1 TD 8 : Variables normales corrélées

(correction page ??)

Abordé lors de cette séance	
programmation	`numpy` ou calcul matriciel
algorithme	décorrélation de variables normales

Le calcul matriciel est aujourd'hui très répandu et présent dans la plupart des logiciels mathématiques gratuits tels que *R*[1], *SciLab*[2], *Octave*[3] ou payants *Gauss*[4], *Matlab*[5], *S+*[6]. Le language *Python* propose un module qui reprend le calcul matriciel proposé par tous ces langages avec des notations similaires. C'est un module qu'il faut télécharger sur Internet à l'adresse suivante : http://numpy.scipy.org/[7]. Un tutoriel en anglais est aussi disponible à l'adresse suivante : http://www.scipy.org/Tentative_NumPy_Tutorial[8].

Ce TD appliquera le calcul matriciel aux vecteurs de variables normales corrélées[9].

Première demi-heure : création d'un jeu de données aléatoires

1) La première étape consiste à construire des variables aléatoires normales corrélées dans une matrice $N \times 3$. En vous inspirant du TD précédent, on cherche à construire cette matrice au format `numpy`. Le programme suivant est un moyen de construire un tel ensemble à l'aide de combinaisons linéaires. Complétez les lignes contenant des .....

```
import random
import numpy as np

def combinaison () :
    x = random.gauss(0,1) # génère un nombre aléatoire
    y = random.gauss(0,1) # selon une loi normale
    z = random.gauss(0,1) # de moyenne null et de variance 1
    x2 = x
    y2 = 3*x + y
    z2 = -2*x + y + 0.2*z
    return [x2, y2, z2]

mat = [ ............ ]
npm = np.matrix ( mat )
```

1. http://www.r-project.org/, c'est le plus utilisé par les chercheurs dans des domaines à ceux que l'ENSAE aborde.
2. http://www.scilab.org/
3. http://www.gnu.org/software/octave/
4. http://www.aptech.com/
5. http://www.mathworks.com/
6. http://spotfire.tibco.com/products/s-plus/statistical-analysis-software.aspx
7. Il faut faire attention de bien choisir la version correspondant à votre système d'exploitation (Windows, Linux, Apple) et à la version de votre langage *Python*.
8. voir aussi http://www.scipy.org/NumPy_for_Matlab_Users
9. voir http://fr.wikipedia.org/wiki/Covariance, ou aussi http://fr.wikipedia.org/wiki/D%C3%A9composition_en_valeurs_singuli%C3%A8res.

2) A partir de la matrice `npm`, on veut construire la matrice des corrélations.

```
npm = ...               # voir question précédente
t   = npm.transpose ()
a   = t * npm
a  /= npm.shape[0]
```

A quoi correspond la matrice `a` ?

Seconde demi-heure : matrice de corrélation

3) Construire la matrice des corrélations à partir de la matrice `a`. Si besoin, on pourra utiliser le module `copy`.

```
import copy
b = copy.copy (a)    # remplacer cette ligne par b = a
b [0,0] = 44444444
print b              # et comparer le résultat ici
```

4) Construire une fonction qui prend comme argument la matrice `npm` et qui retourne la matrice de corrélation. Cette fonction servira plus pour vérifier que nous avons bien réussi à décorréler.

```
def correlation ( npm ) :
    ..........
    return .....
```

Pour la suite, un peu de mathématique. On note M la matrice `npm`. $V = \frac{1}{n}M'M$ correspond à la matrice des **covariances** et elle est nécessairement symétrique. C'est une matrice diagonale si et seulement si les variables normales sont indépendantes. Comme toute matrice symétrique, elle est diagonalisable. On peut écrire :

$$\frac{1}{n}M'M = P\Lambda P' \tag{1}$$

P vérifie $P'P = PP' = I$. La matrice Λ est diagonale et on peut montrer que toutes les valeurs propres sont positives ($\Lambda = \frac{1}{n}P'M'MP = \frac{1}{n}(MP)'(MP)$).

On définit alors la racine carrée de la matrice Λ par :

$$\Lambda = diag(\lambda_1, \lambda_2, \lambda_3) \tag{2}$$

$$\Lambda^{\frac{1}{2}} = diag\left(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \sqrt{\lambda_3}\right) \tag{3}$$

On définit ensuite la racine carrée de la matrice V :

$$V^{\frac{1}{2}} = P\Lambda^{\frac{1}{2}}P' \tag{4}$$

On vérifie que $\left(V^{\frac{1}{2}}\right)^2 = P\Lambda^{\frac{1}{2}}P'P\Lambda^{\frac{1}{2}}P' = P\Lambda^{\frac{1}{2}}\Lambda^{\frac{1}{2}}P' = V = P\Lambda P' = V$.

Troisième demi-heure : calcul de la racine carrée

5) Le module `numpy` propose une fonction qui retourne la matrice P et le vecteur des valeurs propres L :

```
L,P = np.linalg.eig(a)
```

Vérifier que $P'P = I$. Est-ce rigoureusement égal à la matrice identité ?

6) Que fait l'instruction suivante :

```
print np.diag(L)
```

7) Ecrire une fonction qui calcule la racine carrée de la matrice $\frac{1}{n}M'M$ (on rappelle que M est la matrice `npm`) [10].

Quatrième demi-heure : décorrélation

La fonction suivante permet d'obtenir l'inverse de la matrice `a`.

```
np.linalg.inv(a)
```

8) Chaque ligne de la matrice M représente un vecteur de trois variables corrélées. La matrice de covariance est $V = \frac{1}{n}M'M$. Calculer la matrice de covariance de la matrice $N = MV^{-\frac{1}{2}}$ (mathématiquement).

9) Vérifier numériquement.

Pour aller plus loin ou pour ceux qui ont fini plus tôt

10) A partir du résultat précédent, proposer une méthode pour simuler un vecteur de variables corrélées selon une matrice de covariance V à partir d'un vecteur de lois normales indépendantes.

11) Proposer une fonction qui crée cet échantillon :

```
def simultation (N, cov) :
    # simule un échantillon de variables corrélées
```

10. http://fr.wikipedia.org/wiki/Racine_carr%C3%A9e_d'une_matrice

```
    # N : nombre de variables
    # cov : matrice de covariance
    ...
    return M
```

12) Vérifier que votre échantillon a une matrice de corrélations proche de celle choisie pour simuler l'échantillon.