TP Statistiques et Séries Chronologiques
Université de Lorraine

Régression Linéaire Multiple

Clément Dell'Aiera

1 Exercice 1

1. Que font les commandes *pnorm*, *qnorm* , *rnorm*, *dnorm* ? Utilisez l'aide de *R*.
2. Tracer la fonction de densité de la loi normale. Faites varier les paramètres et afficher les différentes courbes sur le même graphique.
3. Simuler 2 vecteurs X et Y contenant chacun $N = 100$ variables indépendantes identiquement distribuées suivant une loi normale $\mathcal{N}(0, 1)$.
4. Afficher les points de coordonnées $(X[j], Y[j])$ dans le plan, pour j allant de 1 à 100.
5. Tracer la fonction de répartition empirique des $X[j]$.
6. Soit $\mathcal{E}$ une v.a. de loi exponentielle de paramètre 1, et U une v.a. suivant une loi uniforme sur $[0, 2\pi]$. On pose

$$(X, Y) = (\sqrt{\mathcal{E}}\cos(U), \sqrt{\mathcal{E}}\sin(U)).$$

Quelle est la loi du couple (X, Y) ? (Vous pouvez le prouvez, ou observer grâce à R ce qu'il se passe en simulant ces variables et en les traçant.)

2 Exercice 2

1. Une table est déjà en mémoire dans R : la table *stackloss*. Analyser la rapidement.
2. Tracer *stack.loss* en fonction de *Air.Flow*. Qu'en pensez-vous ?
3. Effectuer la régression linéaire de *stack.loss* en fonction des autres variables. Quelles sont celles qui sont significatives ?

3 Exercice 3

Le fichier *ozone.dta* contient les variables suivantes, pour une série de journées (qui sont ici nos individus) :
– l'identifiant de la journée,

– le maximum d'ozone (variable maxO3)
– l'heure à laquelle le maximum d'ozone a été obtenu (heure),
– les températures à 6h, 9h, 12h, 15h, 18h (resp. T6 à T18)
– la nébulosité à 6h, 9h, 12h, 15h, 18h (resp. Ne6 à Ne18)
– la projection du vent sur l'axe est-ouest à 12h (Vx),
– le maximum d'ozone de la veille (maxO3v).

Le but est de modéliser la valeur des pics d'ozone en fonction de grandeurs physiques facilement mesurables (température, heure, nébulosité, vent) afin d'avoir des approximations de la qualité de l'air faciles et rapides à obtenir.

1. Importer la table, et afficher un résumé de ce qu'elle contient.
2. Tracer *maxO3* en fonction de *T12*, puis effectuer un régression linéaire. Ajouter la droite de régression sur le graphique. Soignez la présentation.
3. Afficher les résultats de la régression.
4. Extraire les résidus et tracer leur densité estimée.
5. Effectuer la régression de *maxO3* sur toutes les variables, et supprimer récursivement celles qui ne sont pas significatives, jusqu'à ce qu'elles le soient toutes.

4 Exercice 4

1. Sur le site *data.gouv.fr*, vous pourrez trouver des tables de données publiques en libre accès. Choisissez un thème qui vous intéresse, puis une table en conséquence. Télécharger là.
2. Les tables sont souvent au format *.xls* : vous aurez besoin d'installer un package pour pouvoir les lire. La commande pour ce faire est *install.packages("nom du package")*. Installer le package *gdata*.
3. Analyser votre table.

5 Exercice 5

1. Télécharger et installer le package *ISwR* (Introductory Statistics with R).
2. Utiliser la commande *summary* pour analyser rapidement la table *bp.obese*. L'échantillon provient d'un échantillon de population mexicaine en Californie, et la table décrit 3 variables : le sexe (femme = 1, homme =0), le ratio d'obésité (*obese*) et la pression sanguine systolique en *mm* de mercure (*bp*).
3. Représenter les données dans un graphe, en utilisant des symboles différents pour les hommes et les femmes.
4. Expliquer la pression sanguine en fonction du ration d'obésité, puis du ratio d'obésité et du sexe.
5. Tracer sur un même graphe les courbes correspondant aux régressions dans les 2 modèles. Soignez la présentation (couleurs différentes, légende,...)

6 Exercice 6

- Télécharger la librairie *MASS*.
- Analyser rapidement la table *cats* et afficher les variables les unes en fonctions des autres par paires.
- Effectuer une régression linéaire selon le modèle $Hwt \sim Bwt * Sex$. Cela apporte-t-il quelque-chose par rapport au modèle $Hwt \sim Bwt + Sex$?
- Visualiser les composantes de votre régression.
- En extraire les prédiction, les coefficients, les résidus, lés résidus studentisés, et la formule du modèle.
- Tracer le *qqplot* des résidus studentisés ainsi que la première bissectrice.
- Tracer le graphe des résidus contre les prédictions.
- Tracer le graphe des distance de Cook.
- Observer les attributs que vous donne *summary*
- Afficher le R^2 ajusté de la régression, le nombre de degrés de liberté résiduels, la matrice de variance-covariance des paramètres estimés.

7 Problème du voyageur de commerce et algorithme du recuit simulé

D'après le livre de Michel Benaïm et Nicole El Karoui, *Promenade aléatoire, Chaîne de Markov et simulations, martingales et stratégies*, exemple 3.1.8.

La méthode du recuit simulé est un algorithme d'optimisation proche de celui de Metropolis, et consiste à se promener aléatoirement sur l'espace (fini) des états d'un système selon une loi construite de façon à ce que la promenade converge vers un état qui minimise une certaine fonctionnelle.

On se donne une fonction $h :]0; \infty[\to]0; 1]$ telle que

$$h(x) = xh(\frac{1}{x}),$$

par exemple $min(1, x)$ ou bien $\frac{x}{1+x}$. La fonction $V : E \to \mathbb{R}_+$ est la fonction "coût" à minimiser.

La terminologie "recuit simulé" vient d'une technique métallurgique consistant à faire fondre de façon répétée le métal puis à le faire lentement refroidir pour en améliorer les propriétés. En effet, on va se donner un schéma de décroissance d'une quantité analogue à la température, que nous noterons T_n. Ce schéma est crucial pour que l'algorithme converge vers un minimum global de la fonction V et ne reste pas piégé dans un de ses minima locaux. Vous pourrez choisir l'un des schémas suivants :

- décroissance logarithmique :

$$T_n = \frac{C}{\log(n)}$$

- recuit par palier :

$$T_n = \frac{1}{k} \text{ pour } e^{(k-1)C} \leq n < e^{kC}$$

Voici l'algorithme :

Initialiser X_0. Choisir le nombre de pas de la marche aléatoire m. Pour n allant de 1 à $m-1$, répéter :

1. Choisir un voisin y de X_n aléatoirement.
2. Tirer $U \sim \mathcal{U}_{[0,1]}$.
3. Si $U < h(\exp(\frac{1}{T_n})(V(X_n) - V(y))\frac{N(y)}{N(X_n)})$, accepter $X_{n+1} = y$, sinon refuser i.e. $X_{n+1} = X_n$.

Le problème auquel nous allons appliquer cet algorithme est celui d'un commerçant devant visiter un ensemble fini $E = X_1, ..., X_N$ de villes, une et une seule fois. Pour minimiser son temps et son argent, il souhait trouver le chemin l qui minimise la distance, soit, avec nos notations :

$$V(l) = \sum_{j=1}^{N-1} d(X_{l(j)}, X_{l(j+1)}),$$

où l'on voit un chemin comme une permutation de l'ensemble E. Nous travaillerons avec des villes disposées aléatoirement dans le carré $[0;1] \times [0;1]$.

1. Créer une fonction qui calcule le coût d'un chemin donné l.
2. Créer une fonction qui affiche un chemin donné l.
3. Choisir une loi de transition sur les chemins, et l'implémenter.
4. Implémenter l'algorithme du recuit simulé sur ce problème, à l'aide d'une fonction si possible. Afficher le chemin obtenu, ainsi que l'évolution de la longueur du chemin en fonction du nombre d'itérations . Qu'en pensez vous ? De combien d'itérations avez-vous besoin pour obtenir un chemin plausible ?