```
rm(list=ls(all=TRUE))
graphics.off()

x11()

library(DiceKriging)
library(DiceView)

set.seed(1234)

########### ETAPE 0 ###############

fx_exo2 <- function(X)
  sin(30*(X-0.9)^4)*cos(2*(X-0.9)) + (X-0.9)/2

N_BT = 100                 ### Nombre de points de la base de test
S_BT = seq(0,1,length.out=N_BT)    ### Simulation d'une premiere base de test
## S_BT : entrees de la base de test
## Y_BT : sortie du simulateur

Y_BT <- fx_exo2(S_BT)   ## Y_BT : sortie du simulateur sur la BT

plot(S_BT,Y_BT,type='p')

########### ETAPE 1 ################
N_BA = 15                ## Nombre de points de la base d'apprentissage (BA)
X <- runif(N_BA) # plan aleatoire
S_BA <- as.matrix(X,nc=1)

Y_BA <- fx_exo2(S_BA)   ## Y_BA : sortie du simulateur sur la BA

points(S_BA,Y_BA,pch=3,col=2)
legend("topright",y=c(0.17,0.33),c('Fonction théorique','Pts de la base d apprentissage'),lty=c(1,0),
       pch=c(-1,3),col=c(1,2),cex=0.8)

########### ETAPE 2 ################

metamodel <- km(formula=~1,design=S_BA,response=Y_BA,covtype="matern5_2") # Creation du metamodele processus gaussien

prediction <- predict.km(metamodel,S_BT,'UK',checkNames=FALSE) # Prédiction du metamodele sur la base de test =>
utilisation de la fx predict.km

x11()
par(mfrow=c(2,1))
plot(S_BT,Y_BT,type='l',main='Estimation de f par métamodèle PG')
points(S_BA,Y_BA,pch=3,col='red')
lines(S_BT,prediction$mean,col='blue')
lines(S_BT,prediction$lower95,col='blue',lty=2)
lines(S_BT,prediction$upper95,col='blue',lty=2)
plot(S_BT,prediction$sd,type='l',main="sqrt(MSE) de l'estimation")

# avec DiceView
par(mfrow=c(2,1))
sectionview(metamodel)
plot(S_BT,prediction$sd,type='l',main="sqrt(MSE) de l'estimation")

#### Calcul du Q2 su BT ####
# Creation de la fonction qui permet de calculer le Q2
Q2 <- function(y,yhat)
  1-mean((y-yhat)^2)/var(y)
Q2_BT <- Q2(Y_BT,prediction$mean) # Calcul du Q2 sur la base de test

#### Affichage des resultats ####
print(paste('Nombre de points de la base d apprentissage :',N_BA))
print(paste('Q2 sur Base de test :',Q2_BT))

########### ETAPE 3 ################

## Planification adaptative : une itération
maxi <- which.max(prediction$sd)  # Recherche du point où le MSE est maximal
points(S_BT[maxi],prediction$sd[maxi],pch=16,col=3)
S_BA2 <- rbind(S_BA,S_BT[maxi])   # rajout du point dans la BA
Y_BA2 <- c(Y_BA,fx_exo2(S_BT[maxi])) # rajout du Y correspondant dans la BA
metamodel2 <- km(formula=~1,design=S_BA2,response=Y_BA2,covtype="matern5_2") # Mise à jour du métamodèle PG
prediction2 <- predict.km(metamodel2,S_BT,'UK',checkNames=FALSE) # Mise à jour des prédictions et du MSE du métamodèle

# Affichage
par(mfrow=c(2,1))
sectionview(metamodel2)
plot(S_BT,prediction2$sd,type='l',main="sqrt(MSE) de l'estimation avec ajout d'un point")
points(S_BT[maxi],prediction2$sd[maxi],pch=16,col=3)

## Planification adaptative : Automatisation du processus
nIte <- 10
```

```
for (i in 1:nIte){
  maxi <- which.max(prediction2$sd)   # Recherche du point où le MSE est maximal
  S_BA2 <- rbind(S_BA2,S_BT[maxi])    # rajout du point dans la BA
  Y_BA2 <- c(Y_BA2,fx_exo2(S_BT[maxi]))
  capture.output(metamodel2 <- km(formula=~1,design=S_BA2,response=Y_BA2,covtype="matern5_2")) # Mise à jour du
métamodèle PG
  prediction2 <- predict.km(metamodel2,S_BT,'UK',checkNames=FALSE) # Mise à jour des prédictions et du MSE du
métamodèle

  # Affichage
  par(mfrow=c(2,1))
  sectionview(metamodel2)
  plot(S_BT,prediction2$sd,type='l',main="sqrt(MSE) de l'estimation avec ajout d'un point")
  points(S_BT[maxi],prediction2$sd[maxi],pch=16,col=3)
  Sys.sleep(2)
}
# Rque : capture.output empeche km d'afficher à chaque fois les résultats de l'optim
```