Année : 2008-2009 1er semestre
Niveau : MASTER IS 1ère année
Cours : Logiciel R
Enseignant : A. Illig

TP3 : Représentations graphiques en R et estimation d'une densité

Table des matières

1 Les graphiques avec R

R est un logiciel qui offre une multitude de possibilités graphiques qu'il est impossible de détailler dans ce document (`demo(graphics)` permet d'obtenir un avant-goût de ces possibilités). Cependant, les paragraphes suivants vous permettrons d'acquérir les bases nécessaires à l'utilisation de graphiques en R.

R fonctionne au moyen de deux types de fonctions graphiques :
- les fonctions graphiques principales ou de "haut niveau" qui créent un nouveau graphique,
- les fonctions graphiques secondaires ou de "bas niveau" qui permettent d'ajouter de nouveaux éléments sur un graphique déjà produit ou de prédéfinir les paramètres graphiques d'un graphique à produire.

1.1 Les fonctions graphiques de "haut niveau"

Le tableau Tab. 1 fournit une liste non exhaustive des fonctions graphiques principales.

Fonction graphique	Description
`plot(X)`	Graphe des valeurs de `X` en fonction de leur indice
`plot(X,Y)`	Graphe des valeurs de `Y` en fonction des valeurs de `X`
`matplot(X,Y)`	Graphe de la 1ère colonne de `Y` en fonction de la 1ère colonne de `X`, graphe de la 2ème colonne de `Y` en fonction de la 2ème colonne de `X` ...
`pairs(X)`	Si `X` est une matrice ou un data.frame, `pairs` crée tous les graphes bivariés entre les colonnes de `X`
`contour(X,Y,Z)`	Courbes de niveau (`X` et `Y` sont les vecteurs de localisation dans le plan, `Z` est la matrice des niveaux de dimension `c(length(X),length(Y))`
`pie(X)`	Diagramme circulaire de `X`
`boxplot(X)`	Boîte à moustaches de `X`
`dotchart(X)`	Graphe de Cleveland de `X`
`hist(X)`	Histogramme des fréquences de `X`
`barplot(X)`	Diagramme en bâtons de `X`
`plot.ts(X)`	Graphe de la série temporelle `X`
`qqplot(X,Y)`	Graphe quantile-quantile des échantillons `X` et `Y`

Tab. 1 – Fonctions graphiques de haut-niveau

Chaque fonction possède des options graphiques dont certaines sont communes à plusieurs fonctions. Le tableau Tab. 2 liste un certain nombre d'options graphiques.

Option graphique (valeur par défaut)	Description
`add=FALSE`	Si `add=TRUE`, superpose le graphe au graphe précédent
`axes=TRUE`	Si `axes=FALSE`, ne trace pas les axes ni le cadre
`type`	Type de graphique (''`p`'' points, ''`l`'' lignes, ''`b`'' lignes et points, ''`h`'' lignes verticales, ''`s`'' et ''`S`'' escaliers...)
`xlim, ylim`	Limites inférieure et supérieure des axes
`xlab, ylab`	Annotations des axes
`main`	Titre du graphique

TAB. 2 – Options graphiques

1.2 Fonctions graphiques de "bas niveau"

Plusieurs fonctions graphiques secondaires de R permettent d'ajouter de nouveaux éléments à un graphique déjà existant. Le tableau TAB 3. recense un grand nombre de ces fonctions.

Fonction graphique secondaire	Description
`points(X,Y)`	Ajoute les points de coordonnées définies par les vecteurs `X` et `Y`
`lines(X,Y)`	Analogue de `points(X,Y)` avec des lignes
`text(x,y,labels)`	Ajoute le texte défini par `labels` au point `(x,y)`
`legend(x,y,legend)`	Inscrit la légende donnée par `legend` au point `(x,y)`
`title(main)`	Ajoute le titre donné par `main`
`abline(b,a)`	Trace une ligne d'équation `y=ax+b`
`abline(h=b)`	Trace une ligne d'équation `y=b`
`abline(v=k)`	Trace une ligne d'équation `x=k`
`rug(X)`	Dessine des traits verticaux aux points d'abscisses données par `X`

TAB. 3 – Fonctions graphiques secondaires permettant l'ajout d'éléments graphiques

Remarque : Il est possible de rajouter sur un graphique, au point de coordonnées *(x,y)*, une expression mathématique au moyen de la fonction `text` :

```
text(x,y,expression(...))
```

Si l'on souhaite inclure dans une expression mathématique la valeur numérique d'une variable `v`, on peut faire appel aux fonctions `substitute` et `as.expression` sous la forme suivante :

```
text(x,y,as.expression(substitute(label==value,list(value=v))))
```

Aux fonctions graphiques secondaires s'ajoutent les paramètres graphiques. Ils peuvent être utilisés sous forme d'options graphiques pour certains ou bien à l'aide de la fonction `par()`. La liste de ces paramètres graphiques est obtenue par la commande `help(par)`. Le tableau TAB. 4 dresse la liste des paramètres que nous serons amenés à utiliser en TP.

Paramètre	Description
`bg`	Spécifie la couleur du fond graphique (ex : `bg=''lightyellow''`)
`font`	Contrôle le style du texte (1 : normal, 2 : italique, 3 : gras, 4 : gras italique)
`pch`	Contrôle le type de symbole (ex : `pch=25`)
`col`	Contrôle la couleur des symboles
`lwd`	Gère la taille des lignes
`xaxt, yaxt`	Si `xaxt=''n''` (`yaxt=''n''`) l'axe des abscisses (ordonnées) n'est pas tracé
`new`	Si `new=TRUE`, le nouveau graphique sera tracé sur l'ancien graphique

TAB. 4 – Paramètres graphiques

1.3 La gestion des graphiques

Un dispositif graphique s'ouvre lors de l'éxécution de la première fonction graphique. L'appel à une autre fonction graphique remplace, dans ce même dispositif graphique, l'ancien graphique par le graphique courant. Pour visualiser simultanément les deux représentations graphiques, deux approches sont possibles :

- La commande `par(new=TRUE)`, intercalée entre les deux commandes graphiques, permet de superposer le nouveau graphique à l'ancien dans le dispositif graphique.
- Il est aussi possible d'ouvrir un nouveau dispositif graphique grâce à la commande `X11()` sous Linux ou (`windows()` sous MS Windows) avant de taper la deuxième commande graphique. On visualise alors deux graphiques chacun dans une fenêtre graphique différente.
- Une autre possibilité consiste à afficher plusieurs graphiques dans un même dispositif graphique au moyen de la commande `layout` qui partitionne le dispositif en plusieurs parties où s'affichent successivement les différents graphiques. Par exemple,

```
> # Dispositif graphique à 6 fenetres
> layout(matrix(1:6,3,2))
> layout.show(6)
```

Enfin, pour sauvegarder un graphique dans un fichier *eps* (Encapsulated PostScript), il suffit de taper la commande `dev.copy2eps(file=''Plot.eps'')`.

1.4 Une illustration graphique

```
> # Tracé de la série chronologique Nile dans un dispositif graphique
> plot.ts(Nile)
> # Titre oublié?
> title("Un petit titre")
> # Tracé de la série chronologique AirPassengers dans le même dispositif
> plot(AirPassengers, lwd=3, col=''skyblue2'', col.main=''skyblue4'',
+ main="Une jolie serie chronologique", font.main=4, font=2)
> # Données cars attachées
> attach(cars)
> # Ouverture d'un autre dispositif graphique
> x11()
> # Utilisation de la fonction pairs
> pairs(cars)
> # Ouverture d'un autre dispositif graphique
> x11()
> # Partition du nouveau dispositif
> layout(matrix(1:4,2,2),width=c(4,1),height=c(2,2))
> # Graphiques de Cleveland et boîtes à moustaches des données cars
> dotchart(speed, main="Cleveland speed")
> text(5,20,as.expression(substitute(min==value1,list(value1=min(speed)))))
> text(5,18,as.expression(substitute(MAX==value2,list(value2=max(speed)))))
> dotchart(dist, main="Cleveland dist")
> boxplot(speed,main="Boxplot speed")
> boxplot(dist, main"Boxplot dist")
> # Ouverture d'un autre dispositif graphique
> x11()
> # Histogrammes des données speed
> par(bg="lightgreen")
> layout(matrix(1:2,1,2))
> barplot(speed, main="Diagramme en batons des données speed")
> hist(speed, main="Histogramme des donnees speed",col="tomato")
> rug(speed)
> # Sauvegarde de ce joli graphique
> dev.copy2eps(file="Histogramme.eps")
> # Détachons les données cars
> detach(cars)
```

2 Zoom sur l'histogramme

2.1 Statistique descriptive

Pour une série statistique d'observations $(x_j)_{j=1\dots n}$ (d'un caractère généralement continu) regroupées en k classes $[a_i, a_{i+1}[$ d'effectif n_i, l'histogramme est un bon moyen de visualiser la répartition des données. Un histogramme est formé de rectangles accolés de bases $[a_i, a_{i+1}[$ et de hauteurs ω_i proportionnelle à la fréquence $f_i = \frac{n_i}{n}$ si les classes ont la même amplitude et de hauteur proportionnelle à $f_i/(a_{i+1} - a_i)$ sinon. On remarque qu'en choisissant,

$$w_i = \frac{f_i}{a_{i+1} - a_i} \; \forall i = 1 \dots k \tag{1}$$

la somme des aires des rectangles est égale à 1 :

$$\sum_{i=1}^{k} \frac{f_i}{a_{i+1} - a_i} \times (a_{i+1} - a_i) = \sum_{i=1}^{k} f_i = 1.$$

En R, la commande `hist` permet de tracer l'histogramme de données contenues dans un vecteur avec les options classiques `main`, `xlim`, `xlab`, `ylab` ... et les options spécifiques :

- `breaks` : vecteur des délimitations entre classes (par défaut toutes les classes ont même amplitude).
- `freq` : si `freq=FALSE`, l'aire sous le graphe est égale à 1 (par défaut `freq=TRUE` et la hauteur d'un rectangle est égale à l'effectif de la classe).
- `nclass` : nombre de classes.
- `labels` : permet d'associer un nom à chaque rectangle.

Par ailleurs, la commande `h=hist` permet aussi de conserver les caractéristiques de l'histogramme dans la liste `h`. Précisons certaines de ces caractéristiques :

- `breaks` : $n + 1$ points de frontière entre classes.
- `counts` : vecteur des effectifs n_i.
- `mids` : vecteur des centres des classes.
- `density` : hauteur des rectangles pour l'estimation de densité.

Par exemple, pour afficher les centres de classe, il suffit de taper la commande `h$mids`.

2.2 Estimation d'une densité

Si maintenant $(x_j)_{j=1\dots n}$ est un échantillon d'observations d'une variable aléatoire continue X de densité f, l'histogramme de la série $(x_j)_{j=1\dots n}$ fournit une estimation de la densité f en chaque point x :

$$(\hat{f}_n(x))_{obs} = \sum_{i=1}^{k} \omega_i 1_{[a_i, a_{i+1}[}(x)$$

avec ω_i défini par (1). En particulier, la probabilité

$$\mathbb{P}(X \in [a_i, a_{i+1}[) = \int_{a_i}^{a_{i+1}} f(x)\, dx$$

est approximée par

$$\int_{a_i}^{a_{i+1}} (\hat{f}_n(x))_{obs}\, dx = \omega_i(a_{i+1} - a_i) = f_i.$$

Autrement dit, la probabilité $\mathbb{P}(X \in [a_i, a_{i+1}[)$ est estimée par la proportion d'observations situées dans la classe $[a_i, a_{i+1}[$.

Interprétation probabiliste : Soit $(X_j)_{j=1\ldots n}$ un échantillon de même loi que X dont les valeurs observées correpondent à la série $(x_j)_{j=1\ldots n}$. La proportion de variables de l'échantillon appartenant à la classe $[a_i, a_{i+1}[$ est :

$$\pi_i = \frac{1}{n} \sum_{j=1}^{n} 1_{[a_i, a_{i+1}[}(X_j)$$

dont la valeur observée est $(\pi_i)_{obs} = f_i$. L'estimateur de la densité s'écrit :

$$\hat{f}_n(x) = \sum_{i=1}^{k} W_i 1_{[a_i, a_{i+1}[}(x)$$

avec $W_i = \pi_i/(a_{i+1} - a_i)$ et $(W_i)_{obs} = (\pi_i)_{obs}/(a_{i+1} - a_i) = \omega_i$.

Défaut de l'estimateur :
- il dépend de la partition en classes $([a_i, a_{i+1}[)_{i=1\ldots k}$,
- a_1 et a_{k+1} ne peuvent être infinis mais doivent pourtant être proches du support de f éventuellement égal à $\mathbb{R}$,
- k et $(a_i)_{i=1\ldots k}$ doivent dépendre de n pour que $\hat{f}_n$ converge vers f quand n tend vers l'infini.

Autres estimateurs : Partant de la définition de la densité en tant que dérivée de la fonction de répartition $f(x) = \frac{dF}{dx}(x)$, on peut définir :

$$\hat{f}_n(x) = \frac{\hat{F}_n(x + \delta) - \hat{F}_n(x - \delta)}{2\delta}$$

où $\hat{F}_n$ est la fonction de répartition empirique de l'échantillon :

$$\hat{F}_n(x) = \frac{1}{n} \sum_{j=1}^{n} 1_{]-\infty, x]}(X_j).$$

Finalement,

$$\begin{aligned}\hat{f}_n(x) &= \frac{1}{2\delta n}\sum_{j=1}^{n}\left(1_{\{X_i\leq x+\delta\}} - 1_{\{X_i<x-\delta\}}\right)\\ &= \frac{1}{\delta n}\sum_{j=1}^{n}\frac{1}{2}1_{[-1,1]}\left(\frac{x-X_i}{\delta}\right).\end{aligned}$$

Plus généralement, on peut remplacer le noyau uniforme $\frac{1}{2}1_{[-1,1]}(\cdot)$ par une densité de probabilité K plus lisse :

$$\hat{f}_n(x) = \frac{1}{\delta n}\sum_{j=1}^{n} K\left(\frac{x-X_i}{\delta}\right).$$

En R, la fonction `density` permet de construire l'estimateur à noyau de la densité d'une série d'observations avec en option le choix du noyau `kernel` et la taille de la fenêtre `bw` :

```
> attach(cars)
> hist(dist, nclass=20,freq=FALSE)
> h=hist(dist,nclass=20,freq=FALSE,plot=FALSE)
> est1=density(dist,kernel=''rectangular'', n= length(h$mids),
 from=min(h$mids), to=max(h$mids))
> est2=density(dist,kernel=''gaussian'', n= length(h$mids),
 from=min(h$mids), to=max(h$mids))
> plot(h$mids,h$density,type=''l'',col=3,xlab='' '', ylab='' '', yaxt=''n'',
xlim=range(h$breaks))
> par(new=TRUE)
> plot(est1$x,est1$y,type=''l'',col=4,xlab='' '',ylab='' '', yaxt=''n'',
xlim=range(h$breaks))
> par(new=TRUE)
> plot(est2$x,est2$y,type=''l'',col=5, xlab='' '',ylab='' '', yaxt=''n'',
xlim=range(h$breaks))
```

3 Exercices

3.1 Exercice 1

1. Charger les données `trees` et afficher les noms des différentes variables. Tracer sur un même graphique tous les graphes bivariés entre les variables de `trees`.
2. Semble-t-il y avoir une relation linéaire entre les variables `Volume` et `Girth` ? entre les variables `log(Volume)` et `log(Girth)` ? Si oui, calculer la droite de régression entre ces deux variables à l'aide de la fonction `lm`. Sur un même graphique, représenter le nuage de points et la droite de régression à l'aide de la fonction `abline`.
3. De la relation $V = \pi R^2 H$ donnant le volume d'un cylindre en fonction de sa hauteur et du rayon de sa base, ne peut-on pas songer à une relation linéaire reliant les variables `log(Volume)`, `log(Girth)` et `log(Height)` ? Enregistrer les résultats de la première régression entre les variables `log(Volume)` et `log(Girth)` dans la liste `lm1`. Puis, utiliser la fonction `update` pour effectuer la nouvelle regression et enregistrer le résultat dans la liste `lm2`.
4. Pour la variable `Height`, tracer dans un dispositif à deux fenêtres l'histogramme et le graphique de Cleveland.
5. Toujours pour la variable `Height`, créer un histogramme à trois classes `"petit"`, `"moyen"`, `"grand"` et afficher sous l'histogramme les données en faisant appel à la fonction `rug`.

3.2 Exercice 2

1. A l'aide de la fonction `rnorm`, générer un vecteur `X` de réalisations d'une loi $\mathcal{N}(11, 1)$ de taille 100. Calculer la moyenne et la variance empiriques de `X`.
2. Dans un premier dispositif graphique, tracer l'histogramme de `X` pour 10 classes, en vert sur fond bleu et avec un joli titre en italique. Enregistrer dans une liste `h` les caractéristiques de l'histogramme.
3. Calculer les valeurs de l'estimateur à noyau uniforme de `X` aux points `h$mids` et les enregistrer dans une liste `est1`. Faire de même pour l'estimateur à noyau gaussien et une liste `est2`.
4. Créer un vecteur `x` composés des réels allant de `min(h$mids)` à `max(h$mids)` et distants de 0.1 à l'aide de la fonction `seq`. Créer un vecteur `y` comportant les images de `x` par la densité d'un loi $\mathcal{N}(11, 1)$ (voir la fonction `dnorm`).
5. Ouvrir un nouveau dispositif graphique et représenter simultanément la densité d'une loi $\mathcal{N}(11, 1)$, l'estimation par histogramme, l'estimation par noyau uniforme et l'estimation par noyau gaussien de la densité de l'échantillon. Commenter le graphique obtenu.