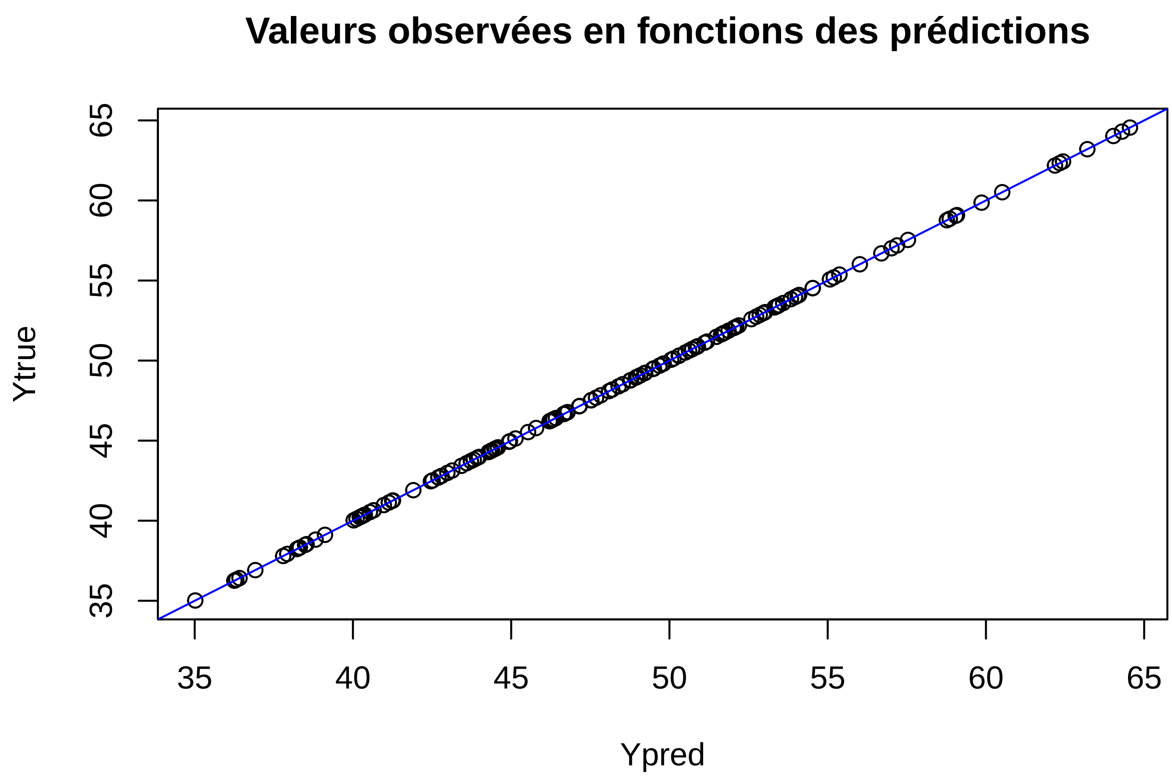
```
famille="matern5_2"
Model <- km(formula = ~1+.+.^2+I(X2^2)+I(X6^2)+I(X5^2)+I(X1^2)-X1:X6-X3:X5-X4:X5-X2:X3-X4:X7, design = 
```

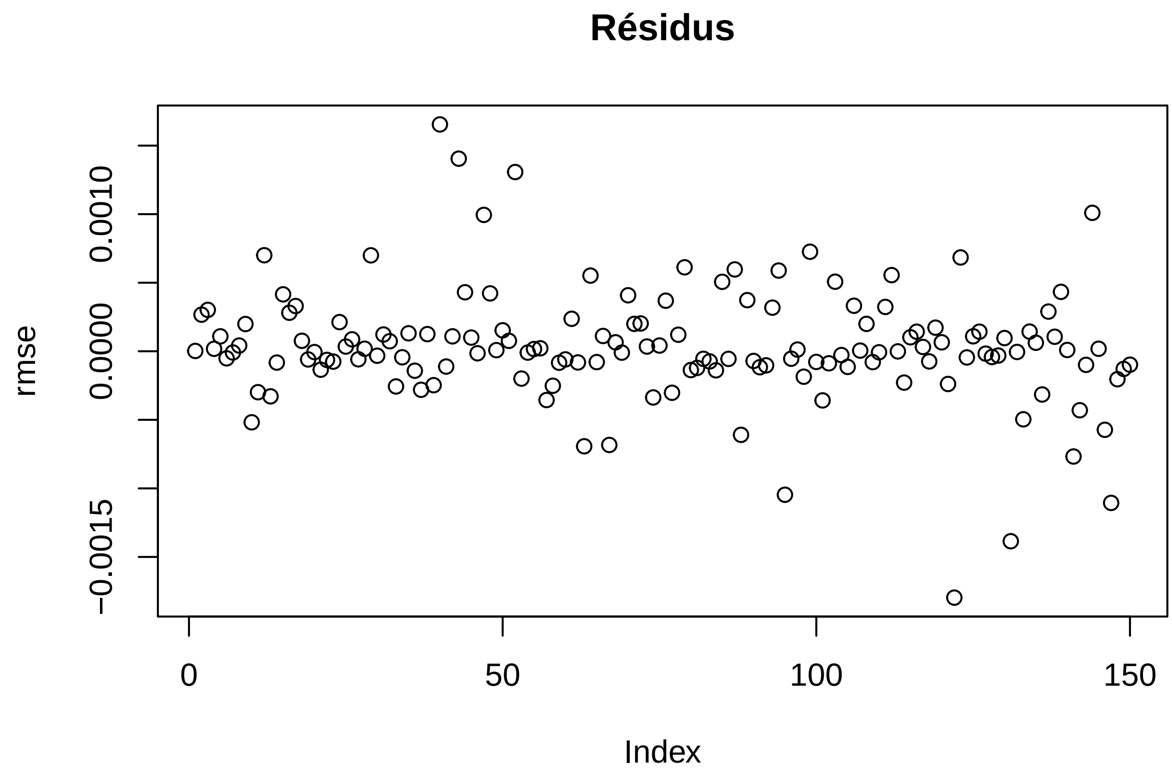
```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=9) to new 11 after EncodeVars() -- should no longer happen!
```

```
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X5 + X6 + X7 + I(X2^2) + I(X6^2) + I(X5^2) + I(X1^2) +
##     X1:X2 + X1:X5 + X1:X7 + X2:X5 + X2:X6 + X2:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.996341 1.992466 1.993762
##   - best initial criterion value(s) :  4.04318e-07
##
## N = 5, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=   4.0432e-07  |proj g|=   3.2064e-06
## At iterate     1  f =    4.043e-07  |proj g|=    3.2062e-06
##
## iterations 1
## function evaluations 2
## segments explored during Cauchy searches 1
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 0
## norm of the final projected gradient 3.20619e-06
## final function value 4.04304e-07
##
```

```
## F = 4.04304e-07
## final  value 0.000000
## converged
```

```r
pred <- predict(object = Model, newdata = X_test1, type="UK" , checkNames=FALSE, se.compute=TRUE)
Mean<-pred $mean
rmse(y_test,Mean, trace=TRUE)
```






```
## [1] 0.0004244927
```

avec Cross Validation

```
Observations<-Observations[sample(nrow(Observations)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(Observations)),breaks=5,labels=FALSE)
lR<-seq(1,5)
#Perform 10 fold cross validation
for(i in 1:5){
  #Segement your data by fold using the which() function
  testIndexes <- which(folds==i,arr.ind=TRUE)
  y_app_cv<-Observations[-testIndexes,8]
  X_app_cv<-Observations[-testIndexes,1:p]
  y_test_cv<-Observations[testIndexes,8]
  X_test_cv<-Observations[testIndexes,1:p]

  X_app1_cv=X_app_cv[, c(-3, -4)]
  X_test1_cv=X_test_cv[, c(-3, -4)]
  famille="matern5_2"
  Model <- km(formula = ~1+.+.^2+I(X2^2)+I(X6^2)+I(X5^2)+I(X1^2)-X1:X6-X3:X5-X4:X5-X2:X3-X4:X7, design =
  pred <- predict(object = Model, newdata = X_test1_cv, type="UK" , checkNames=FALSE, se.compute=TRUE)
  Mean<-pred $mean
```

```
  lR[i] <- rmse(y_test_cv,Mean)
}
```

```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=9) to new 11 after EncodeVars() -- should no longer happen!
```

```
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X5 + X6 + X7 + I(X2^2) + I(X6^2) + I(X5^2) + I(X1^2) +
##     X1:X2 + X1:X5 + X1:X7 + X2:X5 + X2:X6 + X2:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.985485 1.998102 1.996341 1.996244 1.993762
##   - best initial criterion value(s) :  3.372999e-07
##
## N = 5, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=    3.373e-07  |proj g|=   7.2838e-06
## At iterate     1  f =   3.3724e-07  |proj g|=    7.2828e-06
##
## iterations 1
## function evaluations 2
## segments explored during Cauchy searches 1
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 0
## norm of the final projected gradient 7.28283e-06
## final function value 3.37244e-07
##
## F = 3.37244e-07
## final  value 0.000000
## converged
```

```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=9) to new 11 after EncodeVars() -- should no longer happen!
```

```
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X5 + X6 + X7 + I(X2^2) + I(X6^2) + I(X5^2) + I(X1^2) +
##     X1:X2 + X1:X5 + X1:X7 + X2:X5 + X2:X6 + X2:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
```

```
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.996341 1.992466 1.993762
##   - best initial criterion value(s) :  2.867385e-08
##
## N = 5, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=   2.8674e-08  |proj g|=   1.9465e-07
## At iterate     1  f =   2.8673e-08  |proj g|=    1.9466e-07
##
## iterations 1
## function evaluations 2
## segments explored during Cauchy searches 1
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 0
## norm of the final projected gradient 1.9466e-07
## final function value 2.86733e-08
##
## F = 2.86733e-08
## final  value 0.000000
## converged

## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=9) to new 11 after EncodeVars() -- should no longer happen!

##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X5 + X6 + X7 + I(X2^2) + I(X6^2) + I(X5^2) + I(X1^2) +
##     X1:X2 + X1:X5 + X1:X7 + X2:X5 + X2:X6 + X2:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.99367 1.996341 1.996626 1.991921
##   - best initial criterion value(s) :  5.72539e-07
##
## N = 5, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=   5.7254e-07  |proj g|=   8.5069e-06
## At iterate     1  f =   5.7247e-07  |proj g|=    8.5053e-06
##
## iterations 1
## function evaluations 2
## segments explored during Cauchy searches 1
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 0
## norm of the final projected gradient 8.50535e-06
## final function value 5.72469e-07
```

```
##
## F = 5.72469e-07
## final  value 0.000001
## converged

## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=9) to new 11 after EncodeVars() -- should no longer happen!


##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X5 + X6 + X7 + I(X2^2) + I(X6^2) + I(X5^2) + I(X1^2) +
##     X1:X2 + X1:X5 + X1:X7 + X2:X5 + X2:X6 + X2:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.989814 1.980276 1.996016 1.996626 1.992937
##   - best initial criterion value(s) :  4.094824e-08
##
## N = 5, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=   4.0948e-08  |proj g|=   7.1621e-07
## At iterate     1  f =   4.0947e-08  |proj g|=    7.1619e-07
##
## iterations 1
## function evaluations 2
## segments explored during Cauchy searches 1
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 0
## norm of the final projected gradient 7.16188e-07
## final function value 4.09474e-08
##
## F = 4.09474e-08
## final  value 0.000000
## converged

## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=9) to new 11 after EncodeVars() -- should no longer happen!


##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X5 + X6 + X7 + I(X2^2) + I(X6^2) + I(X5^2) + I(X1^2) +
##     X1:X2 + X1:X5 + X1:X7 + X2:X5 + X2:X6 + X2:X7 + X5:X6 + X5:X7 +
##     X6:X7
```

```
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.993739 1.996626 1.993762
##   - best initial criterion value(s) :  1.951791e-07
##
## N = 5, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=   1.9518e-07  |proj g|=    2.859e-06
## At iterate     1  f =   1.9517e-07  |proj g|=    2.8588e-06
##
## iterations 1
## function evaluations 2
## segments explored during Cauchy searches 1
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 0
## norm of the final projected gradient 2.85875e-06
## final function value 1.95171e-07
##
## F = 1.95171e-07
## final  value 0.000000
## converged
```

```
print(mean(lR))
```

```
## [1] 0.0004245099
```

9.Choix de la méthode finale et prédictions

Nous retenons la dernière méthode qui donne un RMSE de l'ordre de 10^{-4} même avec crossValidation.

Pour la prédiction finale, la base d'apprentissage contiendra toute la base de données avec ces 550 observations portant que la base de test contiendra les 150 individus dont on doit prédir leur Y .

```
set.seed(12345)
Observations = read.csv("defi_observations.csv", header = TRUE)
p<-7
n<-550
X_train=Observations[,1:p]
Y_train=Observations[,(p+1)]
X_test2=read.csv("defi_apredire.csv",header=TRUE)
X_t=X_test2[,]
X_train1=X_train[, c(-3, -4)]
X_test2=X_test2[, c(-3, -4)]
famille="matern5_2"
ModelF <- km(formula = ~1+.+.^2+I(X2^2)+I(X6^2)+I(X5^2)+I(X1^2)-X1:X6-X3:X5-X4:X5-X2:X3-X4:X7, design =
```

```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=9) to new 11 after EncodeVars() -- should no longer happen!
```

```
##
## optimisation start
## ------------------
## * estimation method   : LOO
## * optimisation method : BFGS
## * analytical gradient : used
## * trend model : ~X1 + X2 + X5 + X6 + X7 + I(X2^2) + I(X6^2) + I(X5^2) + I(X1^2) +
##     X1:X2 + X1:X5 + X1:X7 + X2:X5 + X2:X6 + X2:X7 + X5:X6 + X5:X7 +
##     X6:X7
## * covariance model :
##   - type :  gauss
##   - nugget : NO
##   - parameters lower bounds :  1e-10 1e-10 1e-10 1e-10 1e-10
##   - parameters upper bounds :  1.99172 1.998102 1.996341 1.996626 1.993762
##   - best initial criterion value(s) :  5.540233e-08
##
## N = 5, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=   5.5402e-08  |proj g|=   3.6147e-07
## At iterate     1  f =   5.5402e-08  |proj g|=    3.6147e-07
##
## iterations 1
## function evaluations 2
## segments explored during Cauchy searches 1
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 0
## norm of the final projected gradient 3.61469e-07
## final function value 5.54016e-08
##
## F = 5.54016e-08
## final  value 0.000000
## converged
```

```r
predF <- predict(object = ModelF, newdata = X_test2, type="UK" , checkNames=FALSE, se.compute=TRUE)
bestThetaF = coef(ModelF, "range")
sd2F = coef(ModelF, "sd2")
# moyenne krigeage
pred_finale<- predF$mean
v_Dice <- (predF$sd)^2
#géneration du la base de prédiction
n_test=150
Predfinale=matrix(0, nrow=n_test, ncol=8)
Predfinale[,1:7]=as.matrix(X_t)
Predfinale[,8]= as.matrix(pred_finale)

Predfinale=as.data.frame(Predfinale)
colnames(Predfinale)=c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "Y")

write.csv(Predfinale,"DefiGroupeSROUR.csv", row.names = FALSE)

#on vérifie que c'est bien lisible
LectureDeMonFichier = read.csv("DefiGroupeSROUR.csv", header = TRUE)
message(ncol(LectureDeMonFichier) == 8, ": bon nombre de colonnes")
```

```
## TRUE: bon nombre de colonnes
```

```r
message(nrow(LectureDeMonFichier) == 150, ": bon nombre de lignes")
```

```
## TRUE: bon nombre de lignes
```

```r
print(LectureDeMonFichier)
```

```
##              X1          X2          X3          X4          X5          X6
## 1   0.411601309 0.894667082 0.763477052 0.529624839 0.766795721 0.794756301
## 2   0.703325983 0.276269066 0.639110274 0.755133492 0.525654156 0.207658866
## 3   0.428880938 0.856445250 0.494293169 0.913113287 0.516348806 0.320523331
## 4   0.020926437 0.001307846 0.795911017 0.142316077 0.337128106 0.258195351
## 5   0.959199439 0.814550982 0.441762670 0.680214769 0.606672103 0.646589261
## 6   0.817537927 0.913834011 0.982690988 0.636652859 0.317443681 0.666416287
## 7   0.622883519 0.217458599 0.474901977 0.594381178 0.150970586 0.772329058
## 8   0.134338109 0.515946427 0.294088058 0.965501934 0.510144564 0.628535446
## 9   0.806086040 0.584082038 0.362232656 0.276508078 0.896970342 0.953462048
## 10  0.692783218 0.192636826 0.034217837 0.710073358 0.043380297 0.470089526
## 11  0.596297865 0.683452528 0.488861019 0.900231653 0.444953818 0.487812820
## 12  0.004953830 0.324407198 0.929781985 0.450156044 0.832097645 0.613418488
## 13  0.709030894 0.664104701 0.897647543 0.509547230 0.166963284 0.428878891
## 14  0.519949214 0.144046470 0.019021030 0.121697994 0.202708565 0.374583912
## 15  0.661595809 0.088892540 0.411913071 0.719305859 0.568923868 0.117803665
## 16  0.728043630 0.516256936 0.614136651 0.058125504 0.978104395 0.385637579
## 17  0.211988267 0.369285370 0.958712704 0.508768995 0.482220243 0.222000071
## 18  0.393297630 0.859088731 0.713796089 0.006810328 0.836232611 0.347318162
## 19  0.099673975 0.902796505 0.364672610 0.932373696 0.214454636 0.072194615
## 20  0.983029743 0.927529822 0.004050002 0.033721108 0.185642668 0.550041982
## 21  0.748703385 0.417591268 0.353129793 0.047397736 0.216878215 0.253324032
## 22  0.299008124 0.540319073 0.474118844 0.268275480 0.556943155 0.257117086
## 23  0.348851409 0.767716037 0.923328654 0.329868594 0.344108673 0.387948346
## 24  0.998724761 0.084455822 0.569502471 0.117632811 0.966552276 0.127533981
## 25  0.469983124 0.501150209 0.073783141 0.643466994 0.023430980 0.661005916
## 26  0.404097677 0.980562953 0.922777514 0.504752121 0.437861013 0.275021947
## 27  0.350075379 0.160498000 0.049661702 0.413785366 0.675770480 0.148532629
## 28  0.288298102 0.610455476 0.792575611 0.499867177 0.553862201 0.113910221
## 29  0.075917175 0.832280834 0.852575900 0.268830310 0.486301845 0.509698773
## 30  0.669719144 0.615721949 0.366937605 0.827186107 0.749152932 0.508490983
## 31  0.719206181 0.762352945 0.166011216 0.120465821 0.952046701 0.282771951
## 32  0.298756389 0.995657024 0.776982953 0.486629525 0.467373664 0.731970326
## 33  0.127591353 0.546551230 0.321862963 0.187292993 0.108505155 0.556272509
## 34  0.200155531 0.208113890 0.696113976 0.810732298 0.761631739 0.325717789
## 35  0.393724889 0.584503924 0.568296643 0.615723163 0.504755073 0.034914632
## 36  0.833994708 0.797475173 0.705013200 0.619034624 0.362397965 0.064102058
## 37  0.321804869 0.916930388 0.041939715 0.893895840 0.947935158 0.659015336
## 38  0.672155000 0.219139380 0.262731934 0.857952049 0.522698834 0.658547460
## 39  0.200697999 0.527240968 0.467465928 0.576363802 0.050901949 0.228529654
## 40  0.216361019 0.266782299 0.896816890 0.640457374 0.718943775 0.341928699
## 41  0.952040233 0.692098316 0.197704336 0.901038737 0.351944243 0.293261993
## 42  0.538932080 0.265794639 0.188014876 0.416102835 0.309611155 0.770165823
## 43  0.796534629 0.934350706 0.937764627 0.140229423 0.473886690 0.217773279
## 44  0.915987873 0.295361835 0.081005520 0.288156422 0.588895725 0.641560783
```

```
## 45  0.444384775 0.582337351 0.762721516 0.444000989 0.581069558 0.758234325
## 46  0.215349552 0.949196982 0.328102039 0.332473774 0.019793679 0.763477390
## 47  0.324340681 0.871507696 0.139205269 0.926222243 0.700513041 0.490796224
## 48  0.828923394 0.769049306 0.807887731 0.872223770 0.890707256 0.759304691
## 49  0.645453211 0.302892269 0.616916724 0.194359168 0.773146078 0.020280042
## 50  0.217693194 0.155126530 0.387988219 0.220698333 0.162320791 0.223291317
## 51  0.352499263 0.267157064 0.732074833 0.853316235 0.738659525 0.949347803
## 52  0.201549316 0.933179925 0.150537773 0.989385477 0.642056502 0.386329621
## 53  0.900130307 0.895944321 0.122955907 0.471381604 0.044322344 0.798336108
## 54  0.754018555 0.622440270 0.230753376 0.383924812 0.358087872 0.627455242
## 55  0.821094877 0.011296893 0.805353787 0.546465619 0.815389625 0.435371363
## 56  0.419923200 0.793158557 0.587276935 0.314269752 0.439154642 0.276371345
## 57  0.838066478 0.619400013 0.052954718 0.835929819 0.897834735 0.738238293
## 58  0.050065871 0.814664605 0.355663242 0.770406658 0.060187817 0.611289444
## 59  0.376027734 0.884161202 0.240188154 0.151143362 0.696478248 0.872871456
## 60  0.074849217 0.872033441 0.098813198 0.242196373 0.320114102 0.343196302
## 61  0.406894022 0.470463060 0.936518884 0.986065640 0.367301789 0.079464334
## 62  0.875515927 0.011911446 0.818078666 0.330847368 0.206538340 0.755865441
## 63  0.235765846 0.323918297 0.471121316 0.073751022 0.768206000 0.549039881
## 64  0.415154933 0.490397227 0.812381326 0.939839228 0.419174814 0.489480545
## 65  0.007488793 0.495986141 0.707090972 0.675537389 0.311297600 0.054234880
## 66  0.156361496 0.106448309 0.748647860 0.351937923 0.453747190 0.564337478
## 67  0.330498126 0.453635785 0.907376332 0.863185269 0.424980849 0.501806643
## 68  0.151155049 0.761679906 0.614067774 0.933238722 0.668270083 0.236602334
## 69  0.797166580 0.300089135 0.245913304 0.268978176 0.706833128 0.357671393
## 70  0.399392641 0.459283856 0.442235041 0.709771838 0.043106016 0.761138716
## 71  0.459711072 0.997083778 0.722481209 0.425315045 0.241514476 0.609681011
## 72  0.349299230 0.544333821 0.612442011 0.632495366 0.945552375 0.321944568
## 73  0.428597611 0.872149355 0.175015192 0.723930435 0.610998829 0.084812620
## 74  0.293405992 0.139924661 0.627788353 0.720704145 0.639476709 0.241893880
## 75  0.359300866 0.457686851 0.357199034 0.557609136 0.646378323 0.025885267
## 76  0.311386749 0.284668697 0.177228781 0.409404228 0.877086202 0.735927408
## 77  0.252015685 0.978304417 0.057784788 0.193048678 0.125976090 0.950646088
## 78  0.525319364 0.462875522 0.622416739 0.236701944 0.899182987 0.888908192
## 79  0.358738265 0.869477100 0.531538949 0.815977871 0.917699199 0.887427581
## 80  0.377339387 0.282831721 0.206176916 0.534229206 0.462057650 0.927899652
## 81  0.594768221 0.190651374 0.342134282 0.936440678 0.316156392 0.116763903
## 82  0.612176829 0.067430736 0.442619957 0.827392359 0.195828937 0.828395548
## 83  0.243596822 0.092823209 0.344732147 0.245568891 0.293250316 0.388979499
## 84  0.029365144 0.226438021 0.506534631 0.069867018 0.523445226 0.814469312
## 85  0.792216022 0.402486092 0.280475911 0.501622726 0.597173909 0.556511827
## 86  0.095025099 0.342687678 0.960758919 0.620007927 0.044998999 0.008262263
## 87  0.334571643 0.037260832 0.221875334 0.836745579 0.505877984 0.429627986
## 88  0.253485191 0.733076191 0.460245023 0.054184451 0.293739498 0.849145855
## 89  0.340011633 0.985516000 0.141409034 0.725042785 0.627990619 0.834234686
## 90  0.840303617 0.351478350 0.008984613 0.317251203 0.470077059 0.884161928
## 91  0.521768841 0.486716804 0.439468782 0.263098839 0.652342699 0.700103650
## 92  0.310276956 0.071434639 0.040094854 0.796754755 0.304198997 0.032179042
## 93  0.356249658 0.709890524 0.834837737 0.352185391 0.845022202 0.092124465
## 94  0.111989676 0.465512232 0.791822406 0.093517106 0.922431013 0.985893189
## 95  0.824974979 0.111050329 0.108511348 0.079605064 0.324378181 0.307966884
## 96  0.851645255 0.866854068 0.561545613 0.664664893 0.621045236 0.550300236
## 97  0.175277219 0.644910198 0.687200384 0.957521652 0.947452510 0.273888384
## 98  0.983856027 0.559827174 0.739743443 0.508827025 0.859011387 0.613070698
```

```
## 99  0.656474818 0.900956529 0.208930149 0.760159161 0.061162533 0.986950015
## 100 0.345437306 0.187540622 0.377957241 0.996352371 0.495129725 0.056477477
## 101 0.180892660 0.114693172 0.319574233 0.099738075 0.813570677 0.508146232
## 102 0.188789811 0.101884496 0.769842853 0.464372990 0.279278551 0.458171810
## 103 0.727003539 0.144478952 0.823335996 0.532865382 0.819200251 0.673349767
## 104 0.030625277 0.618447193 0.570446773 0.552958159 0.585163967 0.960841252
## 105 0.584666816 0.590475518 0.733259299 0.878279839 0.471574047 0.673762366
## 106 0.750016854 0.551254706 0.178138885 0.717924527 0.758644631 0.585893066
## 107 0.083689540 0.796656078 0.363375221 0.310102579 0.872793498 0.120634135
## 108 0.734986994 0.524868857 0.483972848 0.905208289 0.921692217 0.073551780
## 109 0.212006292 0.347062289 0.159222895 0.283855372 0.304394913 0.358610681
## 110 0.392843458 0.159127218 0.777314183 0.066351679 0.347790204 0.180198543
## 111 0.697448426 0.278361964 0.167477098 0.122856511 0.149729693 0.617411971
## 112 0.479736079 0.321498750 0.989685236 0.945585349 0.653048365 0.537193977
## 113 0.514348536 0.922533174 0.940571087 0.126610678 0.895860275 0.213503022
## 114 0.960557266 0.714952920 0.828544016 0.553830377 0.116554197 0.337247520
## 115 0.352266295 0.313754705 0.230282247 0.281456232 0.466853141 0.970453946
## 116 0.437291781 0.376925124 0.452202629 0.059405061 0.272008021 0.719498862
## 117 0.282660231 0.038003895 0.713472493 0.850966723 0.486467096 0.721796915
## 118 0.740431397 0.821052619 0.480577405 0.599276480 0.195937720 0.352858731
## 119 0.770630880 0.663192800 0.949000386 0.195386423 0.958556244 0.876093820
## 120 0.053352330 0.600032982 0.974563877 0.806417854 0.145987527 0.579531576
## 121 0.129623877 0.926268864 0.082445351 0.935685447 0.201750571 0.190953559
## 122 0.449406118 0.430983230 0.992267437 0.334365345 0.128645960 0.435300704
## 123 0.219031853 0.674302758 0.939260717 0.689621578 0.602793836 0.602049983
## 124 0.460901272 0.357186061 0.428504478 0.241656193 0.644342664 0.238679780
## 125 0.503620226 0.677664535 0.581887110 0.324728117 0.239428201 0.523302713
## 126 0.028889730 0.817181754 0.762753590 0.730978261 0.006051353 0.039537765
## 127 0.108500021 0.911233520 0.475634841 0.174728347 0.702586798 0.794616421
## 128 0.265720366 0.990976143 0.363701463 0.266853303 0.577447380 0.706991709
## 129 0.380933684 0.673509677 0.009748787 0.181240209 0.198078084 0.676059741
## 130 0.608219477 0.787894590 0.826401187 0.459129313 0.331036919 0.761493001
## 131 0.375325718 0.747063217 0.540714253 0.152743409 0.392425481 0.863598506
## 132 0.959654021 0.592727461 0.626906773 0.603482881 0.095554127 0.535736937
## 133 0.052377697 0.981500129 0.334919410 0.748107769 0.814800017 0.564427272
## 134 0.030180734 0.407653346 0.773352027 0.617371321 0.916221549 0.023813468
## 135 0.127673000 0.655685371 0.286461565 0.688279476 0.600582285 0.429246492
## 136 0.145955378 0.165828887 0.637974613 0.990504819 0.425909039 0.666116319
## 137 0.403855272 0.186890728 0.457603682 0.662660268 0.854974583 0.977181518
## 138 0.403433403 0.381453553 0.480223866 0.190297028 0.717524921 0.856134772
## 139 0.938765343 0.637786380 0.432273642 0.381905765 0.464992600 0.881687334
## 140 0.414454876 0.906589429 0.916113429 0.772954703 0.769287406 0.734101109
## 141 0.620222901 0.119342077 0.660894399 0.134427676 0.268157196 0.996555114
## 142 0.682047944 0.695446456 0.616123410 0.807857533 0.335627021 0.915991990
## 143 0.625679979 0.273650197 0.687235755 0.042358467 0.646001289 0.292843530
## 144 0.798758465 0.602881383 0.060462625 0.160059185 0.744572307 0.475678028
## 145 0.234733217 0.758136530 0.121801331 0.411818012 0.569992813 0.542454878
## 146 0.911428126 0.451117955 0.754199036 0.988703375 0.658832161 0.412058114
## 147 0.808708142 0.504562717 0.306917947 0.942200247 0.019192549 0.353805312
## 148 0.562738487 0.655352012 0.356378503 0.371152896 0.696540397 0.042635830
## 149 0.664369058 0.892867247 0.061680939 0.585934291 0.982188257 0.409196397
## 150 0.409642222 0.407943380 0.341650677 0.447098853 0.530168588 0.551152195
##             X7        Y
## 1   0.11577590 38.66005
```

```
## 2   0.66234750 58.16141
## 3   0.57879106 45.03721
## 4   0.17095286 52.51382
## 5   0.41519265 44.36287
## 6   0.09933596 41.29761
## 7   0.54596801 49.96233
## 8   0.88385268 43.42668
## 9   0.93069042 47.02313
## 10  0.96319632 53.79947
## 11  0.03758351 46.00984
## 12  0.83090470 45.57022
## 13  0.09339836 47.11483
## 14  0.94533267 54.20869
## 15  0.96283919 62.49416
## 16  0.89392489 53.64449
## 17  0.25213042 51.02115
## 18  0.13776447 45.13338
## 19  0.19086286 45.19975
## 20  0.32002985 42.19775
## 21  0.60792009 54.26093
## 22  0.74718810 49.42830
## 23  0.86298274 44.04010
## 24  0.54829584 66.07361
## 25  0.44595458 44.49798
## 26  0.70685767 43.59762
## 27  0.81509407 57.47677
## 28  0.24594282 50.98754
## 29  0.26572907 39.57233
## 30  0.03404122 48.33808
## 31  0.96269153 51.57428
## 32  0.83246758 37.26857
## 33  0.32769406 41.88673
## 34  0.55819380 52.40349
## 35  0.71398636 53.84425
## 36  0.68636223 52.91007
## 37  0.08950088 39.40309
## 38  0.31357192 51.96135
## 39  0.15578035 46.93853
## 40  0.14604950 51.07464
## 41  0.44493176 50.31446
## 42  0.19689627 48.11931
## 43  0.91512578 48.32143
## 44  0.26969084 52.44649
## 45  0.45968406 43.72417
## 46  0.37142596 35.39411
## 47  0.89806482 42.32791
## 48  0.56683534 44.83979
## 49  0.02065082 62.25896
## 50  0.58567697 52.56871
## 51  0.74861627 46.58516
## 52  0.22686702 41.42486
## 53  0.48767558 40.53352
## 54  0.42744969 46.56455
## 55  0.56623814 60.09184
```

```
## 56  0.06465026 45.97026
## 57  0.74634446 47.67898
## 58  0.81258038 37.80222
## 59  0.10199600 37.70360
## 60  0.03316296 40.73052
## 61  0.67992482 53.92542
## 62  0.08054065 54.37186
## 63  0.01214751 47.31809
## 64  0.05015488 46.60367
## 65  0.13998813 50.25261
## 66  0.09719800 48.33751
## 67  0.13802561 46.15404
## 68  0.63962555 46.26331
## 69  0.53523804 56.44191
## 70  0.08535563 43.04672
## 71  0.85194697 39.02142
## 72  0.62329152 50.17066
## 73  0.13442084 49.59309
## 74  0.47667247 55.07204
## 75  0.20173830 56.05779
## 76  0.36134976 47.05264
## 77  0.68234267 35.19699
## 78  0.41493673 45.96887
## 79  0.23686091 38.31426
## 80  0.53394309 45.86236
## 81  0.36529858 59.02427
## 82  0.54781885 51.74661
## 83  0.86354651 51.73632
## 84  0.77466345 44.45851
## 85  0.93341895 52.12300
## 86  0.56469298 52.71177
## 87  0.83751410 53.54690
## 88  0.86504355 39.40820
## 89  0.53458815 36.88388
## 90  0.47831805 49.44794
## 91  0.02598975 46.02896
## 92  0.23167587 58.80366
## 93  0.35449482 52.09836
## 94  0.08021525 40.79806
## 95  0.94574644 59.01366
## 96  0.43297525 44.63416
## 97  0.50520116 48.29944
## 98  0.91332724 50.26950
## 99  0.71568273 39.63466
## 100 0.34474249 58.06387
## 101 0.49346614 50.72660
## 102 0.26949329 49.65083
## 103 0.77995690 54.77981
## 104 0.14885399 37.89241
## 105 0.91052240 46.25650
## 106 0.84829555 49.73001
## 107 0.19942658 48.43512
## 108 0.04109790 59.13701
## 109 0.39921119 48.35659
```

```
## 110 0.09948345 55.67856
## 111 0.32125908 50.70886
## 112 0.34781956 50.09750
## 113 0.13142941 48.24864
## 114 0.71889348 48.52604
## 115 0.13452737 44.25605
## 116 0.80067460 46.49730
## 117 0.19626335 48.89815
## 118 0.11353651 46.12655
## 119 0.51336123 45.46669
## 120 0.96943428 41.20443
## 121 0.57718280 42.84184
## 122 0.74509982 48.17381
## 123 0.99009087 42.67785
## 124 0.55584110 54.19356
## 125 0.10159302 44.04984
## 126 0.33121432 45.38496
## 127 0.66522042 37.02125
## 128 0.41295915 36.99391
## 129 0.44614713 41.51140
## 130 0.83658866 42.51375
## 131 0.22699449 39.24583
## 132 0.63479937 47.91322
## 133 0.18842066 37.73991
## 134 0.32520500 55.23198
## 135 0.46667381 43.87499
## 136 0.90243809 47.36415
## 137 0.29263282 47.42058
## 138 0.26267332 45.23301
## 139 0.66887230 45.49882
## 140 0.36594079 39.45705
## 141 0.52459997 50.59474
## 142 0.50829068 43.12810
## 143 0.42532091 56.30844
## 144 0.49597476 50.12972
## 145 0.68435178 41.90672
## 146 0.10985901 53.10203
## 147 0.77974852 51.02643
## 148 0.39128974 55.46133
## 149 0.16658925 46.64353
## 150 0.80510049 48.02363
```