TP d'introduction à R : Réaliser des graphiques

Attention : Ceci est un "TP-cours" où vous contruisez votre propre savoir (quelle chance !). Prenez des notes ! Posez des questions ! On ne devient pas R-guru en 3h sans mobilisation du cerveau...

1 Les différents types de fonctions graphiques

1.1 Les fonctions d'interaction avec les sorties (devices)

Il existe de nombreuses fonctions pour ouvrir un nouveau périphérique graphique (e.g. pdf(), jpeg(), postscript(), x11(), png(), ...). Elles ne sont pas toutes disponibles pour tous les systèmes d'exploitation. Pour en savoir plus, consultez l'aide de Devices :

```
?Devices
```

Le dispositif utilisé par défaut est donné par `getOption("device")`.

Lorsqu'une fonction graphique est utilisée en R, si aucune fenêtre graphique n'a déjà été créée, R en ouvre une automatiquement. Le dernier périphérique ouvert devient le périphérique actif dans lequel s'affichent les graphes suivants. La fonction `dev.list()` affiche la liste des périphériques ouverts.

Exercice 1 *Que font les commandes suivantes :*

```
x11(); x11(); pdf();
dev.list()
dev.cur()
dev.set(3)
dev.cur()
dev.off(2)
dev.list()
dev.off()
dev.off()
```

Remarque : On peut avoir des explications sur les paramètres de type graphiques en invoquant l'aide `?par`.

Exercice 2 *Créez un vecteur de 1000 points contenant des données numériques issues d'une loi normale centrée réduite (*`rnorm(1000)`*. Exportez ce graphique dans un fichier pdf.*

En utilisant une librairie spécifique, on peut également exporter ces graphiques dans un format vectoriel que l'on peut intégrer dans les rapports de stage ou autres documents. Ainsi, on peut exporter en `tikz`, le format vectoriel pour dessiner avec LaTeX. Pour charger une librairie qui n'est pas chargée de base dans l'interpréteur, et éventuellement l'installer au préalable, on procède comme suit :

```
# Si le package n'est pas installé, à ne faire qu'une seule fois
install.packages("tikzDevice", repos="http://R-Forge.R-project.org")
require(tikzDevice)
```

On peut maintenant ouvrir un device qui va correspondre à un fichier .tex :

```
tikz('normal.tex', standAlone = TRUE, width=5, height=5)
```

Tout ce qui sera dessiné entre cette commande et la commande `dev.off()` sera intégré au graphique représenté dans le fichier "normal.tex", qui sera compilable indépendamment de tout autre fichier grâce au paramètre `standAlone`.

Exercice 3 *Exportez votre histogramme de l'exercice précédent dans un fichier .tex. Compilez celui-ci (on peut le faire directement depuis la console R avec la commande :* `tools::texi2dvi('normal.tex',pdf=T)`*, et admirez. Comparez le rendu avec celui de l'affichage direct via* `hist()`*.*

1.2 Les fonctions interactives

Elles permettent d'interagir directement avec un graphique en cours d'affichage. Les principales sont :

— `locator()` qui permet de récupérer les coordonnées des points lorsque l'on clique dessus.
— `identify()` qui permet d'identifier des points. Elle donne le rang des points dans le jeu de données.

Par exemple, pour récupérer les coordonnées de 5 points, on utilisera :

```
coordonnees <- locator(5)
```

Remarque utile : pour stopper la saisie (si on a oublié le paramètre indiquant le nombre de saisies, par exemple), il suffit de cliquer droit.

Exercice 4 *Faire un affichage simple du vecteur* `mtcars$mpg` *avec la fonction* `plot`*. Expérimentez les fonctions* `locator()` *et* `identify()`*.*

1.3 Les fonctions bas niveau

Ces fonctions permettent de retoucher un graphique déjà existant. Les plus courantes sont :

— `points()`,
— `abline()`,
— `arrows()`,
— `lines()`,
— `seqments()`,
— `polygon()`,
— `rect()`,
— `box()`,
— `axis()`,
— `title()`,
— `rug()`,
— `grid()`,
— `legend()`,
— `text()`,
— `mtext()`

Reprenons notre jeu de données de maïs (voir TP précédent). Si tout va bien, il est sauvegardé dans votre espace de travail. Vérifiez en tapant `summary(mais)`.

On commence avec ce graphique de base, représentant la hauteur des pieds de mais en fonction de la masse de ces pieds.

```
plot(mais$Hauteur~mais$Masse)
```

Tel quel, il n'est pas très joli. Rajoutons-lui des couleurs, qui vont en plus être informatives de l'état du pied (attaqué ou pas attaqué).

```
plot(mais$Hauteur~mais$Masse,
     col=ifelse(mais$Couleur=="Jaune",
                "yellow",
                ifelse(mais$Couleur=="Rouge",
                       "red",
                       "orange")),
     pch=19)
```

Exercice 5 *En vous aidant de l'aide (et oui...), rajoutez à ce graphique :*

1. *Un titre "Hauteur des pieds de maïs en fonction de la masse", en gras, et des labels d'axes respectivement "Hauteur" en ordonnées et "Masse" en abscisses.*
2. *Une légende indiquant les couleurs des pieds, en bas à droite, dans un rectangle.*
3. *Une droite de régression (calculée avec* `reg = lm(mais$Hauteur ~ mais$Masse`*), en bleu. Remarquez que ~ permet de confronter un vecteur à un autre, c'est une façon de dire "en fonction de".*
4. *Une grille de dix cases de côté, en gris clair (*`"lightgray"`*).*

1.4 Les fonctions de haut niveau

Ce sont celles que l'on utilise le plus souvent parce qu'elles donnent un graphique complet. Elles sont très nombreuses, ce sont par exemple :

— **plot()**,
— **hist()**,
— **pie()**,
— **barplot()**,
— **boxplot()**,
— **pairs()**,...

Nous allons en explorer quelques unes.

2 Les histogrammes

La fonction `hist()` permet d'afficher un vecteur sous forme d'histogramme, c'est-à-dire que l'on représente par des rectangles verticaux le nombre d'éléments du vecteur qui appartiennent aux classes définies par les extrémités inférieures du rectangle.

Exercice 6 *Tapez dans le terminal la commande* `hist(mais$N.grains)`. *Vous remarquez que par défaut, il y a 50 valeurs possibles pour le nombre de grain dans chaque classe. On voudrait affiner un peu cet histogramme prenant des tailles de classe différentes. Le paramètre* `breaks` *peut prendre une valeur numérique définissant le nombre de classes voulues. Essayez pour les valeurs 5,20,50. Que remarquez-vous sur les valeurs prises en ordonnées ?*

Par défaut, l'axe des ordonnées représente le nombre d'éléments dans la classe. Mais pour pouvoir comparer à des distributions statistiques, il est souvent plus intéressant de représenter la proportion d'élément dans la classe, plutôt que le nombre (absolu) des élements. C'est le paramètre `probability` qui joue sur cet aspect de la représentation.

Exercice 7 *Représentez l'histogramme précedent avec le paramètre* `probability = T`. *Remarquez les changements de valeur.*

Exercice 8 *En observant le résultat de la commande suivante, vous examinerez le rôle de chacun des paramètres (vous pouvez tester en les enlevant pour voir la différence) :*

```
hist( mais$Hauteur, col = grey(0.9), border = grey(0.2),
main = paste("Taille de", nrow(mais), " pieds de maïs"),
xlab = "Taille [cm]",
ylab = "Effectifs relatifs",
labels = TRUE, las = 1, ylim = c(0, 0.01), probability = T)
```

Exercice 9 *Pour voir si la distribution s'ajuste bien à une loi normale, on va représenter sur le même diagramme la densité de la loi normale de même moyenne et même écart-type que l'échantillon.*

```
x <- seq(from = min(mais$Hauteur, na.rm=T), to = max(mais$Hauteur, na.rm=T), length = 100)
lines(x, dnorm(x, mean(mais$Hauteur, na.rm = TRUE), sd(mais$Hauteur,
na.rm = TRUE)))
```

Vous devriez obtenir le diagramme de la figure 1. Pensez-vous que l'adéquation à la loi normale soit correcte ?

Vers l'analyse... un test statistique existe pour savoir si des données suivent une loi normale. Il s'agit du test de Shapiro, que l'on appelle avec la commande `shapiro.test`. Si la p-valeur retournée par le test est forte (par exemple, supérieure à 0.05), alors le test est positif.

Exercice 10 *Lancez la commande* `shapiro.test(mais$Hauteur)`. *Concluez...*

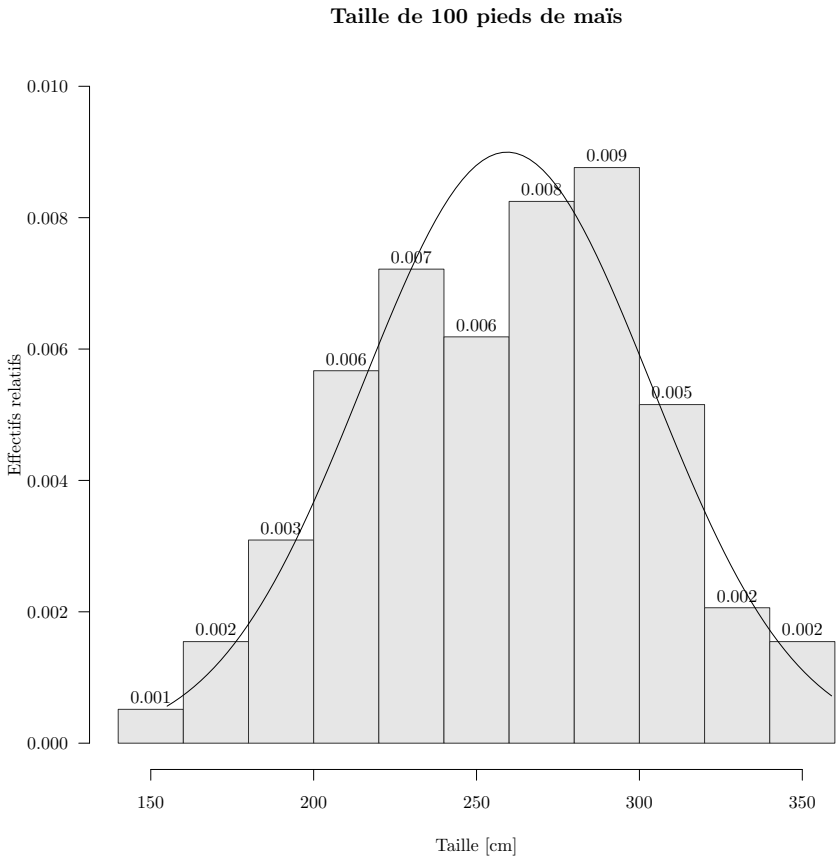

FIGURE 1 – Un bel histogramme

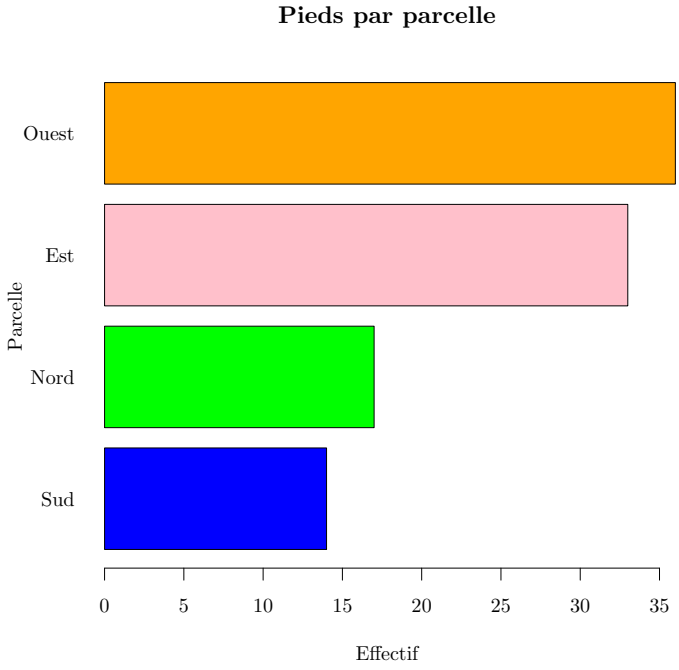

FIGURE 2 – Un beau diagramme en bâtons.

3 Représenter des variables discrêtes

Quand la nature de la variable est discrête, on peut la représenter sous forme de diagramme en bâton. C'est le cas par exemple du nombre de jours d'attaque dans le dataframe "mais".

```
barplot(table(mais$Parcelle))
```

Exercice 11 *En jouant sur les paramètres d'orientation (*`horiz`*), et de couleur des colonnes, essayez d'obtenir la figure 2 (Ouest est en orange, Est en rose, Nord en vert, et Sud en bleu).*

Exercice 12 *Une autre possibilité pour représenter ces données est le camembert ("piechart" en anglais). Explorez l'aide de la fonction* `pie` *pour obtenir la figure 3.*

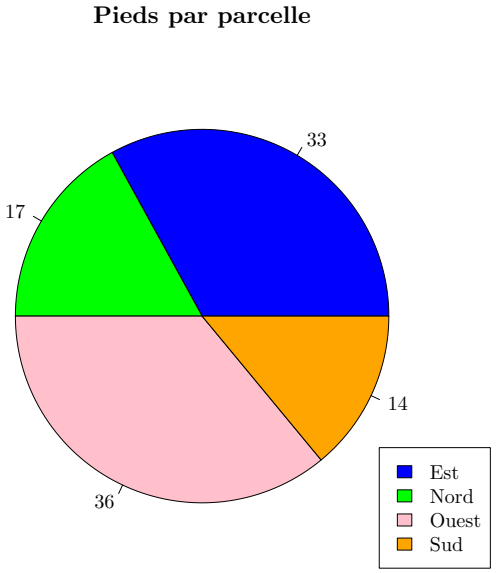

FIGURE 3 – Un beau camembert.

4 Représenter des variables continues avec des boîtes à moustaches

Nous avons déjà vu l'utilisation pour les variables continues des histogrammes. On peut également utiliser une représentation en boîte à moustaches :

```
boxplot( mais$Masse.grains, col = grey(0.8),
main = paste("Masse des grains pour ", nrow(mais), " pieds de maïs"),
ylab = "g", las = 1)
rug(mais$Masse.grains, side = 2)
```

Exercice 13 *Recopiez la commande ci-dessus, et observez. Que fait la commande* `rug` *? En observant le résultat de* `summary(mais$Masse.grains)`*, dites à quoi correspondent les limites de la boîte. Et le trait du milieu.*

Les boîtes à moustaches permettent de comparer facilement les groupes d'individus, par exemple ici la hauteur des pieds attaqués par rapport à la hauteur des pieds non attaqués.

```
boxplot(mais$Hauteur.J7 ~ mais$Attaque)
```

Exercice 14 *Étudiez les distributions de nombres de jours d'attaque en fonction de si un traitement a été versé ou non.*

5 Représentation des contingences

Pour analyser un jeu de données, il peut être utile de se donner une vue d'ensemble. On va s'intéresser maintenant au jeu de données stocké dans le dataframe `iris` (chargé de base dans R).

Exercice 15 *Regardez le résumé de ce jeu de données. Combien d'espèces sont représentées ? Lancez la commande* `pairs(iris)`*. On obtient des nuages de points qui ne sont pas forcément très interprétables...*

Pour améliorer la lisibilité, on va représenter les données un peu différemment. Tout d'abord, on va s'intéresser au croisement des données quantitatives (toutes les longueurs, largeurs, etc.) avec la donnée qualitative.

Exercice 16 *Pour chaque variable quantitative, dessinez des boîtes à moustaches, une par valeur de la variable qualitative (espèce). Indiquez lesquelles vous semblent différentes selon les espèces, et si c'est le cas, classez les espèces selon leurs caractéristiques.*

Pour les variables quantitatives qu'il faut croiser, le nuage de point peut donner une bonne indication sur la corrélation linéaire entre deux données. On s'intéresse plus particulièrement à la représentation de la longueur des pétales en fonction de leur largeur :

```
plot(iris$Petal.Length ~ iris$Petal.Width)
```

On remarque que parmi les points représentés, certains se superposent, ce qui ne rend pas très lisible les zones où ils apparaissent. On peut alors bruiter les données pour rendre ces zones un peu plus "lisibles" :

```
plot(jitter(iris$Petal.Length) ~ jitter(iris$Petal.Width))
```

Il reste à regarder ce que l'on peut faire pour observer le croisement de deux données qualitatives, comme par exemple, sur le jeu de données de maïs, la parcelle et l'enracinement.

Exercice 17 *À l'aide de la fonction* `table`*, créez un tableau de contingence* `tc` *entre ces deux données. Puis représentez-le à l'aide de plot.*

On aimerait avoir une représentation un peu plus explicite de ce tableau de contingence.

Exercice 18 *Pour créer des graphiques en ballon, on a recours aux librairies et à la commande suivantes :*

```
library(gdata)
library(gtools)
library(gplots)
balloonplot(tc, dotsize=10)
```

Il est possible que les libraries ci-dessus ne soient pas installées de base dans R. Si elles ne le sont pas, vous pouvez les installer en utilisant la commande `install.packages("nom_de_la_librairie")`*. Observez le résultat...*

6 Les courbes...

La fonction `plot` permet également de dessiner des graphiques tout à fait classiques, par exemple une représentation de fonction.

Exercice 19 *Définir une fonction* `f` *représentant la fonction mathématique suivante :*

$$x \mapsto \frac{\sqrt{3x^4+4}}{x^2 + x\log(x^2+1)+1}$$

et tracez sa courbe pour x entre -10 et 10.