Vecteurs Aléatoires, Conditionnement, Simulation
TP 1

B. Delyon, V. Monbet

Master 1 - 2013-2014

Exercice 1 - Simulation d'un vecteur Gaussien

Questions préparatoires (TD)
Soit $Z \sim \mathcal{N}(0, \mathbb{I}_d)$ un vecteur aléatoire d-dimensionel.

1. Soit Σ une matrice symétrique définie positive. La matrice S est une racine carrée de Σ si et seulement si $\Sigma = SS^T$.
   Vérifier que $S = PD^{1/2}$ est une racine carrée de Σ avec $D^{1/2}$ une matrice diagonale ayant les racines carrées des valeurs propres de Σ sur sa diagonale et P la matrice des vecteurs propres associés.

2. (a) Donner la loi de SZ
   (b) Soit μ un vecteur de $\mathbb{R}^d$. Donner la loi de $X = \mu + SZ$.
   (c) Quelle est la loi de la première composante de $X = (X_1, \cdots, X_d)^T$?

3. Supposons maintenant que $d = 2$. On rappelle que la densité conditionnelle de Y sachant $X = x$ est
$$f(y|x) = \frac{f(x,y)}{f(x)}$$
   Quelle est la loi conditionnelle de X_1 sachant X_2? On montra en particulier que
$$E(X_1|X_2 = x_2) = \mu_2 + \sigma_{12}/\sigma_{11}(x_1 - \mu_1)$$
$$Var(X_1|X_2 = x_2) = \sigma_{11} - s_{12}\frac{1}{\sigma_{22}}s_{12}$$
   Remarque : le dernier résultat se généralise à des vecteurs de dimension supérieure à 2.

Soit $Z = (X, Y)$ un vecteur aléatoire gaussien de dimension $m + d$, de moyenne et de covariance
$$\mu_Z = \begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}, \quad \Sigma_Z = \begin{pmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{XY} & \Sigma_Y \end{pmatrix}$$
Si la matrice Σ_Y est inversible, alors la loi conditionnelle de $X|Y = y$ est gaussienne de moyenne
$$\mu_{X|Y=y} = \mu_X + \Sigma_{XY}\Sigma_Y^{-1}(y - \mu_Y)$$
et de covariance
$$\Sigma_{X|Y=y} = \Sigma_X - \Sigma_{XY}\Sigma_Y^{-1}\Sigma_{XY}$$

Question 1 - Simulation

1. Ecrire une fonction `mvrgauss` qui permette de simuler n réalisations d'un vecteur gaussien de moyenne μ et de variance Σ. L'objet résultant devra être une matrice de dimension $n \times d$.

```
> mvrgauss <- function(n,mu,Sigma) {
    d = length(mu)
    X = matrix(NA,n,d) # n réalisations d'un d vecteur
    Z = matrix(...,n,d)
    E = eigen(Sigma)
    P = ... # vecteurs propres
    D = ... # valeurs propres
    X = ...
    return(X)
}
```

2. Simuler 500 réalisations un vecteur gaussien de dimension 3 de moyenne μ et de covariance Σ avec

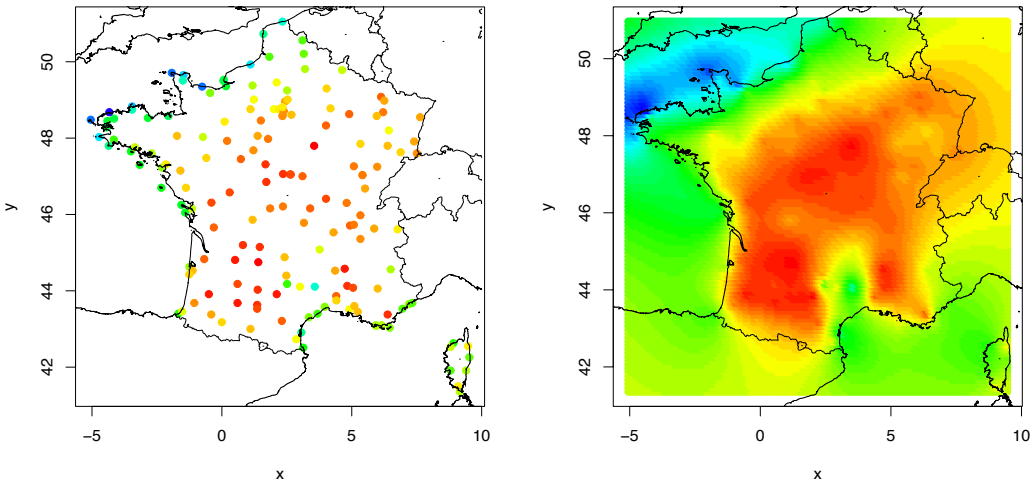
$$\mu = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 1 & .7 & -.3 \\ .7 & 2 & 0 \\ -.3 & 0 & 3 \end{pmatrix}$$

```
> d = 3
> mu = matrix(c(1,0,-1),d,1)
> Sigma = matrix(c(1,.7,-.3,.7,2,0,-.3,0,3),d,d)
> n = 500
> X =  mvrgauss(n,mu,Sigma)
> pairs(X,pch=20)
```

Tracer les nuages de points (utiliser `pairs`). et interpréter les graphiques obtenus. Estimer les densités marginales de l'échantillon généré, en utilisant par exemple l'histogramme. Pour chaque composante, superposer la densité théorique.

Question 3 - Prévision, krigeage

On dispose de données de température moyenne journalière en plusieurs villes de France au mois d'août 2003. On suppose que le vecteur aléatoire correspondant suit une loi de Gauss multivariée.



1. Commençons par visualiser les données en utilisant les instructions ci-dessous. Le fichier `data_temperature.Rdata` contient une table `D` contenant les longitudes et l'attitudes des points ainsi que la température mesurée en ces points. Le fichier `frontieres_france.Rdata` contient une table `C` représentant les lignes de frontière.

```
>load("~/Dropbox/ENSEIGNEMENT/PrInfoM1/data_temperature.Rdata")
>load("~/Dropbox/ENSEIGNEMENT/PrInfoM1/frontieres_france.Rdata")
>load("~/Dropbox/ENSEIGNEMENT/PrInfoM1/grille_france.Rdata")
>plot3 = function(x,y,z,C = NULL,add=F,col.scale=NULL){
L <- 64
c <- rainbow(L) # créer un vecteur correspondant à des couleurs
c <- c[length(c):1] # inverse le sens du vecteur c
ind = z
w = which(!is.na(z))
if (is.null(col.scale)) {
min.z = min(z[w])
max.z = max(z[w])
} else {
min.z = col.scale[1]
max.z = col.scale[2]
}
ind[w] <- round( (z[w]-min.z)/(max.z-min.z)*(L-20))+20 # pour déterminer
           #la couleur des lignes de D
ind[ind<20] = 20
ind[ind>64] = 64
if (add==F) {
plot(x,y,col=c[ind],pch=19)
if (!is.null(C)) {lines(C$x,C$y)}
} else {
```

```
points(x,y,col=c[ind],pch=19)
}
}
>plot3(T$lon,T$lat,T$data,C=C)
```

2. On suppose que le processus observé est stationnaire ce qui implique que sa moyenne est la même quelque soit le point considéré (cette hypothèse est un peu forte...) et que sa covariance entre 2 points ne dépend que de la distance qui les sépare. Ici un point correspond à une composante du vecteur gaussien.
   Estimer la moyenne empirique commune de chaque composante du vecteur gaussien.

3. L'estimation de la covariance est un peu plus technique et peut se faire de différentes façons. On propose ici d'utiliser l'estimateur du maximum de vraisemblance. La dimension de la covariance est grande puisqu'elle dépend du nombre de points observés. Il n'est pas raisonnable d'estimer autant de paramètres. On choisit alors une forme paramétrique pour la covariance. Une étude préliminaire, qu'on ne refait pas ici, montre qu'il convient de choisir une forme exponentielle c'est à dire que

   $$C(x, y) = C_0 \exp(-\lambda d(x, y))$$

   où $d(x, y)$ est la distance entre les points x et y, C_0 est la covariance pour une distance 0 (ie la variance). Et λ est alors le seul paramètre à estimer. Pour calculer la distance entre deux points on devra prendre ne considération que les points sont défit pas leur position géographique qui tient compte de la rotondité de la terre. On utilisera la fonction `m_lldist_p` (fournie sur la page web).

   (a) On peut programmer une fonction pour calculer la covariance en fonction de la distance et du paramètre

   ```
   cov.th.e <- function(d,par) { # covariance exponentielle
   eps = 10
   cov.max = 16.2 # covariance maximum = variance
   cov = cov.max*exp(-(d/par[2]))
   wd = which(d<eps)
   cov[wd] = 16.2
   return(cov)
   }
   ```

   ainsi que la log-vraisemblance

   ```
   ll_gauss_mult = function(par,x,y,z) {
   n = length(z)
   C = matrix(0,n,n)
   for (i in 1:n) {
   d = m_lldist_p(x,y,x[i],y[i])
   C[i,] = cov.th.e(d,par)
   }
   z = matrix(z,n,1)
   ll =  ...  # log-vraisemblance
   return(-ll) # -, pour l'optimisation numérique
   }
   ```

(b) On peut alors calculer et tracer la log-vraisemblance pour différentes valeurs du paramètre.

```
lambda.tab = seq(40,400,by=10)
ll = matrix(0,1,length(lambda.tab))
for (j in 1:length(lambda.tab)) {
ll[j] = ll_gauss_mult(...)
}
plot(lambda.tab,ll,pch=20)
```

Donner la valeur optimale de λ au sens de la vraisemblance. On l'appelle `lambda0` dans la suite.

4. On peut alors calculer la covariance entre un point de la France de coordonnées géographiques `(lon.y,lat.y)` et n'importe quel autre point de coordonnées géographiques `(lon.x,lat.x)`

```
dst <- m_lldist_p(lon.x,lat.x,lon.y,lat.y)
C.xy <- cov.th.e(dst,lambda0)
```

Choisir deux points et calculer leur covariance.
Calculer les covariances entre le point de coordonnées (-1.7,48) [Rennes] et tous les points pour lesquels on dispose d'observations. On notera `C.XY` cette covariance.

5. De façon plus générale, construire la matrice de covariance de la température en un ensemble de n points. On note `pts.kp` la liste des points.

```
C.YY <- matrix(0,n.kp,n.kp )
cnt = 0
for (i in pts.kp){
     dst <- m_lldist_p(D$lon[pts.kp],D$lat[pts.kp],D$lon[i],D$lat[i])
     cnt = cnt+1
     C.YY[cnt,] <- cov.th.e(dst,lambda0)
}
```

6. Calculer l'espérance conditionnelle de la température à Rennes sachant les températures observées.

7. Recommencer pour tous les points de la grille définie dans le fichier `grille.dat`. On obtient un vecteur `X.hat` de la même longueur que `G$lon` et `G$lat`. On peut alors tracer la carte :

```
plot3(G$lon,G$lat,X.hat,C=C)
```

8. Calculer maintenant la variance conditionnelle puis en déduire l'écart-type correspondant. La carte de l'écart-type donne une information sur la qualité de l'estimation de l'espérance conditionnelle.

9. Pour construire ce modèle de prédiction, nous avons supposé que les données étaient issues d'un vecteur gaussien tel que les dépendances entre deux composantes ne dépendent que de la distance géographique entre les composantes. Cette hypothèse est vraisemblablement un peu forte, et il est utile de faire une validation plus approfondie du modèle. Proposer une méthode de validation croisée.