# Enoncé TP optimisation globale – Part 1 Majeure science des données

**UP 4 : exploitation mathématique de simulateurs numériques 2020-21**

## Rodolphe Le Riche, Charlie Sire

*Connaissances apportées par ce TP :*

1. *Comment tester un optimiseur global*
2. *Optimiseur local versus optimiseur global*
3. *Observer l’effet de la variance (le « pas ») dans un optimiseur stochastique simple*

*GROUP 16*

*WANG Yuteng*

*XU Liwei*

*ZHAO Wenxu*

1. **Comment tester un optimiseur global. Prise en main des programmes en R.**

Cette partie A ne nécessite pas de compte-rendu, c’est une prise en main des programmes. Les parties B et C donnent elles lieu à un petit compte-rendu.

* 1. Télécharger les fichiers mis à disposition sur Campus ( ICM / Majeures / Majeure Sciences des données / Exploitation mathématique de simulateurs numériques / Optimisation Globale )

Les décompresser, et parcourir la présentation des codes ci-dessous, qui est un extrait du fichier readme.txt .

## Ouvrir 3Dplots.R . Ce fichier permet de visualiser des fonctions 2D et des trajectoires d’optimiseurs. Utiliser l’optimiseur local « random\_search» sur une fonction convexe (la fonction "sphere" ou "quadratic") puis sur une fonction multimodale au choix. Observer et comprendre les sorties et les réglages possibles. Changer l’optimiseur en « lbfgs », observer.

* 1. Ouvrir « main4tests.R ». Nous allons maintenant faire des expériences numériques en répétant des optimisations. Choisir comme optimiseur « random\_search », et faire des tests sur la fonction de votre choix en dimension 5. Répéter les expériences avec l’optimiseur « lbfgs ». Utiliser le fichier

« postproc\_tests\_optimizers.R » pour faire des courbes synthétiques qui comparent les deux optimiseurs.

Vous savez maintenant vous servir des programmes.

1. **Optimiseur local versus optimiseur global**
   1. En 2D, visuellement.

Au moyen de 3Dplots.R , comparer l’optimiseur local « lbfgs » à l’optimiseur global

« random\_search » sur la fonction unimodal « quadratic ». Répéter la comparaison sur une fonction multimodale ("ackley", "michalewicz", « schwefel », « rastrigin »). Avec lbfgs, faire varier le point initial. Que se passe-t-il ?

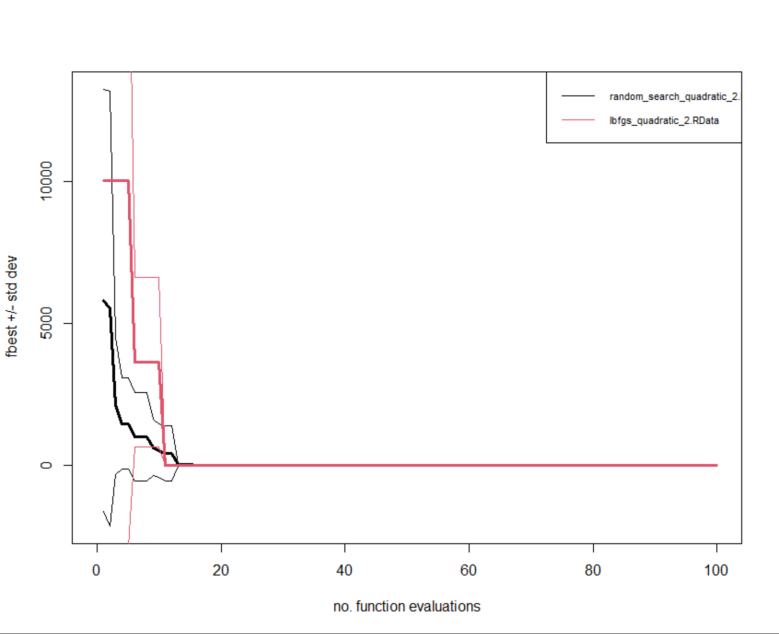
-----------------------------------------------------------------------------------------------------------------------

l’optimiseur lbfgs pour quadratic fonction ：lbfgs\_quadratic\_2.RData

l’optimiseur random\_search pour quadratic fonction: random\_search\_quadratic\_.RData

Dans postproc\_tests\_optimizers.R





random\_search descends plus vite que lbfgs et presque la même profondeur pour le traitement des fonctions **quadratic**

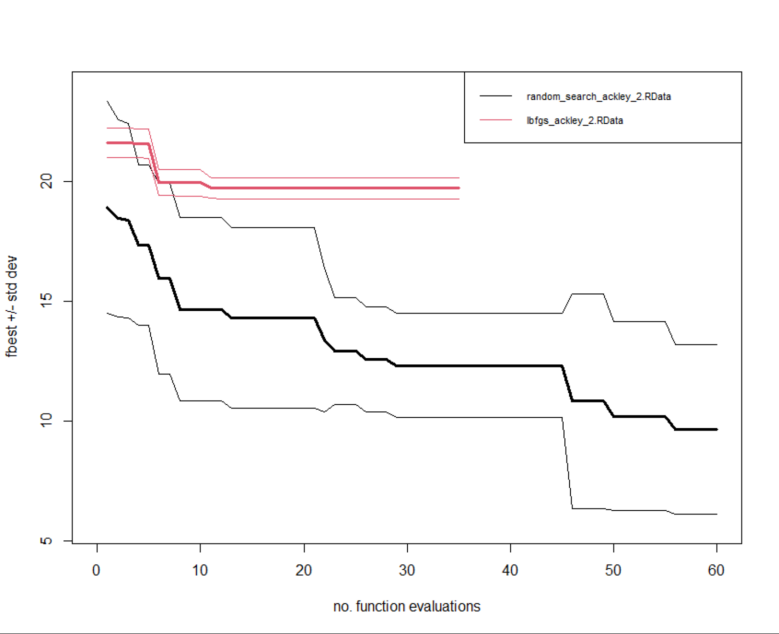
---------------------------------------------------------------------------------------------------------------------------

l’optimiseur lbfgs pour ackley fonction ：lbfgs\_ackley\_2.RData

l’optimiseur random\_search pour ackley fonction: random\_search\_ackley\_2.RData

Dans postproc\_tests\_optimizers.R





random\_search descends plus vite que lbfgs et random\_search a de meilleures performances (profondeur)pour le traitement des fonctions **ackley**

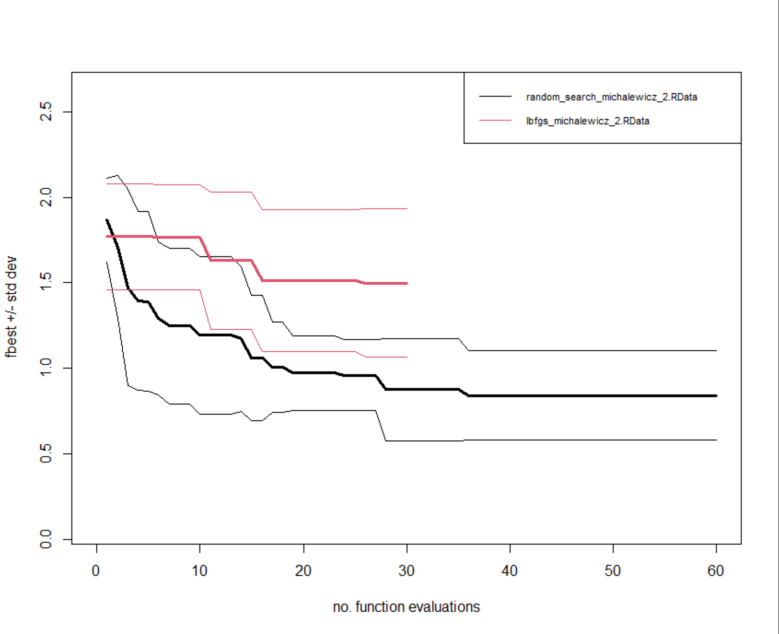
---------------------------------------------------------------------------------------------------------------------------

l’optimiseur lbfgs pour michalewicz fonction ：lbfgs\_michalewicz\_2.RData

random\_search pour michalewicz fonction: random\_search\_michalewicz\_2.RData

Dans postproc\_tests\_optimizers.R





random\_search descends plus vite que lbfgs et random\_search a de meilleures performances (profondeur)pour le traitement des fonctions michalewicz

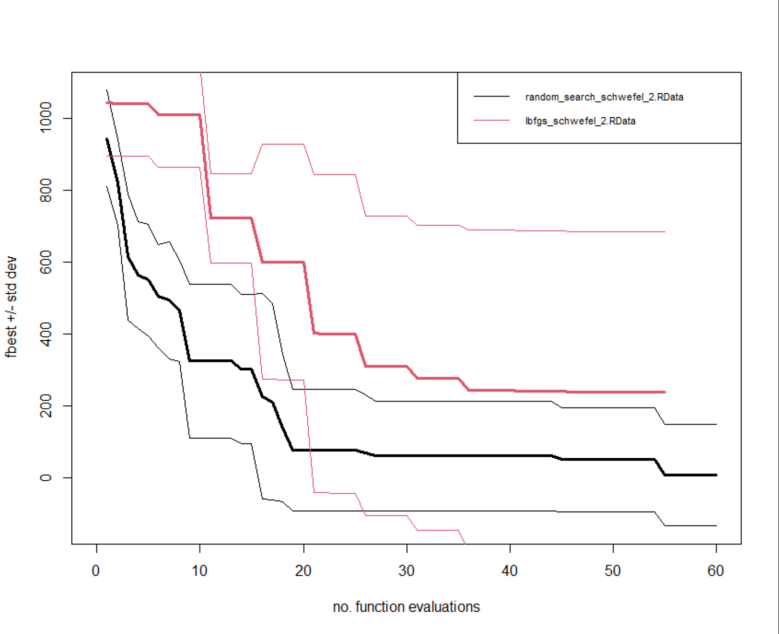
---------------------------------------------------------------------------------------------------------------------------

l’optimiseur lbfgs pour schwefel fonction ：lbfgs\_schwefel \_2.RData

random\_search pour schwefel fonction: random\_search\_schwefel \_2.RData

Dans postproc\_tests\_optimizers.R





random\_search descends plus vite que lbfgs et pluspart de random\_search a de meilleures performances (profondeur)pour le traitement des fonctions **schwefel**

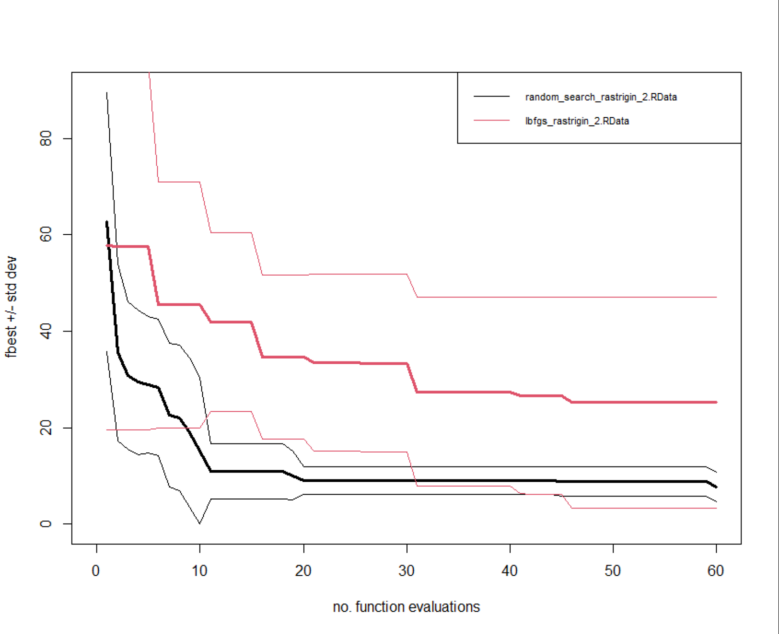
---------------------------------------------------------------------------------------------------------------------------

l’optimiseur lbfgs pour rastrigin fonction ：lbfgs\_rastrigin\_2.RData

random\_search pour rastrigin fonction: random\_search\_rastrigin\_2.RData

Dans postproc\_tests\_optimizers.R





random\_search descends plus vite que lbfgs et pluspart de random\_search a de meilleures performances (profondeur)pour le traitement des fonctions **rastrigin**

---------------------------------------------------------------------------------------------------------------------------

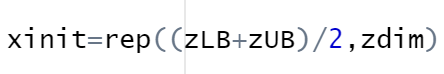
Avec 2D lbfgs, faire varier le point initial

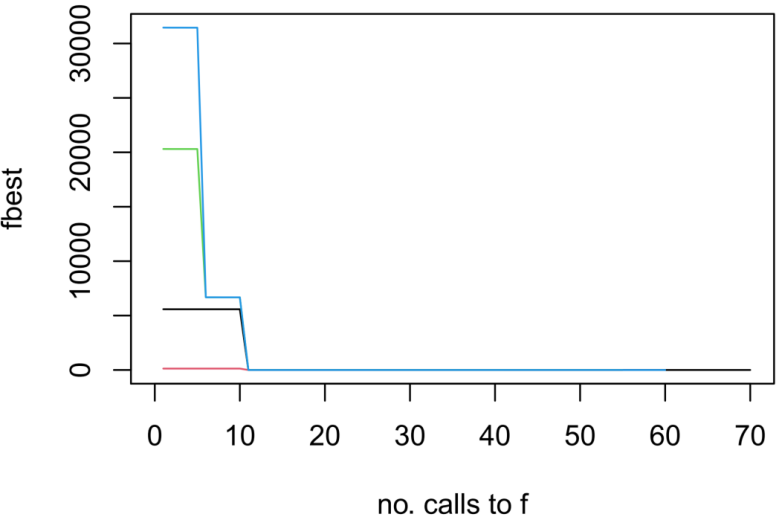
Au debut:

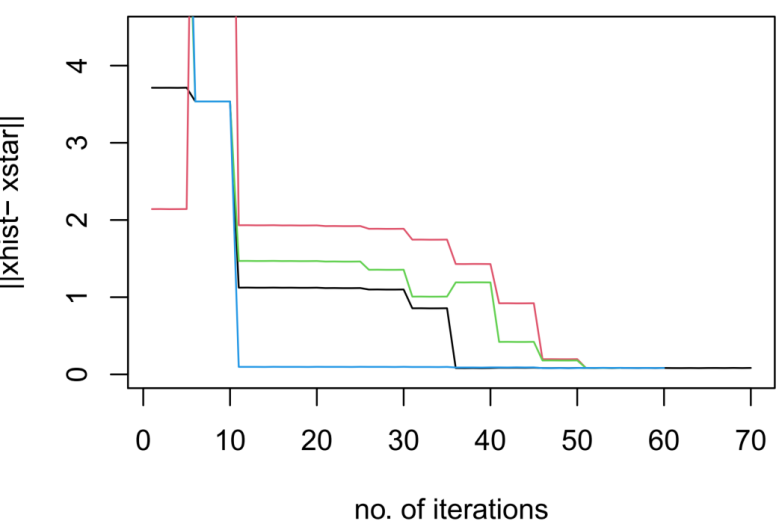
zLB <- -5 # lower bound on parameters

zUB <- 5 # upper bound on parameters

no\_test <<- 4

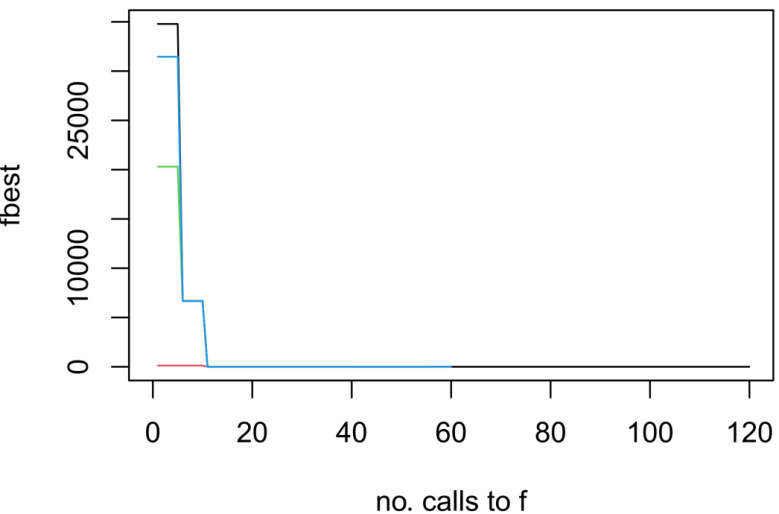


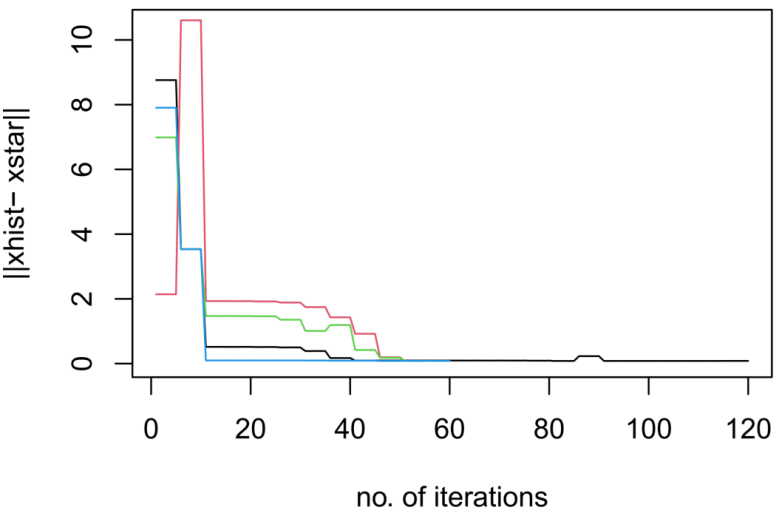




on change:







* 1. En 5D, avec répétition des tests

Choisir une fonction multimodale en dimension 5 et comparer les performances de l’optimiseur local « lbfgs » aux optimiseurs globaux random\_search et cmaes. Faire varier les points initiaux pour lbfgs et cmaes.

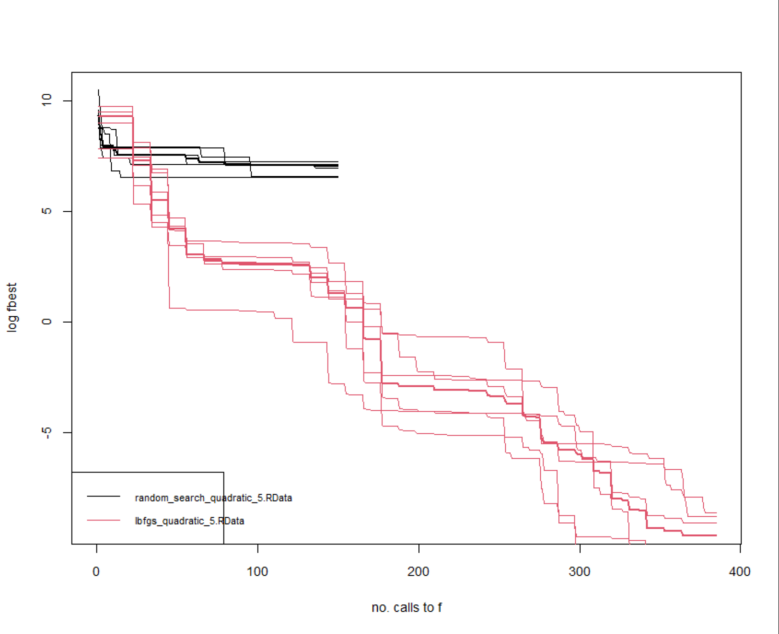
-----------------------------------------------------------------------------------------------------------------------

l’optimiseur lbfgs pour quadratic fonction ：lbfgs\_quadratic\_5.RData

l’optimiseur random\_search pour quadratic fonction: random\_search\_quadratic\_5.RData

Dans postproc\_tests\_optimizers.R





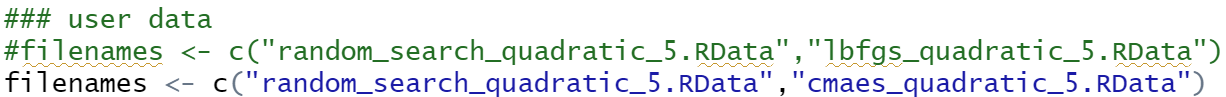
random\_search descends plus vite que lbfgs mais lbfgs a de meilleures performances (profondeur)pour le traitement des fonctions quadratiques

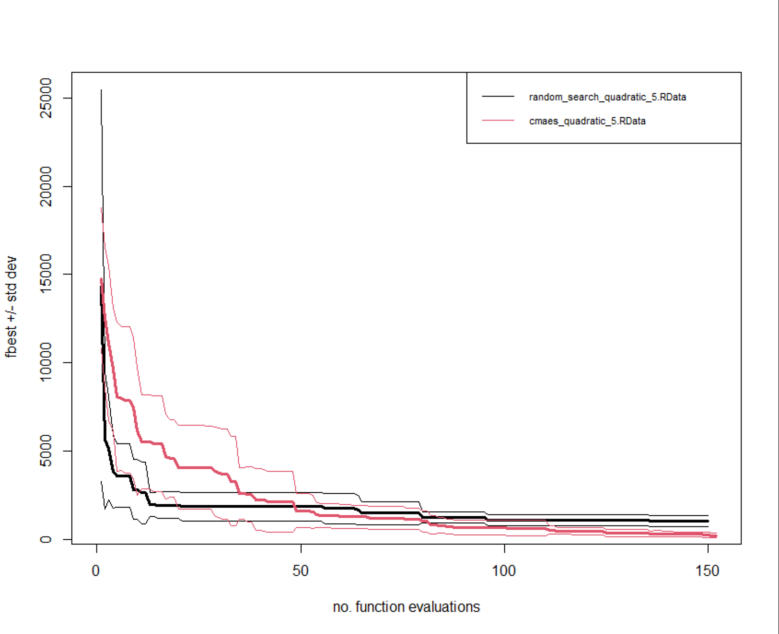
---------------------------------------------------------------------------------------------------------------------------

l’optimiseur lbfgs pour quadratic fonction ：lbfgs\_quadratic\_5.RData

l’optimiseur cmaes pour quadratic fonction: cmaes\_quadratic\_5.RData

Dans postproc\_tests\_optimizers.R





lbfgs descends plus vite que cmaes et presque la même profondeur.

---------------------------------------------------------------------------------------------------------------------------

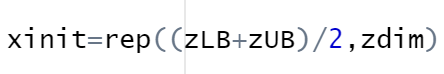
Avec 5D lbfgs, faire varier le point initial

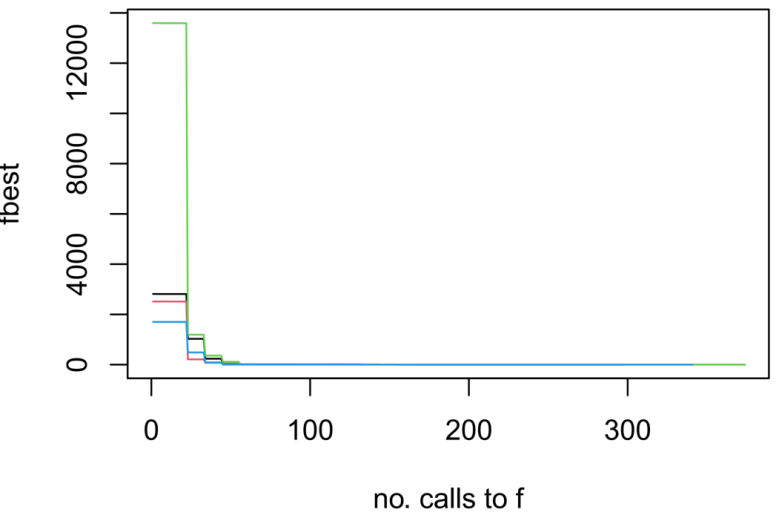
Au debut:

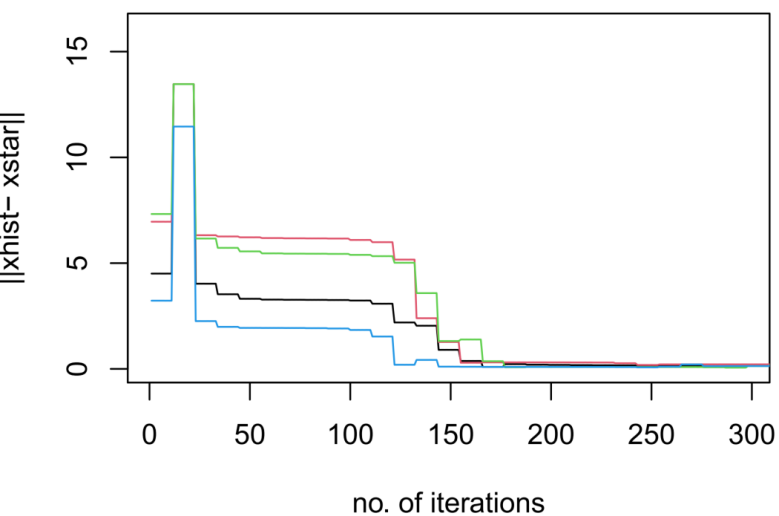
zLB <- -5 # lower bound on parameters

zUB <- 5 # upper bound on parameters

no\_test <<- 4

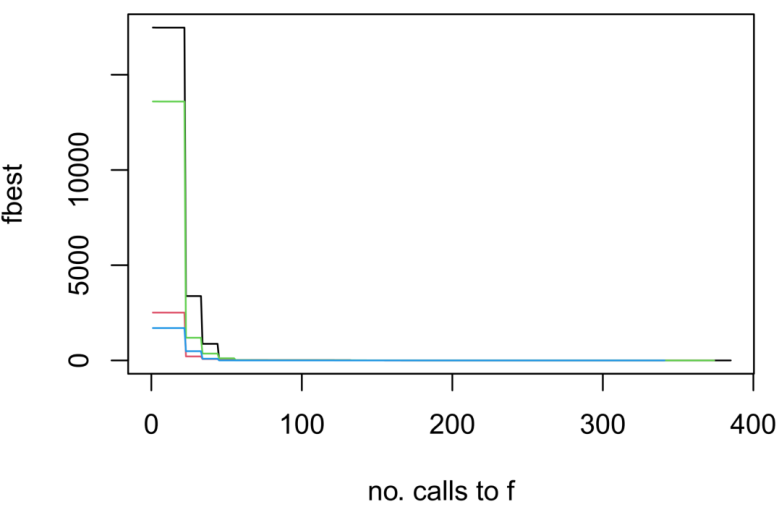


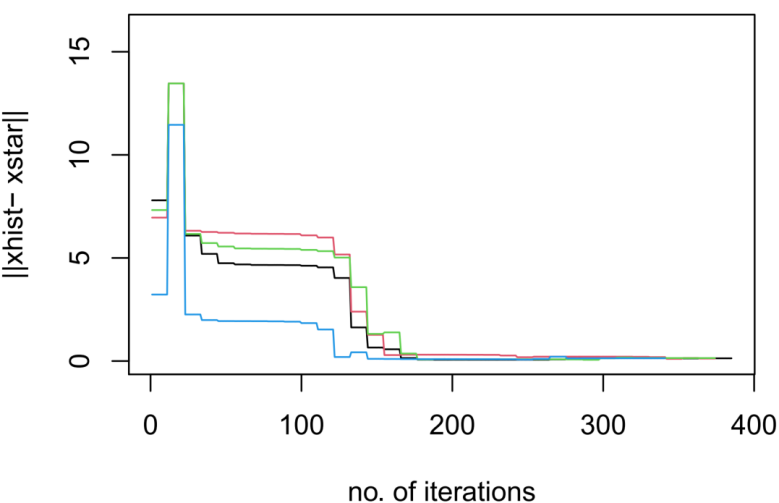




On change







---------------------------------------------------------------------------------------------------------------------------

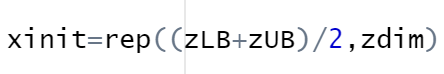
Avec 5D cmaes, faire varier le point initial

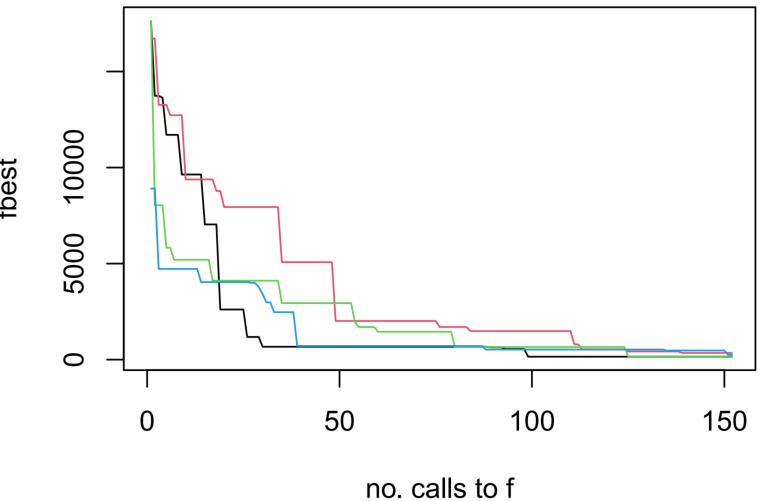
Au debut:

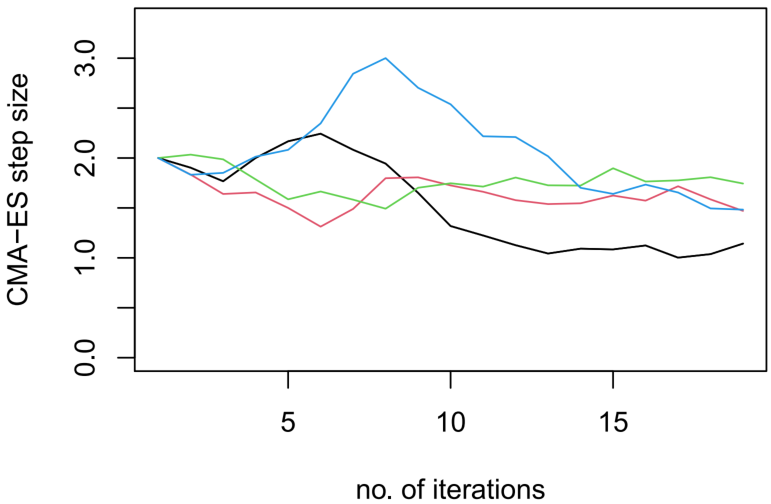
zLB <- -5 # lower bound on parameters

zUB <- 5 # upper bound on parameters

no\_test <<- 4

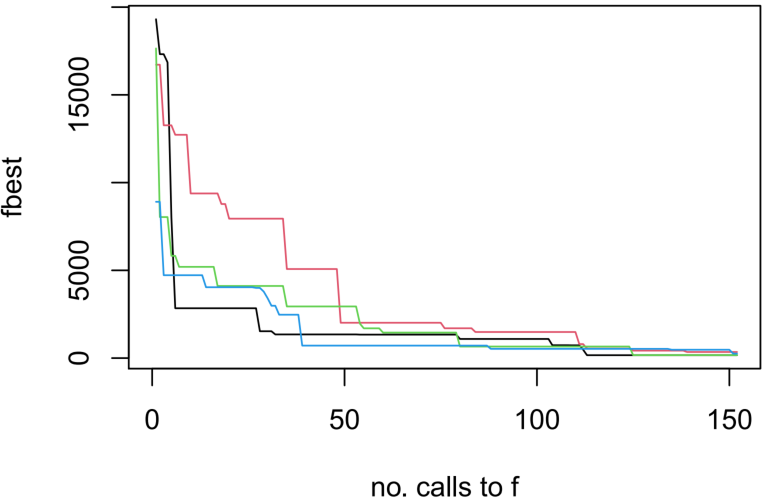


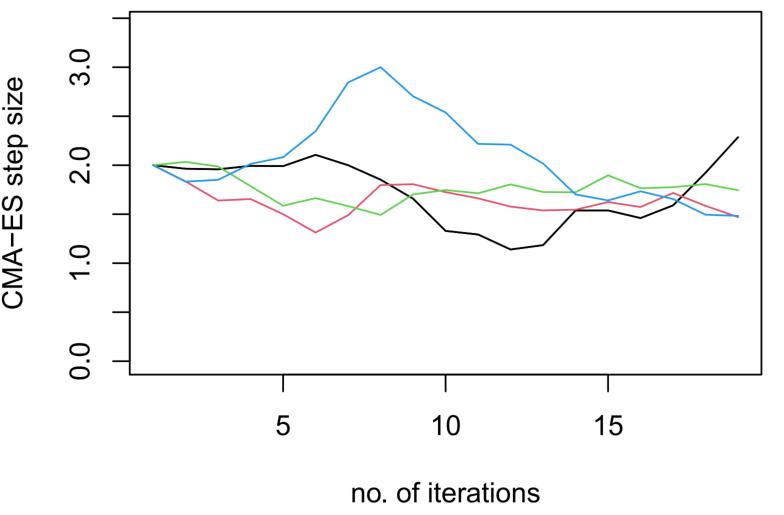




on change:



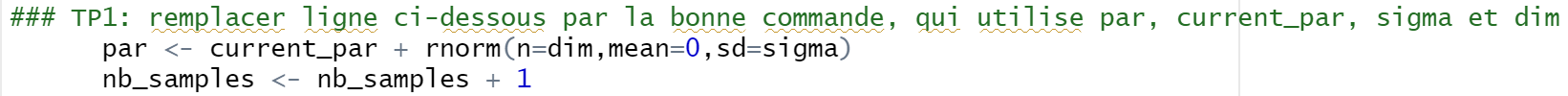




1. **Effet du pas de l’algorithme stochastique ES-(1+1)**

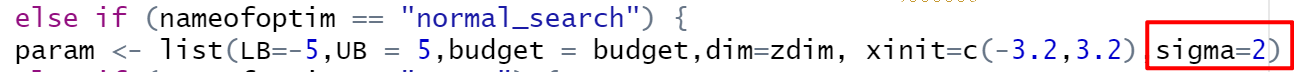
L’algorithme ES-(1+1) vu en cours est en partie codé dans le fichier normal\_search.R

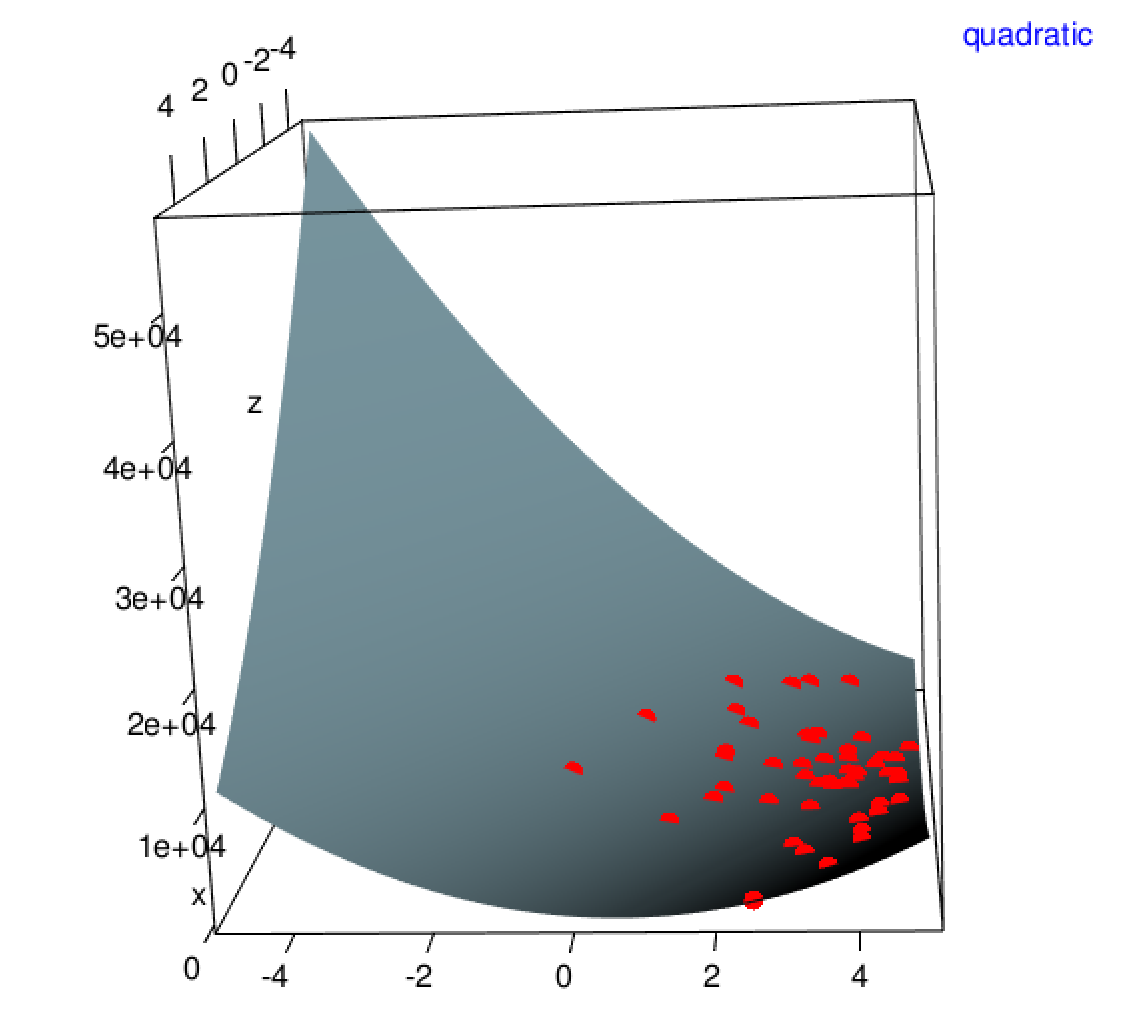
* 1. Compléter le programme normal\_search.R : remplacer la ligne stop("remplacer cette ligne par \"par <- ...\"") ) par le bon code, comme vu en cours dans l’optimiseur ES-(1+1).

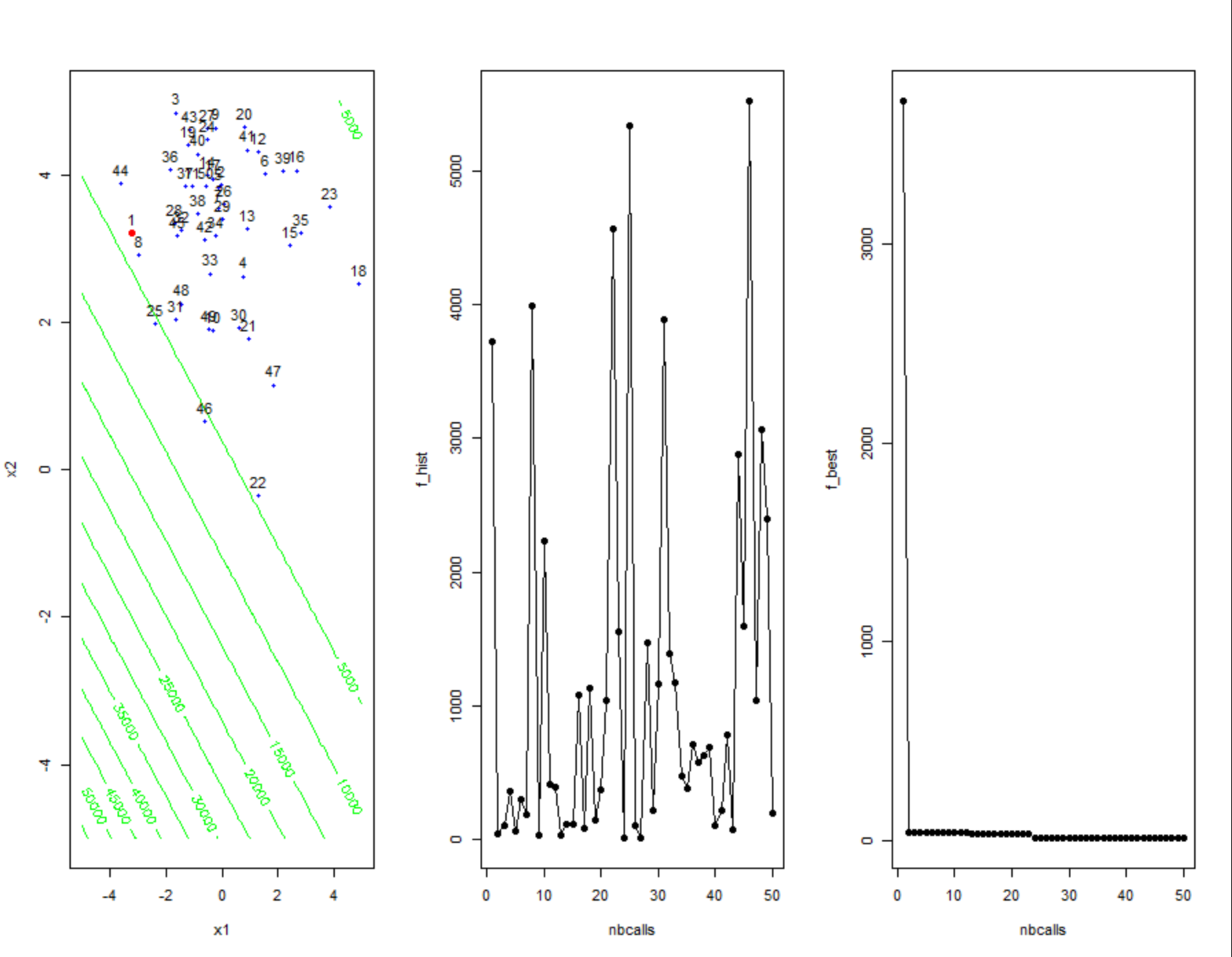


* 1. Observer en 2D (avec 3Dplots.R) que quand sigma est petit, normal\_search se comporte comme un optimiseur local et vice versa.

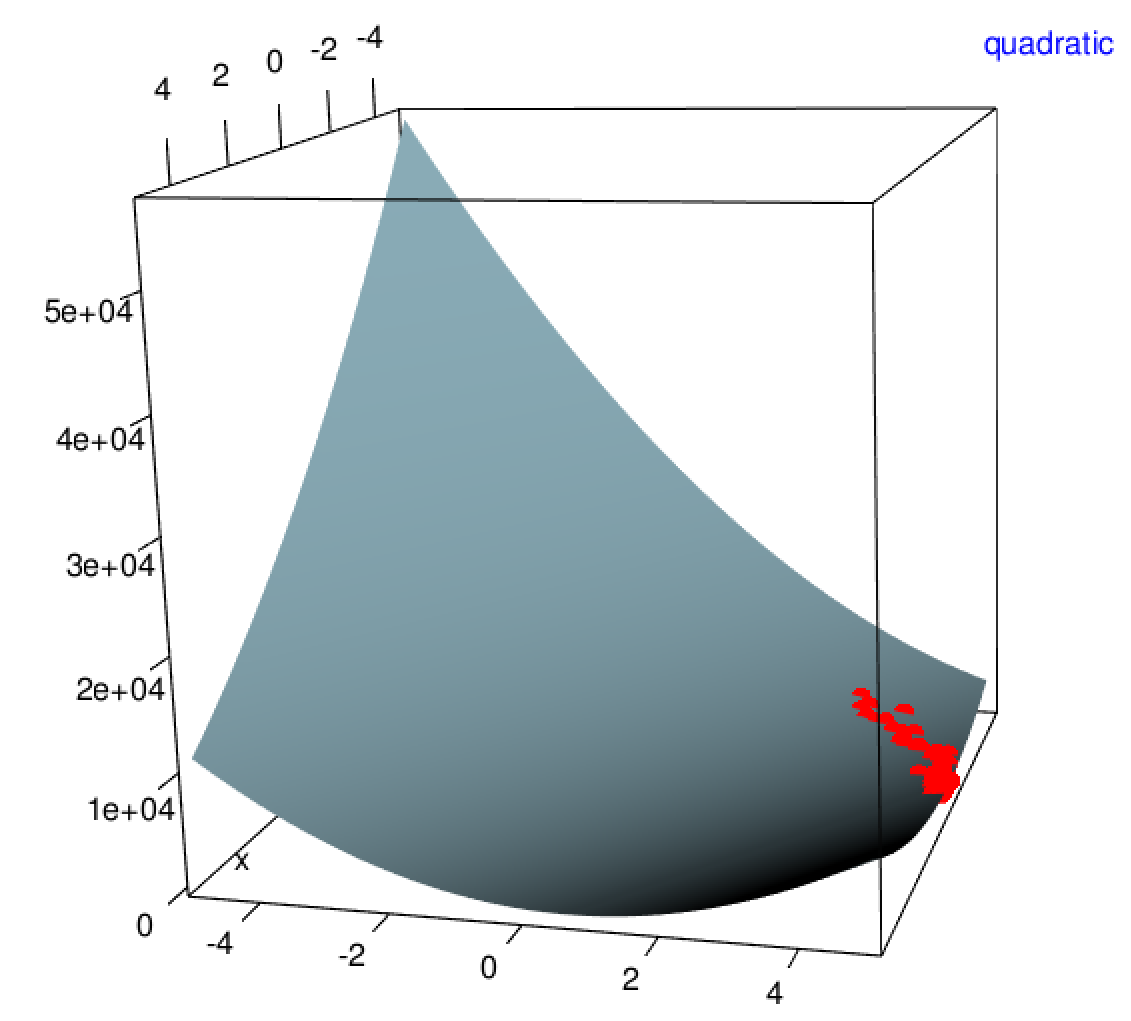
quand sigma dans normal\_search est normal est 2

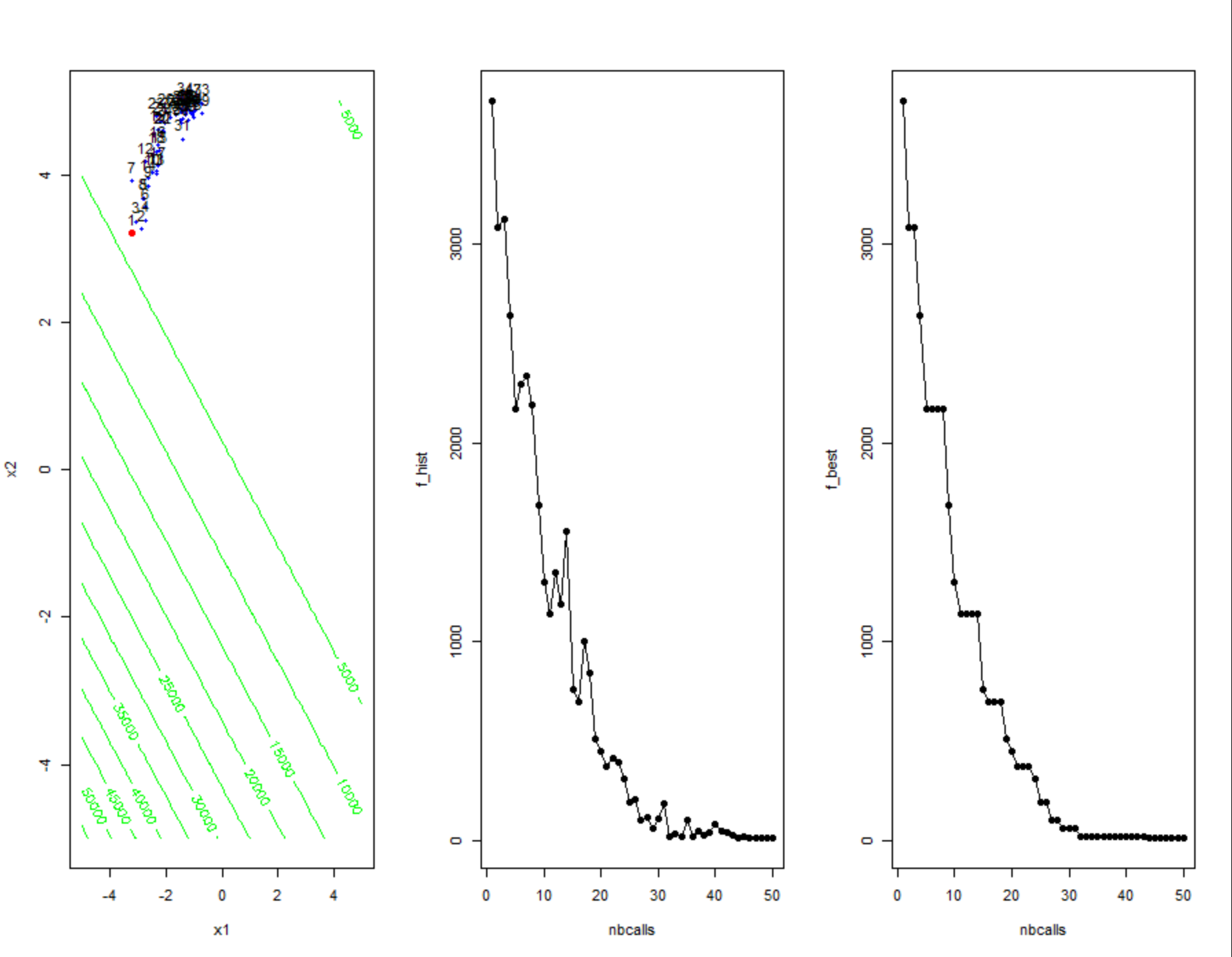






quand sigma dans normal\_search est petit est 0.2





Donc quand sigma est petit, normal\_search se comporte comme un optimiseur local

* 1. Etudier l’effet de la taille du pas (sigma) de normal\_search en 5D sur 1) la fonction sphère (comme pour la théorie du cours) et 2) une fonction multimodale de votre choix. Utiliser

« main4tests.R » pour faire les tests et « postproc\_tests\_optimizers.R » pour faire des courbes synthétiques. Discuter les résultats.

normal\_search en 5D pour la fonction sphère

Couleur :

Noir sigma 0.05

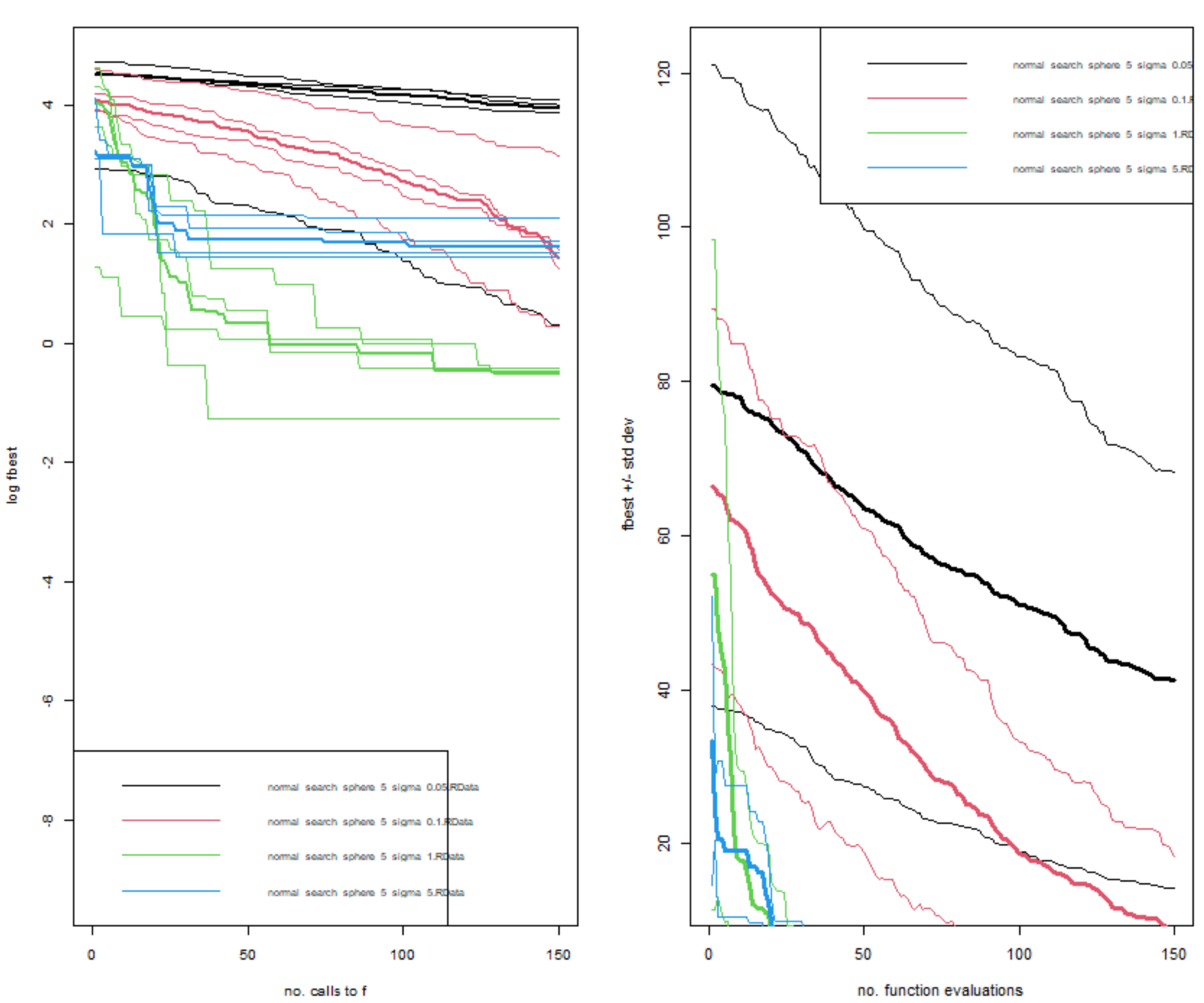
Rouge sigma 0.1

Vert sigma 1.0

Blue sigma 5.0

Dans postproc\_tests\_optimizers.R:





On vois que si on augmente la taille du pas (sigma) de normal\_search en 5D sur la fonction sphère ,si sigma plus grand et il descends plus vite et il a de meilleures performances (profondeur)

Mais si on augmenter sigma est 10,il y a erreur, la taille du pas est si grand, Il passera la frontière.

