EILCO **2023-2024**
Cycle Ingénieur 1ère année

TP2 : Statistiques descriptives, graphiques

1 Analyse univariée

Divers packages permettent d'obtenir des statistiques sur une base de données. Par exemple, la fonction `describe()` du package *Hmisc* permet d'obtenir pour chaque variable : le nombre d'observations, le nombre de valeurs manquantes, la moyenne, les quantiles.

Pour les variables qualitatives, `describe()` donne le nombre d'observations, le nombre de valeurs manquantes, et le nombre de niveaux pour la variable ; ainsi que le nombre d'observations pour chaque niveau de la variable (et la proportion).

Pour visualiser les 6 premières lignes, on peut taper la fonction `head()`.

1.1 Variables quantitatives

On illustrera les fonctions de base en utlisant l'ensemble de données `iris`, disponible dans `R`. Taper les codes suivants :

```
 iris; describe(iris); iris$Sepal.Length; iris[1]; iris[[1]]; iris[1,];
iris[,1]; Slength<-iris$Sepal.Length
```

Indicateurs de tendance centrale et de dispersion Pour une variable quantitative, les statistiques de base qu'on peut calculer sont le minimum, le maximum, la moyenne, la variance et l'écart type, la médiane et les autres quantiles :

```
min(Slength)
max(Slength)
range(Slength)
mean(Slength)
var(Slength)
sd(Slength)
median(Slength)
quantile(Slength)
quantile(Slength, probs=c(0.01, 0.1, 0.53, 0.99))
summary(Slength)
```

Représentations graphiques R possède de nombreux outils pour faire des graphiques, notamment les diagrammes à moustache et les histogrammes :

```
boxplot(Slength, col = c("yellow"), main = "Boxplot", ylab = "Quantiles")
hist(Slength)
hist(Slength, breaks = c(0, 5.4, 6.5, 7.9))
```

On peut également faire apparaître une famille de diagrammes à moustache, en tapant par exemple :

```
boxplot(iris[,c("Sepal.Length","Petal.Width")],
col = c("yellow", "blue"),
main = "Boxplot",
ylab = "Quantiles")
```

1.2 Variables qualitatives

On illustrera les fonctions de base pour une variable qualitative, à nouveau à partir de l'ensemble de données `iris`. La variable qualitative qui nous intéresse est `Species<-iris$Species`. Taper les lignes suivantes :

```
summary(Species)
prop.table(table(Species))
barplot(table(Species))
pie(summary(Species))
```

2 Analyse bivariée

On continue de travailler avec le jeu de données `iris`.

2.1 Variable quantitative vs variable quantitative

Les deux variables quantitatives qui nous intéressent sont `Slength<-iris$Sepal.Length` et `Plength<-iris$Petal.Length`. Taper les lignes suivantes :

```
cov(Slength,Plength)
cor(Slength,Plength)
plot(x=Slength, y=Plength)
```

Pour trouver la droite des moindres carrés (modèle linéaire simple) et l'ajouter aux données, taper :

```
m <- lm(Slength ~ Plength)
summary(m)
plot(Slength ~ Plength)
abline(m, col = "red")
```

2.2 Variable quantitative vs variable qualitative

Pour calculer une fonction f sur les valeurs d'une variable y pour chaque niveau d'un facteur x, on utilise `tapply(X=y, INDEX=x, FUN=f)`. Cela est très utile quand on veut calculer les statistiques d'une variable quantitative y regroupé selon les niveaux d'une variable qualitative x :

```
tapply(Slength, iris$Species, mean)
tapply(Slength, iris$Species, summary)
```

Graphiquement, on peut faire des boxplot de la variable quantitative en fonction de la variable qualitative :

```
boxplot(Slength ~ iris$Species, data = iris, col=c(2,3,4))
```

2.3 Variable qualitative vs variable qualitative

Il est également possible de travailler sur des couples de variables qualitatives, mesurées sur les mêmes individus. On peut faire des tables de contingences et obtenir les fréquences totales et marginales (en lignes et en colonnes), en utilisant les fonctions suivantes `table(,), prop.table()`. Pour obtenir des graphiques, utiliser la fonction `mosaic()`.

3 Quelques précisions sur les graphiques

3.1 Constructions de base et nuages de points

En R les graphiques sont créés par étapes : on crée un plot et ensuite on ajoute des lignes, des points, une légende, ... Nous avons déjà vu comment ajouter une droite à un nuage de points. Pour compléter le graphique, nous pouvons spécifier un titre, changer les étiquettes aux axes, ajouter une légende. Dans cette section, on s'intéressera aux données de `mtcars`, et notamment à `wt<-mtcars$wt` et `mpg<-mtcars$mpg`. Taper :

```
plot(wt, mpg, main="Poids et consommation des voitures",
xlab="poids", ylab="consommation")
abline(h=mean(mpg), col=2, lty=2)
legend(x="topright", lty=2, col=2, legend="consommation moyenne")
```

La fonction `abline(b,a)` permet d'ajouter à un plot existant une droite d'ordonnée à l'origine b et de coefficient directeur a. Par exemple, une façon alternative d'ajouter la droite de régression est la suivante :

```
m <- lm(mpg ~ wt)
intercept<-coef(m)[1]
pente<-coef(m)[2]
```

```
plot(wt, mpg, main="Droite des moindres carrés")
abline(a=intercept, b=pente, col=2)
```

Pour ajouter une ligne quelconque (pas nécessairement une droite) on utilise la fonction `lines()`. Par exemple, pour ajouter la courbe de lissage lowess (locally-weighted polynomial regression) :

```
plot(wt, mpg, main="Courbe de lissage")
lines(lowess(mpg ~ wt), col="blue", lwd=2)
```

On peut ajouter de nouveaux points à l'aide de `points()`. Taper, par exemple :

```
P<- matrix(c(4.5,4.6,5,26,26,30),3,2)
points(P)
```

On peut contrôler le rang des axes à l'aide de `xlim()` et `ylim()` :

```
plot(wt,mpg, main="Mêmes données avec échelle différente",
xlim=c(0,25),
ylim=c(0,100))
```

3.2 Line graphs

La fonction `lines()` ajoute une ligne à un graphique existant et ne peut pas être utilisée pour créer un nouveau graphique. Pour faire le graphe d'une fonction, il faut utiliser la fonction `plot(x=vecteur,type=l)` :

```
mydata<-cos(2*pi/12*(1:36)) + rnorm(36, 0, 0.2)
plot(mydata,type="l",main="type=l")
```

Le paramètre type permet de relier (ou non) les points de façons différentes (voir l'aide). Taper, par exemple :

```
par(mfrow=c(2,2))
plot(mydata, type="p", main="type=p")
plot(1:10, type="S", main="type=S")
plot(1:10, type="s", main="type=s")
```

Avec plot="h" ,on obtient un graphiques à barres :

```
plot(1:10,type="h",main="type=h")
```

3.3 Paramètres graphiques

Pour imposer des paramètres à la totalité des graphiques produits au cours d'une session, on utilisera la fonction `par()`. Cette dernière est souvent utilisée pour visualiser deux ou plusieurs plots dans la même fenêtre avec le paramètre `mfrow=c(l,c)`. Dans ce cas, les graphiques sont visualisés dans une grille avec l lignes et c colonnes (ci-dessous, on prend `hp<-mtcars$hp`) :

```
par(mfrow=c(1,2))
plot(wt, mpg, main="Poids, consommation")
boxplot(hp, main="Puissance")
```

La fonction `par()` permet également de changer les marges des graphiques, tester par exemple `par(mar=c(5, 6, 4, 2) + 0.1)`. Les changements sont implémentés jusqu'à la fermeture de la session. Pour revenir à une seule fenêtre, on peut taper la fonction `layout(1)`.

3.4 Histogrammes

Par défaut, `hist()` affiche l'histogramme avec les effectifs, ce qui correspond au paramètre `freq=TRUE` (en anglais frequency indique les occurrences). Pour afficher l'histogramme avec la densité on utilisera `freq=FALSE`. Dans ce cas, l'aire de chaque rectangle sera égale à la proportion d'observations dans la classe correspondante (de façon à que l'aire totale de tous les rectangles soit bien égale à 1).

```
par(mfrow=c(1,2))
hist(mpg)
hist(mpg,freq=FALSE)
```

Pour superposer la courbe d'une densité donnée :

```
mydata<-rnorm(1000,0,1)
hist(mydata, freq=F, main="Histogramme et densité théorique")
curve(dnorm, from=-5, to=5, add=TRUE, col=2)
```

Une autre façon de superposer la densité de la loi normale centrée réduite et l'histogramme est d'utiliser la fonction `lines()` :

```
mydata<-rnorm(1000,0,1)
hist(mydata, freq=F, main="Histogramme et densité théorique")
x<-seq(-3, 3, by=0.1)
lines(x, dnorm(x), col=2)
```

4 Exercice

Exercice Tracer, sur le même graphique, les graphes des fonctions sinus et cosinus sur l'intervalle $[-\pi,\pi]$. Mettre une légende et un titre.