TD - Introduction à l'analyse statistique dans l'environnement R

Bertrand Iooss
Polytech Nice Sophia

1 Bases de R

Installation

R peut être téléchargé depuis l'un des sites miroirs sur http://cran.r-project.org/mirrors.html.

Session R

Une fois R lancé, il y a une invite de commande (>), où vous pouvez entrer des nombres et effectuer des calculs.

Affectation de variables

Nous attribuons des valeurs aux variables avec l'opérateur d'affectation "=". Taper la variable seule à l'invite de commande affichera sa valeur. Il convient de noter qu'une autre forme d'opérateur d'affectation "<-" est également utilisée.

Fonctions

Les fonctions R sont appelées par leur nom, suivi de parenthèses contenant zéro ou plusieurs arguments. L'exemple suivant applique la fonction c pour combiner trois valeurs numériques dans un vecteur.

Commentaires

Tout texte après le symbole dièse "#" dans la même ligne est considéré comme un commentaire.

Librairie (package)

Parfois, nous avons besoin de fonctionnalités supplémentaires au-delà de celles offertes par la bibliothèque de base de R. Pour installer un package, vous devez utiliser la fonction `install.packages`.

Obtenir de l'aide

R fournit une documentation complète. Par exemple, taper `?c` ou `help(c)` ; cela donne la documentation de la fonction `c` dans R.

Manipulation de données

La fonction `c()` stocke les informations dans un vecteur. Un vecteur peut contenir des informations avec différents modes : numérique, logique (TRUE ou FALSE) ou caractère.

Une séquence régulière donne des nombres entiers, par exemple de 1 à 8 avec :

```
> pages = 1:8
> pages
[1] 1 2 3 4 5 6 7 8
```

Pour créer une matrice, on peut utiliser la fonction `matrix()`, et aussi `rbind()` et `cbind()` :

```
> x = cbind(1:3, 4:6)
```

Nous pouvons alors accéder aux éléments de la matrice `x` par :

```
> x[1,1]
> dim(x)
```

Le tableau (classe `array`) est une généralisation de la matrice à n dimensions.

La structure de données (classe `data.frame`) est un tableau de données composé d'un ou plusieurs vecteurs/facteurs de mêmes ou différents modes. Par exemple :

```
> listing = data.frame(name=c("Pierre","Paul","Jacques"), age=c(30,45,28),
eyes=c("vert","bleu","marron"))
```

Vous pouvez ensuite essayer :

```
> view(listing)
> mean(listing$age)
> table(listing$eyes)
> summary(listing)
```

Opérateurs et fonctions élémentaires

R dispose de 3 types d'opérateurs :

arithmétique	**comparaison**	**logique**
`+` addition	`<` plus petit	`!x` NON logique
`-` soustraction	`>` plus grand	`x & y` ET logique
`*` multiplication	`<=` inférieur ou égal	`x
`/` division	`>=` supérieur ou égal	`xor(x, y)` OU exclusif
`^` puissance	`==` égal	
`%%` modulo	`!=` différent	

`%/%` division entière `%*%` produit matriciel `%o%` produit de Kronecker

R dispose des fonctions élémentaires suivantes pour le traitement des vecteurs : `c(x,y)` (concaténation), `length(x)` (nombre d'éléments), `min(x)`, `max(x)`, `mean(x)`, `median(x)`, `var(x)`, `sd(x)`, `range(x)`, `rank(x)`, `sort(x)`, `order(x)`, `sum(x)`, `prod(x)`, `diff(x)`, `cumsum(x)`.

D'autres fonctions sont définies pour les matrices (`dim`, `nrow`, `ncol`, `diag`, etc.). Certaines fonctions sont génériques et s'appliquent à certains objets : `print()`, `plot()`, `summary()`.

Graphiques

Un graphique basique :

```
> x=seq(-2 * pi, 2 * pi, 0.1)
> plot(x, cos(x))
```

Pour personnaliser les graphiques, utilisez `par` :

```
> help(par)
```

Vous pouvez voir une démonstration des capacités graphiques de R en tapant :

```
> demo(graphics)
```

Séquences aléatoires

Toutes les distributions de probabilité sont disponibles dans R pour obtenir les valeurs de densité, la fonction de distribution, la fonction quantile ou pour générer des échantillons aléatoires. Ces fonctions suivent la forme suivante :

- `dfunc()` pour les valeurs de densité,
- `pfunc()` pour la fonction de distribution,
- `qfunc()` pour les quantiles,
- `rfunc()` pour la génération aléatoire,

où `func` indique le type de loi de probabilité.

Par exemple, pour générer des échantillons aléatoires avec différentes lois de probabilité :

```
> rnorm(100, mean=0, sd=1)     # Loi normale (Gauss)
> runif(100, min=0, max=1)     # Loi uniforme
> rexp(100, rate=1)            # Loi exponentielle
> rlnorm(100, meanlog=0, sdlog=1)   # Loi log-normale
```

Pour tracer la densité de la loi normale entre -5 et 5 :

```
> x = seq(-5, 5, 0.1)
> plot(x, dnorm(x), type="l")
```

2 Analyse univariée

R dispose de nombreux jeux de données d'exemple. Pour voir la liste de ces jeux de données :

```
> data()
```

Dans cet exercice, nous utilisons le dataframe « airquality », qui contient des mesures de la qualité de l'air de New-York en 1973. Tapez :

```
> ?airquality
```

Ensuite, nous renommons ce dataframe pour faciliter l'analyse :

```
> a = airquality
```

La liste des variables d'un dataframe est obtenue avec la commande `names()` :

```
> names(a)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

Et la dimension (nombre d'échantillons et nombre de variables) avec `dim()` :

```
> dim(a)
[1] 153   6
```

2.1 Résumé numérique

Pour obtenir une « synthèse numérique » d'un dataframe, R a la commande `summary()` :

```
> summary(a)
```

Question : Observez-vous des caractéristiques particulières dans ces données ?

2.2 Diagrammes en boîte

Le boxplot (`boxplot()`) résume un échantillon avec ses valeurs extrêmes, ses quantiles et sa médiane :

```
> boxplot(a)
```

Une visualisation plus fine peut être obtenue avec `sapply()`, qui applique une commande à chaque colonne d'une matrice :

```
> x11()
> par(mfrow=c(2,3))
> sapply(a, boxplot)
```

Pour ajouter des légendes, une boucle classique est plus pratique :

```
> names = dimnames(a)[[2]]
> x11()
> par(mfrow=c(2,3))
> for (i in 1:dim(a)[2]) boxplot(a[,i], main=names[i])
```

2.3 Histogrammes

La commande `hist()` trace l'histogramme des fréquences, par exemple :

```
> x11()
> hist(a$Wind)
```

Exercice : Changez le filtre de l'axe des abscisses (le nombre de classes de l'histogramme).

L'histogramme avec des probabilités en ordonnées s'obtient avec l'option `prob=TRUE`. Une approximation de la densité (méthode de noyau) peut être ajoutée :

```
> x11()
> hist(a$Wind, prob=TRUE)
> lines(density(a$Wind))
```

Exercice : Tracez tous les histogrammes sur une même figure.

2.4 Calcul des quantiles

Pour calculer les quantiles, nous utilisons la commande `quantile()`. Par défaut, elle calcule les quantiles à 25 %, 50 % (médiane), 75 %, ainsi que les minimums et maximums. Cependant, l'option `probs` permet de spécifier d'autres quantiles :

```
> quantile(a$Wind)
0%  25%  50%  75%  100%
1.7  7.4  9.7 11.5 20.7
```

Un intervalle de variation classique est basé sur les quantiles 5 % et 95 % :

```
> quantile(a$Wind, probs=c(0.05, 0.95))
5%  95%
4.6  15.5
```

Pour vérifier si un échantillon suit une loi normale, R dispose de la commande `qqnorm()`. L'abscisse représente les quantiles théoriques de la loi normale, et l'ordonnée les quantiles de l'échantillon.

Exercice : Tracez le graphique quantile-quantile de la variable « Wind » par rapport à la loi normale, ainsi que celui de la variable « Ozone » par rapport aux lois normale et log-normale.

2.5 Ajustement à une loi normale pour la variable « Wind »

Exercice : Estimez les paramètres pour ajuster la distribution de la variable « Wind » à une loi normale. Premièrement, utilisez la méthode des moments ; ensuite, utilisez la méthode du maximum de vraisemblance (fonction `fitdistr()` du package `MASS`). Enfin, appliquez un test de Kolmogorov-Smirnov (`ks.test()`) pour rejeter (ou non) l'hypothèse d'adéquation à une loi normale.

---

3 Analyse bivariée

Pour obtenir des nuages de points (scatterplots) :

```
> pairs(a, panel=panel.smooth)
```

Question : Que pouvez-vous dire sur les relations entre les variables ?

Les matrices de variance/covariance et de corrélation du dataframe peuvent être obtenues avec les commandes `var()` et `cor()`.

```
> var(a)
> var(a, na.rm=TRUE)
> cor(a, use="complete.obs")
```

Question : Avez-vous des confirmations sur votre intuition concernant les relations entre les variables ?

---

4 Simulation de nouvelles données

4.1 Simulation de valeurs de radiation solaire, température et vitesse du vent

Exercice : Simulez un échantillon de taille 1000 pour le vecteur (Solar.R, Wind, Temp) sans prendre en compte les corrélations, en suivant une loi uniforme pour Solar.R, une loi normale pour Wind, et une loi normale pour Temp. Visualisez les histogrammes et les nuages de points, puis comparez-les avec ceux des données de qualité de l'air initiales.

Remarque : Pour Wind et Temp, il serait pertinent de simuler des lois tronquées pour obtenir des valeurs réalistes.

4.2 Simulation avec corrélation (loi normale multivariée)

Le package `MASS` contient la fonction `mvrnorm()` qui permet de simuler des échantillons à partir d'une loi normale multivariée.

```
> library(MASS)
> help(mvrnorm)
```

```
> var(a, na.rm=TRUE)
> covar = matrix(c(12.657324, -16.857166, -16.857166, 90.820311), nrow=2, ncol=2)
> x2 = matrix(0, nrow=1000, ncol=3, dimnames=list(1:1000, names[2:4]))
> x2[,1] = runif(1000, min=7, max=334)
> x2[,2:3] = mvrnorm(1000, mu=c(mean(a$Wind), mean(a$Temp)), Sigma=covar)
> summary(x2)
> x3 = x2[x2[,2] > 0, ]
> summary(x3)
> x11()
> pairs(x2, panel=panel.smooth)
> x11()
> pairs(a[,2:4], panel=panel.smooth)
> cor(x2)
> cor(a[,2:4], use="complete.obs")
```

---

5 Construction d'un modèle explicatif pour l'ozone

Nous créons une nouvelle matrice en supprimant les lignes avec des valeurs manquantes :

```
> b = a[!is.na(a[,1]) & !is.na(a[,2]),]
```

5.1 Régression linéaire

Effectuons une régression linéaire (`lm()`) entre l'ozone et les 5 autres variables explicatives :

```
> formule = as.formula(b$Ozone ~ b$Solar.R + b$Wind + b$Temp + b$Month + b$Day)
> m1 = lm(formule, data=data.frame(x=b[,2:6], y=b$Ozone))
> m1
> names(m1)
```

Analyse graphique des résultats de la régression linéaire :

```
> x11()
> y = predict(m1, as.data.frame(b[,2:6]))
> plot(y, b$Ozone, xlab="prediction", ylab="observation")
> lines(y, y)
```

La fonction `plot()` fournit des graphiques des résultats :

```
> plot(m1)
```

Quelques indicateurs quantitatifs, tels que le coefficient de détermination R^2, les t-values, et la table d'analyse de la variance :

```
> summary(m1)
> anova(m1)
```

Pour connaître la contribution de chaque variable dans la variance, nous utilisons la régression standardisée (SRC) :

```
> c = m1$coefficients
> src = rep(0, 5)
> src = matrix(c(src), ncol=5, dimnames=list(c("SRC"), names[2:6]))
> for (i in 1:5) src[i] = c[i+1] * sd(b[,i+1]) / sd(b$Ozone)
> print(src)
> print(src^2)
```

5.2 Amélioration du modèle

Le modèle n'est pas suffisamment prédictif, donc nous ajoutons des termes d'interaction et des termes quadratiques.

```
> formule = as.formula(b$Ozone ~ (b$Solar.R + b$Wind + b$Temp + b$Month)^2
+ I(b$Solar.R^2) + I(b$Wind^2) + I(b$Temp^2) + I(b$Month^2))
> m2 = lm(formule, data=data.frame(x=b[,2:6], y=b$Ozone))
> summary(m2)
```

Pour simplifier le modèle, nous appliquons une procédure de « stepwise » :

```
> m3 = step(m2, trace=0)
> summary(m3)
> x11()
> y = predict(m3, as.data.frame(b[,2:6]))
> plot(y, b$Ozone, xlab="prediction", ylab="observation")
> lines(y, y)
```

---

6 Simulations avec le modèle prédictif

Exercice : En utilisant le modèle de second ordre (m3) et les simulations des variables explicatives du paragraphe 4.2 (matrice x3), simulez la distribution de l'ozone. Comparez-la à celle des observations d'ozone. Pour la variable « Month », utilisez la fonction `sample()` avec l'option `replace=TRUE` pour échantillonner aléatoirement les valeurs `{5,6,7,8,9}`. Ensuite, donnez un intervalle de prédiction [5 %, 95 %] et comparez-le à l'intervalle de la variable observée.