Niveau : M1 IS
Cours : Introduction au logiciel R
Enseignant : Aude Illig

Université Versailles Saint-Quentin
Année 2014/2015 - 1er semestre

TP 3 : Graphiques avec R

**** Manipulation des objets* `matrix` *et* `data.frame` *****

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 des 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 des 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`
`qqplot(X,Y)`	Graphe quantile-quantile des échantillons `X` et `Y`

Table 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="p"`	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, sub`	Titre et sous-titre du graphique

Table 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, ...)` `text(locator(1),labels, ...)`	Ajoute le texte défini par `labels` au point `(x,y)` Analogue avec un choix de la localisation à l'aide de la souris
`legend(x,y,legend, ...)`	Inscrit la légende donnée par `legend` au point `(x,y)`
`title(main,sub)`	Ajoute le titre `main`, le sous-titre `sub`

`abline(b,a)`
`abline(h=b)`
`abline(v=k)`
	Trace une ligne d'équation `y=ax+b`
	Trace une ligne d'équation `y=b`
	Trace une ligne d'équation `x=k`

`rug(x)`	Dessine des traits verticaux aux points d'abscisses données par `x`

Table 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 ou des lignes
`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

Table 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, trois 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 `windows()` sous Windows (`x11()` sous Linux, `quartz()` sous OS X ou `dev.new()` le cas échéant) 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,

```
> layout(matrix(1:6,3,2))     # Création d'un dispositif graphique à 6 fenêtres
> layout.show(6)              # Affichage de l'organisation du dispostif
```

Enfin, pour sauvegarder un graphique dans un fichier *Plot.eps* (Encapsulated PostScript) ou *Plot.pdf* (Portable Document Format), il convient de taper la commande `dev.copy2eps(file="Plot.eps")` ou `dev.copy2pdf(file="Plot.pdf")`.

1.4 Une illustration graphique

Le code suivant permet de réaliser différents graphiques dont celui de la Fig 1.

```
> par(mfrow=c(1,2))
> # Tracé de l'objet Nile de type ts
> plot(Nile)
> title(main="Un petit titre", sub="Un sous-titre")
> # Tracé de l'objet AirPassengers de type ts  dans le même dispositif
> plot(AirPassengers,lwd=3,type="b",col="skyblue2",col.main=3,main="Joli !",font.main=2)
> # Tracé de l'objet volcano de type matrix
> par(mfrow=c(1,2))
> x=10*1:87
> y=10*1:61
> image(x,y,volcano,col=terrain.colors(20))
> contour(x,y,volcano,col=terrain.colors(15))
> windows()
> persp(x,y,volcano,phi=40,theta=30,expand=0.75,col="lightgreen")
> # Etude de l'objet cars de type data.frame
> names(cars)             # Affiche le nom des variables du data.frame
> row.names(cars)         # Affiche le nom des lignes du data.frame
> cars$speed              # Affiche les données de la première variable
> cars[,1]                # Affichage identique par syntaxe matricielle
> attach(cars)            # Accès aux variables par les noms speed et dist
> dev.new()               # Ouverture d'un nouveau dispositif graphique
> pairs(cars)             # Tracé des nuages de points
> matplot(cars,type="l")  # Tracé des courbes sur un même graphique
> windows()      # Ouverture d'un 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(5,16,as.expression(substitute(MAX==value2,list(value2=max(speed)))))
```

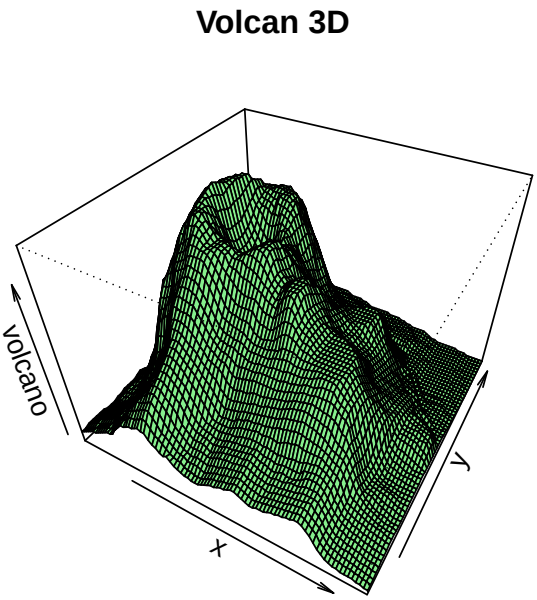

FIGURE 1 – Graphique 3D des données `volcano`

```
> dotchart(dist, main="Cleveland dist")
> boxplot(speed,main="Boxplot speed")
> boxplot(sort(dist), main="Boxplot dist")
> # Histogrammes des données speed
> x11()
> par(bg="lightgreen",mfrow=c(1,2))
> barplot(table(speed), main="Diagramme en bâtons",col=rainbow(10))
> hist(speed, main="Histogramme",col="tomato")
> rug(speed)
> dev.copy2pdf(file="Histogramme.pdf") # Sauvegarde du dernier graphique
> detach(cars)                         # Détachons les données cars
```

2 Zoom sur certaines commandes graphiques

2.1 Diagramme en bâtons

Pour une série statistique d'observations $(x_j)_{j=1...n}$ (généralement x_j est l'effectif de la $j$ème valeur d'un caractère discret), le diagramme en bâtons est formé de n bâtons (ou de barres si l'on donne une largeur aux bâtons) où la hauteur du j ème bâton est égale à x_j.

La commande R pour obtenir un diagramme en bâtons est `barplot(height)` où `height` est le

vecteur ou la matrice de données. Sont disponibles, en plus des options classiques `main`, `xlim`, `xlab`, `ylab` ..., les options suivantes :

- `width` : vecteur de largeurs de barres.
- `space` : espacement inter-barres (ou inter-blocs si `height` est une matrice) exprimé sous forme de fraction de la largeur moyenne des barres.
- `names.arg` : vecteur des noms à inscrire sous les barres.
- `beside` : si `beside=FALSE` les barres sont espacées, si `beside=TRUE` les barres sont juxtaposées
- `col` : un vecteur de couleurs pour les barres.
- `horiz` : si `horiz=FALSE` les barres sont verticales, si `horiz=TRUE` les barres sont horizontales.

Par exemple, les données `VADeaths` de type matrix, représentant le nombre de morts pour 1000 personnes dans plusieurs sous-populations de Virgine en 1940, peuvent être représentées par diagrammes en bâtons.

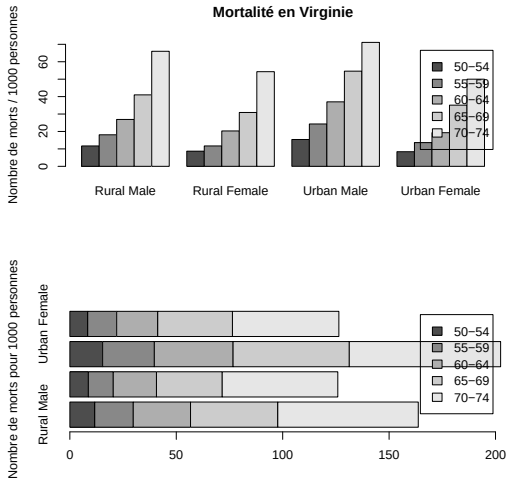

FIGURE 2 – Barplots des données `VADeaths`

Le code produisant la figure FIG. 2 est le suivant :

```
> VADeaths
      Rural Male Rural Female Urban Male Urban Female
50-54       11.7          8.7       15.4          8.4
55-59       18.1         11.7       24.3         13.6
60-64       26.9         20.3       37.0         19.3
65-69       41.0         30.9       54.6         35.1
70-74       66.0         54.3       71.1         50.0
> layout(c(1,2))
> barplot(VADeaths, beside=TRUE,legend=TRUE,main="Mortalité en
+ virginie",ylab="Nombre de morts / 1000 personnes")
> barplot(VADeaths,beside=FALSE,horiz=TRUE,legend=TRUE,ylab=
+ "Nombre de morts pour 1000 personnes")
```

2.2 Diagramme circulaire

Pour une série statistique d'observations $(x_j)_{j=1\ldots n}$ (généralement x_j est l'effectif de la $j$ème valeur d'un caractère discret), le diagramme circulaire est un disque divisé en n secteurs où l'angle (et donc la surface) du j ème secteur est proportionnel à x_j.

Si une population est subdivisée en sous-groupes dont la répartition en pourcentages est `prop` et les noms des sous-groupes sont `noms`, la commande R `pie(prop,noms)` permet de tracer le camembert de la population. Hormis les options classiques, on peut préciser l'angle (en degrés) de départ avec l'option `init.angle` et la couleur de chaque secteur avec l'option `col`.

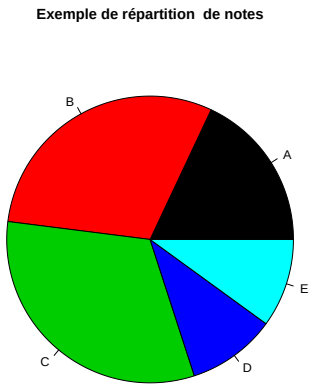

FIGURE 3 – Exemple de diagramme circulaire

La figure FIG. 3 est le résulats des commandes suivantes :

```
> prop=c(18,30,32,10,10)
> noms=c("A","B","C","D","E","F")
> pie(prop,noms,col=1:6, main="Répartition des notes d'une classe
+ d'étudiants en statistique)
```

2.3 Boîtes à moustaches

Pour des données brutes $(x_j)_{j=1\ldots n}$, observations d'un caractère continu, la boîte à moustaches est une alternative à l'histogramme permettant de se faire rapidement une idée de la répartition d'un ensemble de données :

- La partie principale d'une boîte à moustaches est un rectangle horizontal dont la borne inférieure correspond au premier quartile Q_1 des données et la partie supérieure au troisième quartile Q_3. Ainsi 50% des données sont situées dans la boîte et autant à l'extérieur.
- La ligne verticale située à l'intérieur du rectangle est la médiane des données.
- Les pattes de la boîtes sont fixées à la plus petite donnée supérieure à $Q_1 - 1.5(Q_3 - Q_1)$ pour la patte inférieure et la plus grande donnée inférieure à $Q_3 + 1.5(Q_3 - Q_1)$ pour la patte supérieure.
- Les cercles au-delà des pattes correspondent aux données restantes souvant qualifiées d'aberrantes.

La commande R qui trace la boîte à moustaches est `boxplot`. Les données `iris` de type `data.frame`, correspondant aux mesures effectuées sur 50 fleurs pour trois espèces d'iris, fournissent un bon exemple d'application.

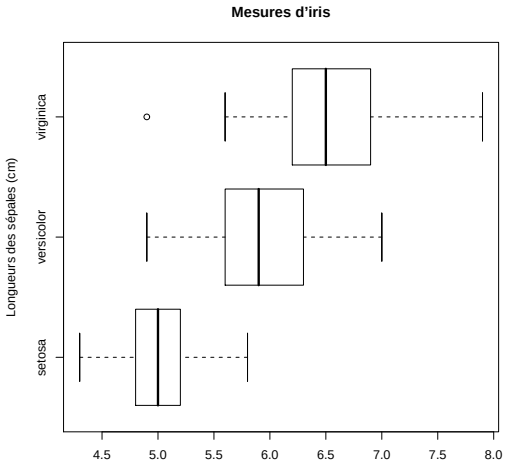

FIGURE 4 – Boîtes à moustaches

La figure FIG. 4 est le résultat des commandes suivantes :

```
> attach(iris)
> boxplot(Sepal.Length ~ Species,horizontal=TRUE,main="Mesures
+ d'iris",ylab="Longueur des sépales (cm)")
> detach(iris)
```

2.4 Histogramme

Pour une série statistique d'observations $(x_j)_{j=1\ldots n}$ (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,

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

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(x)` permet de tracer l'histogramme des données `x` 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.

Traçons, à titre d'exemple, l'histogramme d'un échantillon de taille 100 d'une loi $\mathcal{N}(2,1)$. Afin d'évaluer l'estimation de la densité fournie par l'histogramme, traçons aussi la densité d'une loi $\mathcal{N}(2,1)$ en chacun des centres de classes. Le graphique obtenu à la figure Fig. 5 a été produit par les commandes suivantes :

```
> x=rnorm(100,2,1)
> h=hist(x,freq=FALSE,col="lightgrey",main="Histogramme d'un
+ échantillon gaussien N(2,1)"))
> lines(h$mids,dnorm(h$mids,2,1),lwd=2)
```

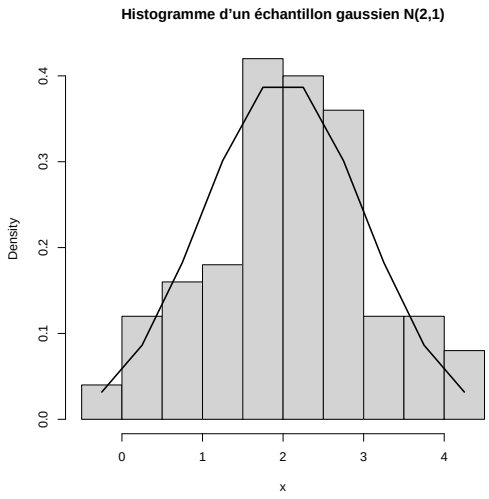

Figure 5 – Histogramme et densité d'un échantillon gaussien

3 Exercices d'application

3.1 Exercice 1

Les données `islands` de type `named vector` de R sont constituées des surfaces (en milliers de miles carrés) de 48 continents ou îles de plus de 10 000 miles carrés.

1. Tracez un histogramme des données.
2. Prenez le logarithme des données `islands` pour former un nouveau vecteur appelé `log-islands`. Tracez l'histogramme des données `log-islands`. Commentez.

3. Ouvrez un dispositif graphique à deux fenêtres. Puis, pour les données `log-islands`, tracez dans la première fenêtre un histogramme avec l'option `breaks=''Sturges''` et dans l'autre fenêtre un histogramme avec l'option `breaks=''Scott''`.
4. Construisez la boîte à moustaches des données `log-islands`.
5. Utilisez la commande graphique `dotchart` pour représenter les données `islands`. Faites la même chose pour les données par ordre croissant (à l'aide de la commande `sort`).
6. Quel(s) graphique(s) vous semble(nt) le(s) plus approprié(s) pour étudier la répartition des données ?

3.2 Exercice 2

Le `data.frame stackloss` de R rassemble 21 observations de 4 variables mesurées dans une entreprise transformant de l'ammoniac en acide nitrique.

1. Affichez les noms des 4 variables de `stackloss`.
2. A l'aide de la fonction `plot`, tracez tous les couples possibles de 2 variables.
3. Utilisez la commande `matplot` pour tracer sur un même graphique les 4 courbes des données `stackloss`. Ajoutez la légende à l'aide de la commande

   ```
   > legend("topleft",legend=names(stackloss),lty=1:4,col=1:4)
   ```

4. Utilisez maintenant la commande `pairs` sur les données `stackloss`. Identifiez le(s) couple(s) de variables qui semblent reliées linéairement ou non linéairement.
5. Effectuez la régression linéaire entre les variables `Water.Temp` et `stack.loss` et stockez les informations relatives à la régression dans une liste nommée `info`. En R, la régression linéaire entre deux variables `var1` et `var2` s'effectue au moyen de la commande `lm` comme suit :

   ```
   > # Calculs des éléments de la régression
   > info=lm(var2~var1)
   > # Affichage des résultats
   > summary(info)
   > # Représentation graphique
   > plot(var1,var2)
   > # Droite de régression
   > abline(info$coefficients)
   ```

6. Testez la normalité des résidus `info$residuals` de la régression en utilisant les commandes `qqnorm`, `qqline`, `ks.test` et `shapiro.test`.