TP 2 : Estimation de processus stochastiques

Komi AGBLODOE / Soheil SALMANI

23 janvier 2020

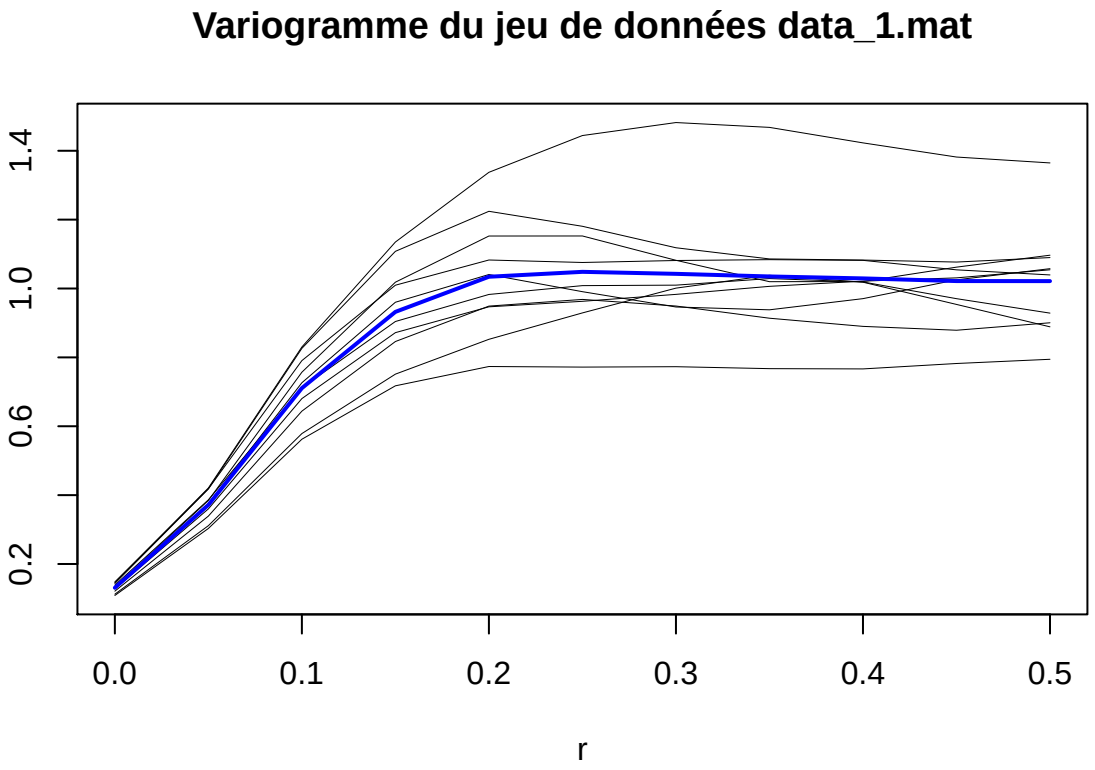
Dans ce TP, il s'agit d'estimer les fonctions de covariance de processus gaussiens en calculant leurs variogrammes. Ensuite, on réutilisera cette estimation afin de générer des données manquantes.

Tâche 1

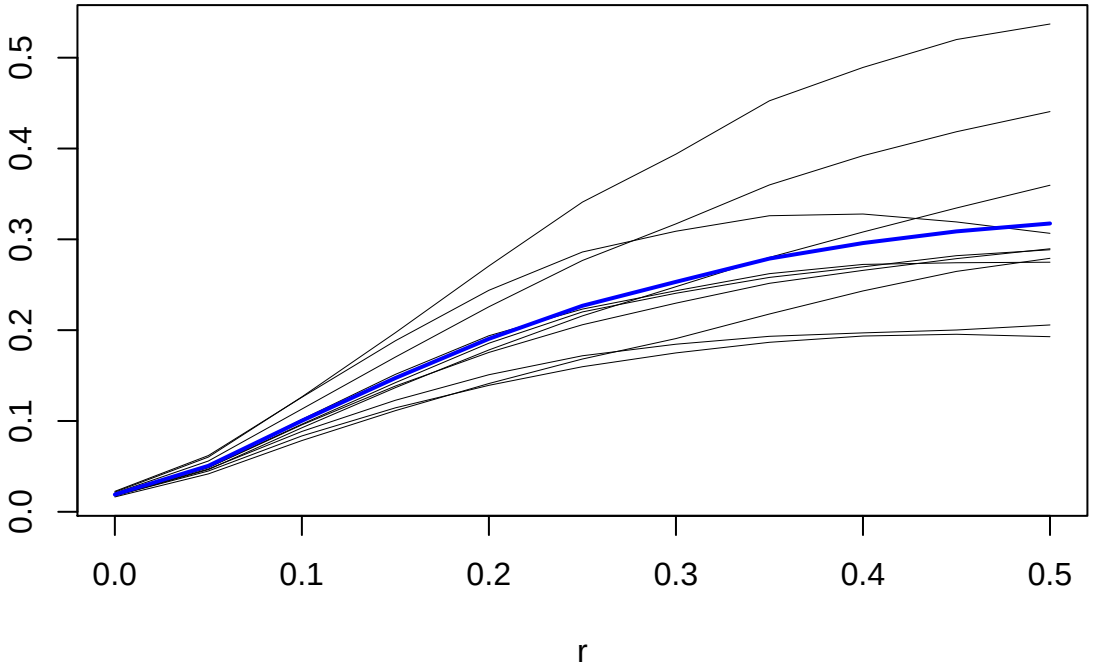
On dispose de trois différents jeux de données contenant chacun 10 réalisations. Dans un premier temps, pour chaque jeu de données, il s'agit de tracer le variogramme pour chaque réalisation à partir de la fonction $\gamma(r)$ tout en faisant varier la valeur de r ($0 \leq r \leq 0.5$), et en fixant ε le plus petit possible. Ensuite, nous traitons la moyenne de ces 10 variogrammes pour chaque jeu de données.

Variogrammes

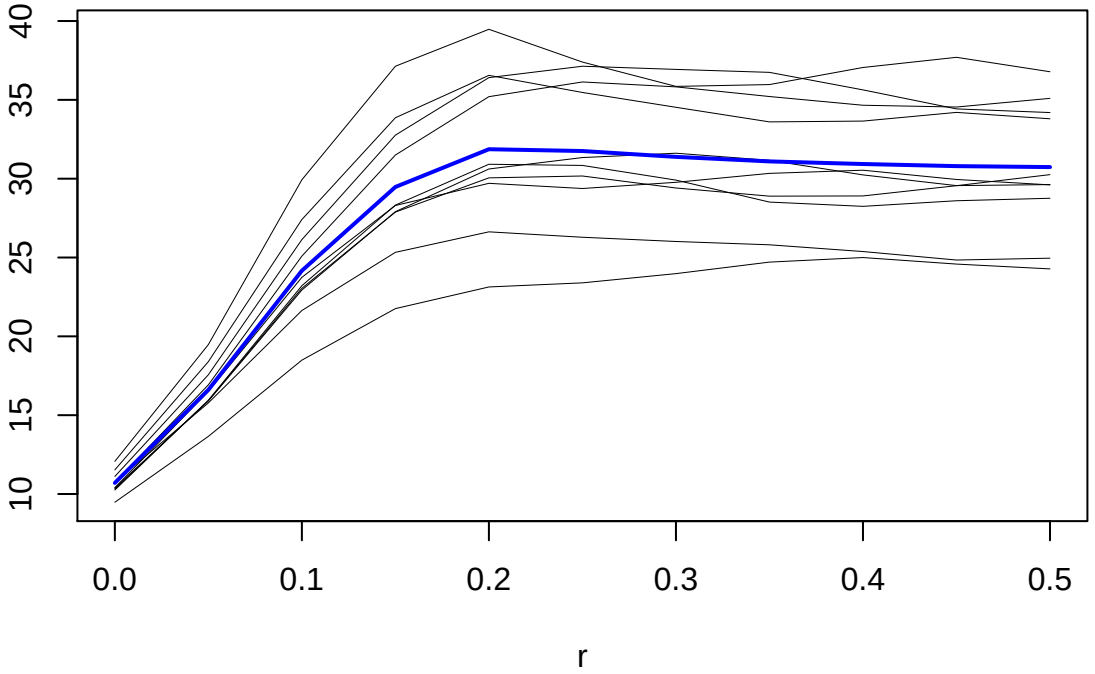
Variogramme de `data_1.mat` :



Variogramme de `data_2.mat` :



Variogramme de `data_3.mat` :



Comment choisir ε ?

ε ne doit pas être choisi trop petit au risque de ne pas avoir $N_\varepsilon(r)$ qui soit nul, et en même temps il doit pas être choisi trop grand non plus afin d'éviter d'avoir un trop grand nombre pour $N_\epsilon(r)$. De ce fait, il doit être choisi de manière empirique et en fonction du jeu de données.

Tâche 2

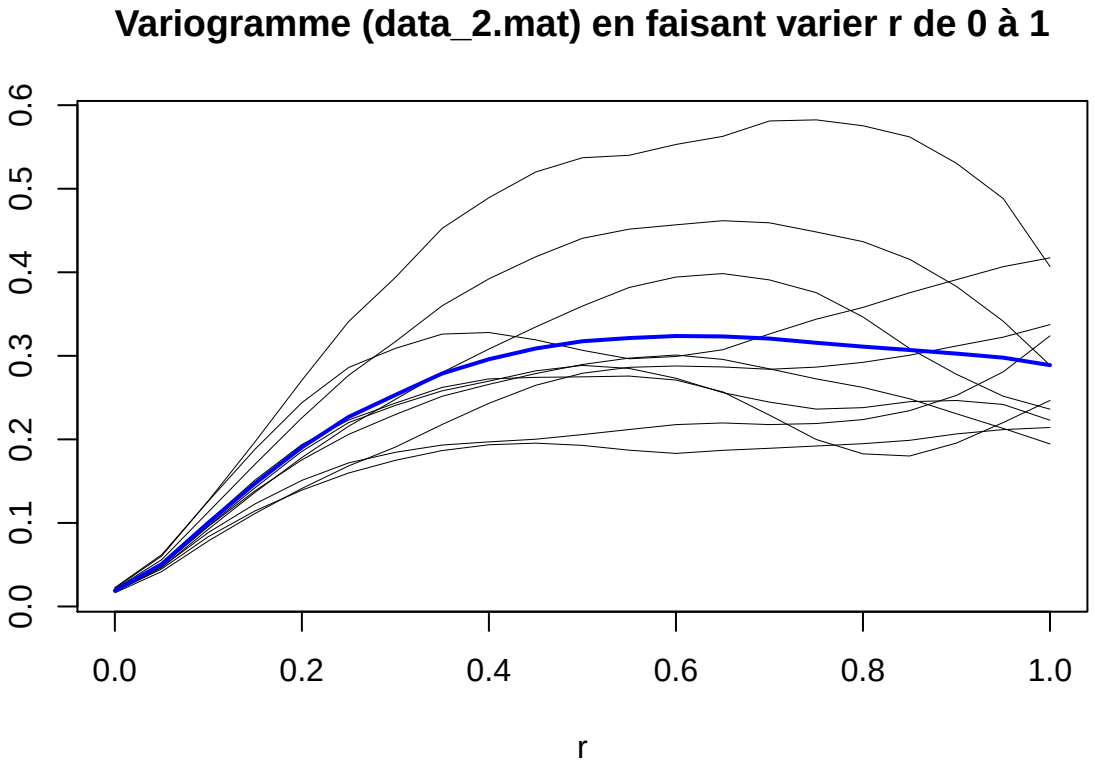
Dans cette partie, il s'agit de retrouver le noyau à partir duquel les données `data_1.mat`, `data_2.mat` et `data_3.mat` ont été obtenues. Les noyaux parmi lesquels l'identification doit être faite sont les suivants : *Gaussien*, *Sphérique + Delta* et *Matérn-3/2*.

Sur le graphe des variogrammes du jeu de données *data_3.mat*, on remarque bien que la courbe représentant la moyenne des 10 variogrammes commencent à partir de la valeur 10 contrairement aux courbes représentant les moyennes des variogrammes des jeux de données `data_1.mat` et `data_2.mat`. Ceci peut être expliqué par le bruit σ_0 du noyau *Sphérique + Delta*.

De ce fait, on estime donc la valeur de σ_0 à 10. De plus graphiquement, on obtient $a = 0.2$ comme portée (la courbe de la moyenne converge à partir de cette valeur) et $\sigma^2 = 32$ (représentant la valeur atteinte à partir de la portée) environ pour le noyau *Sphérique + Delta*. On conclut donc que le jeu de données *data_3.mat* a été obtenu à partir du noyau *Sphérique + Delta* avec comme paramètres $\sigma_0 = 10$, $a = 0.2$ et $\sigma = \sqrt{32}$.

Quand au graphe des variogrammes du jeu de données *data_1.mat*, on remarque bien que la courbe représentant la moyenne des 10 variogrammes ne présente pas assez de variabilité. Ce qui est une caractéristique gaussienne. De plus cette courbe converge à partir de la valeur 0.2. Ce qui représente la valeur de a. On obtient de même graphiquement la valeur 1 comme σ^2. On conclut donc que le jeu de données *data_1.mat* a été obtenu à partir du noyau gaussien avec comme paramètres $a = 0.2$ et $\sigma = 1$.

Pour le jeu de données *data_2.mat*, nous avons remarqué que la courbe représentant la moyenne des 10 variogrammes ne présente pas assez rapidement de convergence (en prenant $0 \leq r \leq 0.5$). Cela ne nous permet pas vraiment de déterminer les paramètres. Ce qui nous a amené à faire varier r de 0 à 1 au lieu de 0 à 0.5.



Ce faisant, on remarque que la convergence intervient entre les valeurs 0.5 et 0.6 pour une valeur de σ^2 approximativement égale à 0.3. Même si nous n'arrivons pas bien à déceler les caractéristiques de manière précise, nous sommes tentés de dire que le jeu de données `data_2.mat` provient d'un noyau *Matérn-3/2* avec comme paramètres a appartenant à l'intervalle $[0.5, 0.6]$ et $\sigma = \sqrt{0.3}$.

Annexe

Code de la tâche 1

Variogramme de `data_1.mat` :

```
library(R.matlab)
data <- readMat("data/data_1.mat")
x <- data$x
y <- data$y
X_1 <- data$X.1
X_2 <- data$X.2
X_3 <- data$X.3
X_4 <- data$X.4
X_5 <- data$X.5
X_6 <- data$X.6
X_7 <- data$X.7
X_8 <- data$X.8
X_9 <- data$X.9
X_10 <- data$X.10
X_incomplet <- data$X.incomplet

epsilon <- 0.05

n <- 32
S <- data.matrix(expand.grid(x, y))

Dist <- as.matrix(dist(S))

r <- seq(0, 0.5, 0.05)
variogr <- matrix(0, ncol = length(r), nrow = 10)

for (l in 1:10) {
  X <- data[[l]]
  X_vect <- as.vector(X)

  for (k in 1:length(r)) {
    cond <- which((r[k] - epsilon <= Dist) & (Dist <= r[k] + epsilon), arr.ind = TRUE)

    N2 <- nrow(cond)
    GAMMA <- rep(NA, N2)
    for (i in 1:N2) {
      GAMMA[i] <- (X_vect[cond[i, 1]] - X_vect[cond[i, 2]])^2
    }

    GAMMA <- 1 / (2 * N2) * sum(GAMMA)
    variogr[l, k] <- GAMMA
  }
}
```

```r
plot(r, variogr[1, ],
  main = "Variogramme du jeu de données data_1.mat",
  type = "l",
  ylim = c(min(variogr), max(variogr)),
  ylab = "",
  lwd = 0.5
)

for (i in 2:10) {
  lines(r, variogr[i, ], lwd = 0.5)
}

lines(r, apply(variogr, 2, mean), type = "l", col = "blue", lwd = 2)
```

Variogramme de `data_2.mat` :

```r
data <- readMat("data/data_2.mat")
x <- data$x
y <- data$y
X_1 <- data$X.1
X_2 <- data$X.2
X_3 <- data$X.3
X_4 <- data$X.4
X_5 <- data$X.5
X_6 <- data$X.6
X_7 <- data$X.7
X_8 <- data$X.8
X_9 <- data$X.9
X_10 <- data$X.10
X_incomplet <- data$X.incomplet

epsilon <- 0.05

n <- 32
S <- data.matrix(expand.grid(x, y))

Dist <- as.matrix(dist(S))

r <- seq(0, 0.5, 0.05)
variogr <- matrix(0, ncol = length(r), nrow = 10)

for (l in 1:10) {
  X <- data[[l]]
  X_vect <- as.vector(X)

  for (k in 1:length(r)) {
    cond <- which((r[k] - epsilon <= Dist) & (Dist <= r[k] + epsilon), arr.ind = TRUE)

    N2 <- nrow(cond)
    GAMMA <- rep(NA, N2)
    for (i in 1:N2) {
      GAMMA[i] <- (X_vect[cond[i, 1]] - X_vect[cond[i, 2]])^2
    }
```

```
    GAMMA <- 1 / (2 * N2) * sum(GAMMA)
    variogr[l, k] <- GAMMA
  }
}

plot(r, variogr[1, ],
  main = "Variogramme du jeu de données data_2.mat",
  type = "l",
  ylim = c(min(variogr), max(variogr)),
  ylab = "",
  lwd = 0.5
)

for (i in 2:10) {
  lines(r, variogr[i, ], lwd = 0.5)
}

lines(r, apply(variogr, 2, mean), type = "l", col = "blue", lwd = 2)
```

Variogramme de `data_3.mat` :

```
data <- readMat("data/data_3.mat")
x <- data$x
y <- data$y
X_1 <- data$X.1
X_2 <- data$X.2
X_3 <- data$X.3
X_4 <- data$X.4
X_5 <- data$X.5
X_6 <- data$X.6
X_7 <- data$X.7
X_8 <- data$X.8
X_9 <- data$X.9
X_10 <- data$X.10
X_incomplet <- data$X.incomplet

epsilon <- 0.05

n <- 32
S <- data.matrix(expand.grid(x, y))

Dist <- as.matrix(dist(S))

r <- seq(0, 0.5, 0.05)
variogr <- matrix(0, ncol = length(r), nrow = 10)

for (l in 1:10) {
  X <- data[[l]]
  X_vect <- as.vector(X)

  for (k in 1:length(r)) {
    cond <- which((r[k] - epsilon <= Dist) & (Dist <= r[k] + epsilon), arr.ind = TRUE)

    N2 <- nrow(cond)
```

```
    GAMMA <- rep(NA, N2)
    for (i in 1:N2) {
      GAMMA[i] <- (X_vect[cond[i, 1]] - X_vect[cond[i, 2]])^2
    }

    GAMMA <- 1 / (2 * N2) * sum(GAMMA)
    variogr[l, k] <- GAMMA
  }
}


plot(r, variogr[1, ],
  main = "Variogramme du jeu de données data_3.mat",
  type = "l",
  ylim = c(min(variogr), max(variogr)),
  ylab = "",
  lwd = 0.5
)

for (i in 2:10) {
  lines(r, variogr[i, ], lwd = 0.5)
}

lines(r, apply(variogr, 2, mean), type = "l", col = "blue", lwd = 2)
```

Code de la tâche 2

Variogramme de `data_2.mat` en faisant varier r de 0 à 1 :

```
data <- readMat("data/data_2.mat")
x <- data$x
y <- data$y
X_1 <- data$X.1
X_2 <- data$X.2
X_3 <- data$X.3
X_4 <- data$X.4
X_5 <- data$X.5
X_6 <- data$X.6
X_7 <- data$X.7
X_8 <- data$X.8
X_9 <- data$X.9
X_10 <- data$X.10
X_incomplet <- data$X.incomplet

epsilon <- 0.05

n <- 32
S <- data.matrix(expand.grid(x, y))

Dist <- as.matrix(dist(S))

r <- seq(0, 1, 0.05)
variogr <- matrix(0, ncol = length(r), nrow = 10)
```

```r
for (l in 1:10) {
  X <- data[[l]]
  X_vect <- as.vector(X)

  for (k in 1:length(r)) {
    cond <- which((r[k] - epsilon <= Dist) & (Dist <= r[k] + epsilon), arr.ind = TRUE)

    N2 <- nrow(cond)
    GAMMA <- rep(NA, N2)
    for (i in 1:N2) {
      GAMMA[i] <- (X_vect[cond[i, 1]] - X_vect[cond[i, 2]])^2
    }

    GAMMA <- 1 / (2 * N2) * sum(GAMMA)
    variogr[l, k] <- GAMMA
  }
}

plot(r, variogr[1, ],
  main = "Variogramme du jeu de données data_2.mat",
  type = "l",
  ylim = c(min(variogr), max(variogr)),
  ylab = "",
  lwd = 0.5
)

for (i in 2:10) {
  lines(r, variogr[i, ], lwd = 0.5)
}

lines(r, apply(variogr, 2, mean), type = "l", col = "blue", lwd = 2)
```