FACULTÉ DES SCIENCES ECONOMIQUES

UNIVERSITÉ DE RENNES 1

M1 Statistique & Économétrie
Ewen Gallic [1]
http://egallic.fr

2015

Logiciel R et programmation

Exercices

Partie 5 : Régressions

Exercice 1 (Exploration rapide des données)
Cet exercice s'appuie sur un jeu de données de consommation de carburant de 392 véhicules. Il provient de la bibliothèque StatLib, maintenue à la Carnegie Mellon University [2].

1. Charger le jeu de données `Auto` contenu dans le *package* `ISLR`, puis regarder sa page d'aide ;

```
library(ISLR)
data("Auto")
?Auto
```

2. Afficher un résumé des différentes variables ;

```
summary(Auto)
```

3. En utilisant la fonction `stargazer()` contenue dans le *package* du même nom, aficher dans la console un tableau de statistiques descriptives en sortie texte ASCII. Prendre soin de limiter à deux le nombre de chiffres des décimales ;

```
library(stargazer)
stargazer(Auto, type = "text", digits = 2)
```

4. Exporter ce tableau dans un fichier `HTML`, en prenant soin d'ajouter le titre suivant : "*Statistiques descriptives*". De plus, changer le séparateur des décimales en une virgule au lieu d'un point ;

1. ewen.gallic[at]gmail.com
2. http://archive.ics.uci.edu/ml/datasets/Auto+MPG

```
stargazer(Auto, type = "html", title = "Statistiques descriptives",
    out = "stat_des.html", digits = 2, decimal.mark = ",")
```

5. Représenter par un nuage de points la relation entre les variables de puissance (`horsepower`) et de consommation (`mpg`), puis sur un autre graphique, la relation entre la masse du véhicule (`weight`) et sa consommation ;

```
library(ggplot2)
ggplot(data = Auto, aes(x = horsepower, y = mpg)) +
geom_point() +
xlab("") + ylab("") +
ggtitle("Consommation (miles par gallon) en fonction de la puissance (Ch)")

ggplot(data = Auto, aes(x = weight, y = mpg)) +
geom_point() +
xlab("") + ylab("") +
ggtitle("Consommation (miles par gallon) en fonction de la masse (livres)")
```

6. Reprendre le code du graphique représentant la consommation en fonction de la masse du véhicule, et faire dépendre la couleur des points du nombre de cylindres (le nombre de cylindres sera considéré comme une variable catégorielle). Puis, ajouter des courbes de tendance pour chaque catégorie de cylindres à l'aide de la fonction `stat_smooth()`. Ces courbes de tendance devront être estimées à l'aide d'une régression linéaire.

```
ggplot(data = Auto, aes(x = weight, y = mpg, colour = factor(cylinders))) +
stat_smooth(method="lm", fullrange=TRUE, alpha = 0.1, se = FALSE) +
geom_point() +
xlab("") + ylab("") +
ggtitle("Consommation (miles par gallon) en fonction de la masse (livres)")
```

7. Afficher un tableau des corrélations entre chaque variables numériques ;

```
library(dplyr)
cor(Auto %>% select(-name))
```

8. En utilisant la fonction `corrplot.mixed()` du *package* `corrplot`, réaliser une visualitation graphique de la matrice de corrélation.

```
library(corrplot)
library(RColorBrewer)
corrplot.mixed(cor(Auto %>% select(-name)),
         col = rev(brewer.pal(10, "Spectral")))
```

Exercice 2 (Régression linéaire)

Cet exercice s'appuie sur le même jeu de données que le précédent.

1. Préparer deux tableaux de données : l'un comprenant 80% des observations, et le second les 20% restantes. Les observations à conserver dans le tableau contenant 80% des observations doivent être tirées au hasard ;

```r
ind <- sample(seq_len(nrow(Auto)), size = 89/100*nrow(Auto))
auto_80 <- Auto[ind,]
auto_20 <- Auto[-ind,]
```

2. En prenant comme jeu de données la base avec 80% des observations, régresser la consommation (`mpg`) sur la puissance (`horsepower`), la masse (`weight`) et l'année de mise en circulation (`year`), en faisant appel à la fonction `lm()` ;

```r
reg <- lm(mpg ~ horsepower + weight + year, data = auto_80)
```

3. Afficher un résumé de l'estimation à l'aide de la fonction `summary`, puis extraire uniquement le tableau des coefficients ;

```r
summary(reg)
# Uniquement les coefficients
reg$coefficients
# Les coefficients estimés, l'écart-type
# la valeur du t de Student
# ainsi que la p-value associée au test de nullité du coefficient
summary(reg)$coefficients
```

4. Observer les graphiques retournés lorsque la fonction `plot()` est appliquée au résultat de l'estimation ;

```r
op <- par(mfrow = c(2,2))
plot(reg)
par(op)
```

5. Créer un tableau de données contenant les résidus de la régression, ainsi qu'une colonne indiquant le numéro des lignes de chaque observation (que l'on peut appeler `index` par exemple) ;

```r
library(dplyr)
residus <- residuals(reg)
residus_df <- data.frame(residus = residus) %>%
mutate(index = row_number())
```

6. Tracer les résidus à l'aide d'un nuage de points (les valeurs de la variable `index` seront représentées en abscisses). Puis, changer la représentation géométrique pour afficher un histogramme des résidus ;

```r
ggplot(data = residus_df, aes(x = index, y = residus)) +
geom_point()

ggplot(data = residus_df, aes(x = residus)) + geom_histogram()
```

7. Construire un intervalle de confiance à 95% pour chacun des coefficients de la régression. Pour un paramètre α, l'intervalle de confiance est donné par :

$$\widehat{\text{I.C.}_\alpha(1-p)} = \left[\hat{\alpha} \pm t_{p/2,n-m-1} \times \hat{\sigma}_{\hat{\alpha}}\right],$$

avec p le risque associé au test, n le nombre d'observations, m le nombre de variables explicatives et $t_{p/2,n-m-1}$ le quantile d'ordre $p/2$ de la Student à $n-m-1$ degrés de liberté.
Pour réaliser les intervalles de confiance, procéder comme suit :
— récupérer le tableau de coefficients issu du résumé de l'estimation, et le stocker dans un objet de type `data.frame` que l'on appellera `coeffs` ;
— récupérer ensuite le nombre de degrés de libertés associés au test de nullité d'un coefficient ;
— ajouter dans le tableau `coeffs` les variables `b_inf` et `b_sup`, qui correspondent respectivement aux bornes inférieures et supérieures de chaque intervalle.
Enfin, comparer les résultats obtenus avec ceux issus de l'application de la fonction `confint()` à l'objet de la régression ;

```r
coeffs <- summary(reg)$coef %>% data.frame()
colnames(coeffs) <- c("estimate", "std", "t_value", "p_value")
ddl <- reg$df.residual
coeffs %>%
  mutate(b_inf = estimate - qt(p = 0.975, df = ddl) * std,
         b_sup = estimate + qt(p = 0.975, df = ddl) * std)
confint(reg, level = 0.95)
```

8. Exporter les résultats de la régression dans un fichier `html`, en s'appuyant sur la fonction `stargazer()` ;

```r
stargazer(reg, type = "html", out = "regression.html", decimal.mark = ",")
```

9. En utilisant le modèle estimé et les données contenues dans la base contenant uniquement 20% des observations, effectuer des prévisions sur la consommation des véhicules et les comparer aux valeurs réelles ;

```r
previsions <- predict(reg, newdata = auto_20,
                      interval = "prediction", level = 0.95)
previsions <- data.frame(previsions)
previsions$mpg <- auto_20$mpg
```