TP R 2: Statistiques descriptives et graphiques de base

Cours de Programmation

Vitorio Perduca, Master 1 Mathématiques et Applications

UFR Math-Info, Université Paris Descartes, septembre 2020

Table des matières

1. Données utilisées

On illustrera les fonctions de base en utilisant le jeu de données `mtcars` disponible dans `R`. Pour une description de ces données, consulter l'aide `?mtcars`. On peut avoir une idée de la classe et du contenu de `mtcars` à l'aide de `str()` :

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

La variable `cyl` représente le nombre de cylindres avec trois modalités possibles (4,6 ou 8), il est donc naturel de la convertir en facteur :

```
mtcars$cyl.factor <- as.factor(mtcars$cyl)
attach(mtcars)
```

2. Analyse univariée

2.1 Variables quantitatives

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 (on rappelle que le quantile d'ordre p est la valeur q t.q. p est la fraction des valeurs observées inférieures à q).

```
min(mpg)
```

```
## [1] 10.4
```

```
max(mpg)
```

```
## [1] 33.9
```

```
range(mpg)
```

```
## [1] 10.4 33.9
```

```
mean(mpg)
```

```
## [1] 20.09062
```

```
var(mpg)
```

```
## [1] 36.3241
```

```r
sd(mpg)
```

```
## [1] 6.026948
```

```r
median(mpg)
```

```
## [1] 19.2
```

```r
quantile(mpg)
```

```
##     0%    25%    50%    75%   100%
## 10.400 15.425 19.200 22.800 33.900
```

```r
quantile(mpg, probs = 0.99) # pour le percentile d'ordre 99
```

```
##    99%
## 33.435
```

```r
quantile(mpg, probs = c(0.01, 0.1, 0.9, 0.99))
```

```
##     1%    10%    90%    99%
## 10.400 14.340 30.090 33.435
```

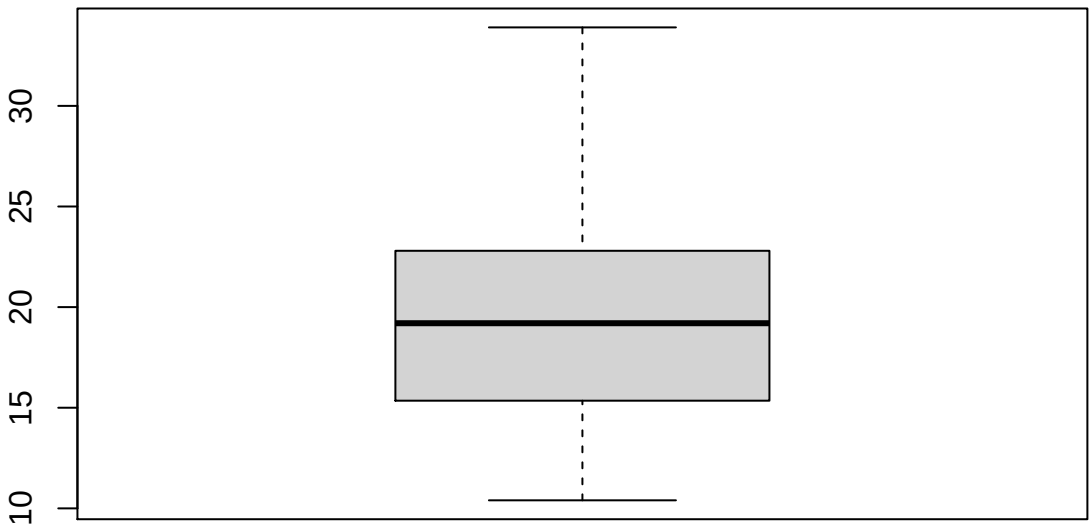
Si il y a des données manquantes, on les exclut du calcul de la statistique à l'aide de l'option `na.rm=TRUE`.

La fonction générique `summary()` permet de décrire rapidement la distribution d'un échantillon et est complémentaire à la boîte à moustache :

```r
summary(mpg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90
```

```r
boxplot(mpg)
```

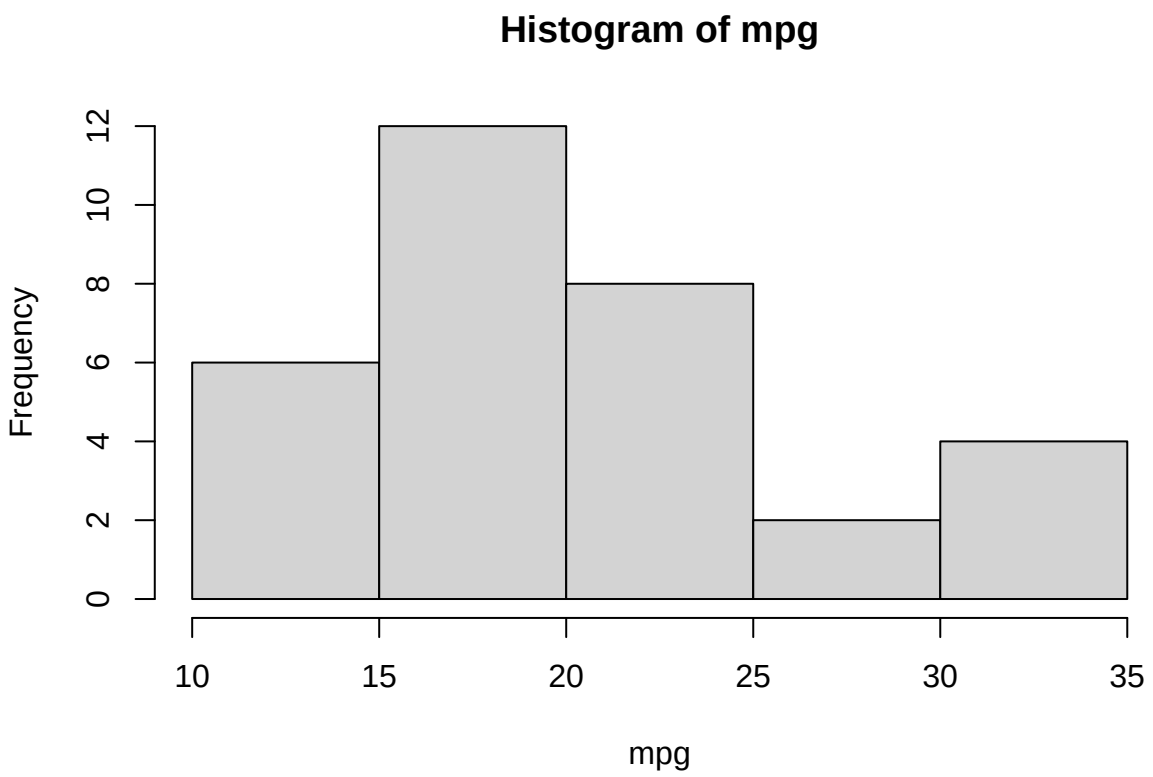


On rappelle que dans un boxplot

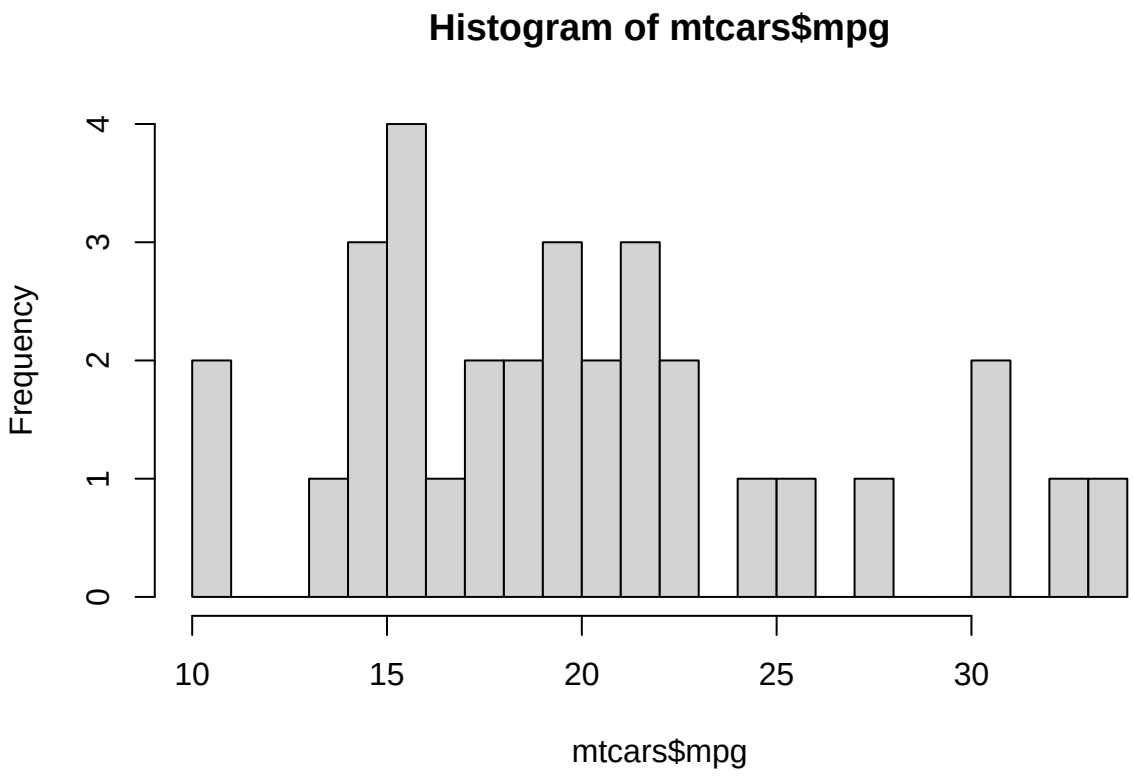
— la hauteur (ou largeur si on choisit l'option `horizontal=TRUE`) est définie par le premier et troisième quartile q_1 et q_3
— le trait horizontal (ou vertical) correspond à la médiane $m = q_2$
— la moustache en bas (ou à gauche) part de q_1 et va jusqu'au minimum de l'échantillon s'il n'y a pas de points extrêmes à gauche, c'est à dire des valeurs inférieures à $q_1 - 1.5 \times (q_3 - q_1)$. S'il y a de tels points, la moustache s'arrête en $q_1 - 1.5 \times (q_3 - q_1)$ et on les place au-dessous (à gauche) de celle-ci.
— la moustache en haut (ou à droite) part de q_3 et va jusqu'au maximum de l'échantillon s'il n'y a pas de poins extrêmes à droite, c'est à dire des valeurs supérieurs à $q_3 + 1.5 \times (q_3 - q_1)$. S'il y a de tels points, la moustache s'arrête en $q_3 + 1.5 \times (q_3 - q_1)$ et on les place au-dessus (à droite) de celle-ci.

L'autre outil graphique pour représenter la distribution d'une variable quantitative est l'histogramme :

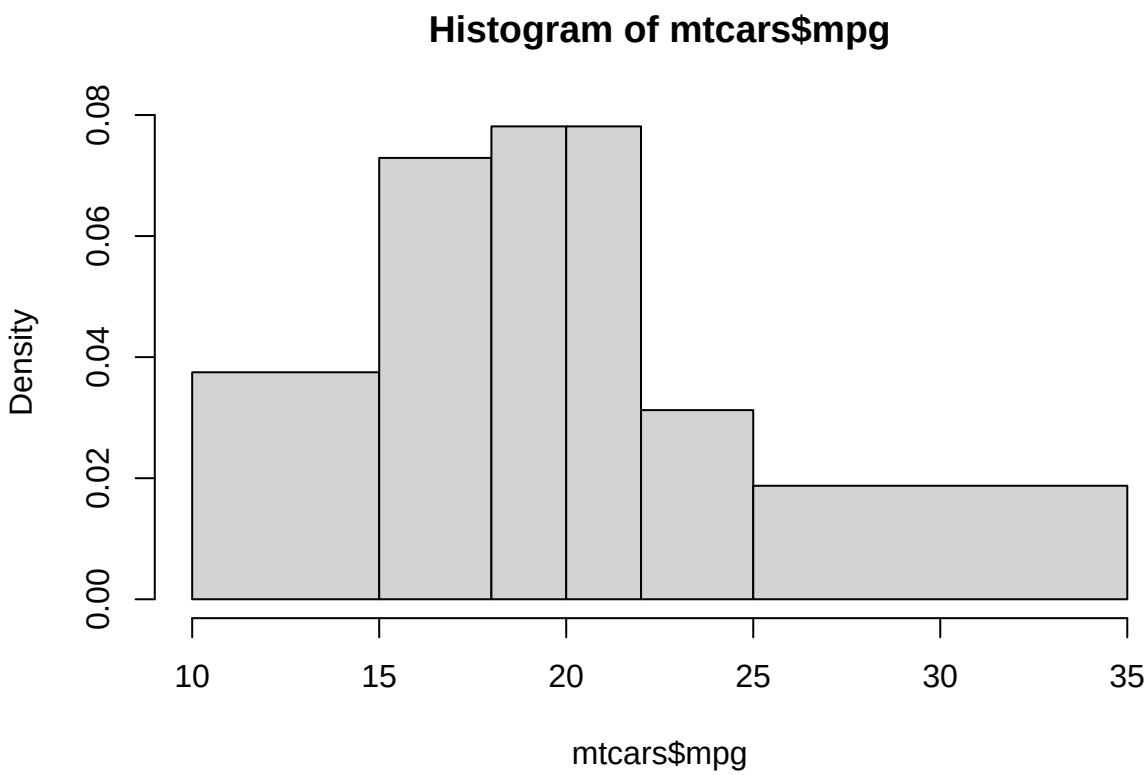
```
hist(mpg)
```




```
hist(mtcars$mpg, breaks = 20) # si on veut 20+1 intervalles
```



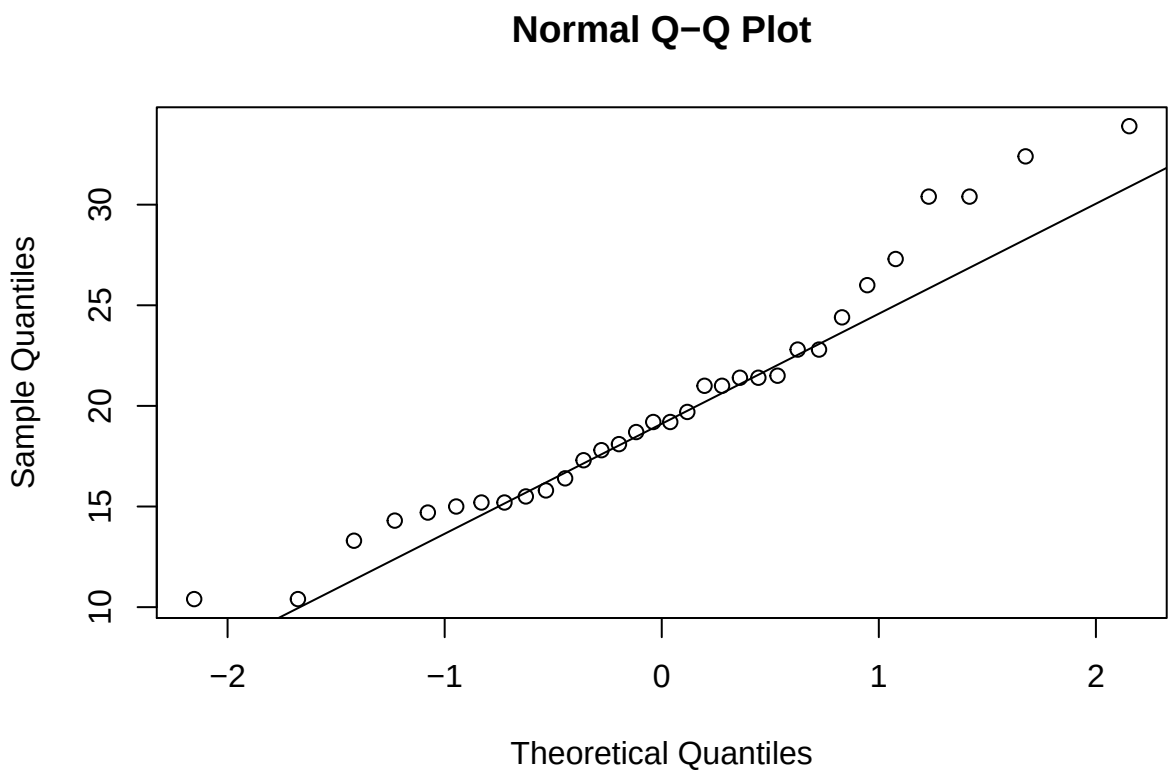

```
hist(mtcars$mpg, breaks = c(10, 15, 18, 20, 22, 25, 35))
```



```
# on spécifie à la main les points qui définissent les intervalles
```

On peut aussi comparer la distribution de l'échantillon avec la distribution théorique d'une loi normale centrée et réduite à l'aide du QQ-plot :

```
qqnorm(mpg)
qqline(mpg) # la droite sample quantiles = theoretical quantiles
```



On peut comparer le QQ-plot précedant à celui de l'échantillon *centré et réduit* :

```
mpg.stand <- (mpg-mean(mpg))/sd(mpg) # mpg centré et réduit
qqnorm(mpg.stand); qqline(mpg.stand)
```

Bien évidemment la distribution des quantiles ne changes pas :

```
hist(quantile(mpg))
hist(quantile(mpg.stand))
```

2.2 Variables qualititatives

Les fonctions `summary()` et `table()` appliquées à un facteur (ou à un vecteur de caractères) comptent les occurrences des différents niveaux.

```
summary(cyl.factor)
```

```
##  4  6  8
## 11  7 14
```

```r
table(cyl.factor)
```

```
## cyl.factor
##  4  6  8
## 11  7 14
```

Pour obtenir les proportions :

```r
prop.table(table(cyl.factor))
```

```
## cyl.factor
##       4       6       8
## 0.34375 0.21875 0.43750
```

En présence de valeurs manquantes, `summary()` compte aussi les `NA` :

```r
x <- as.factor(c('a','a','b','c',NA,'c'))
summary(x)
```

```
##    a    b    c NA's
##    2    1    2    1
```

```r
table(x)
```

```
## x
## a b c
## 2 1 2
```

```r
prop.table(table(x)) # les effectifs sont divisés par le nb total des valeurs non-manquantes
```

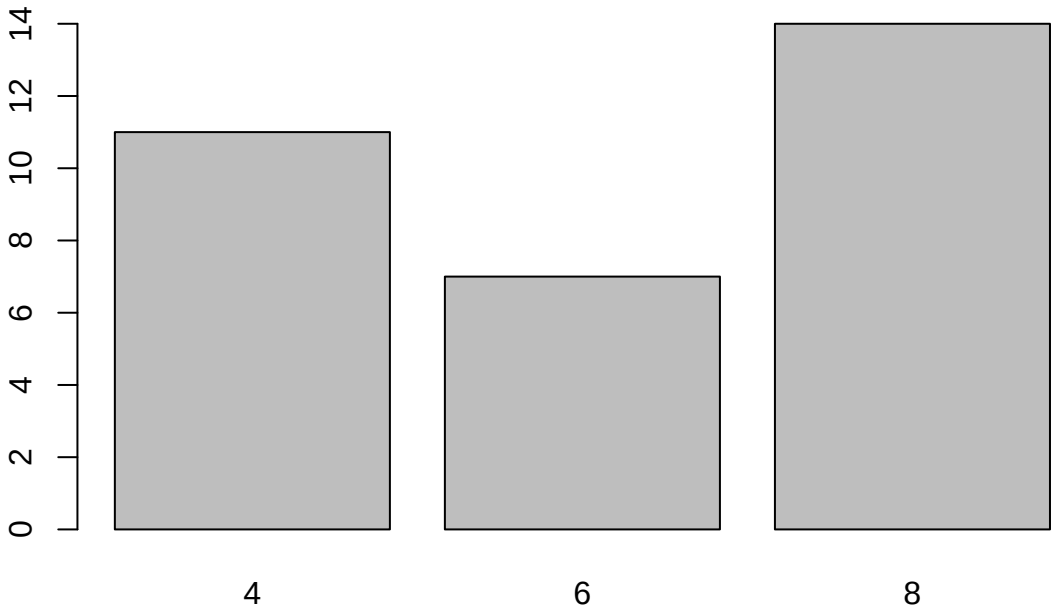
```
## x
##   a   b   c
## 0.4 0.2 0.4
```

```r
prop.table(summary(x)) # les effectifs sont divisés par le nb total des valeurs
```

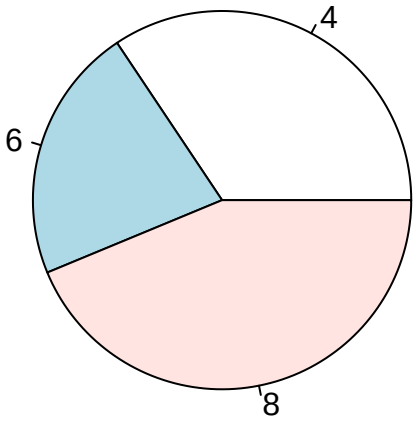
```
##         a         b         c      NA's
## 0.3333333 0.1666667 0.3333333 0.1666667
```

On utilisera un graphiques à barres ou un camembert (déconseillé) :

```r
barplot(table(cyl.factor))
```




```
pie(summary(cyl.factor))
```



3. Analyse bivariée

3.1 Variable quantitative vs variable quantitative

On considère deux variables quantitatives mesurées sur les mêmes individus.

La covariance de deux variable quantitatives mesure les écarts conjoints par rapport à leurs moyennes respectives. La coefficient de corrélation linéaire de Pearson ρ mesure l'intensité de dépendance linéaire entre deux variables quantitatives. ρ appartient à $[-1, 1]$: il vaut $+1$ (-1) si il y a une dépendance linéaire positive (négative) et 0 si il n'y a pas de dépendance linéaire.

```
cov(mpg, wt)
```

```
## [1] -5.116685
```

```
cor(mpg, wt)
```

```
## [1] -0.8676594
```

```
# cov(mpg, wt)/sd(mpg)/sd(wt) # essayer
```

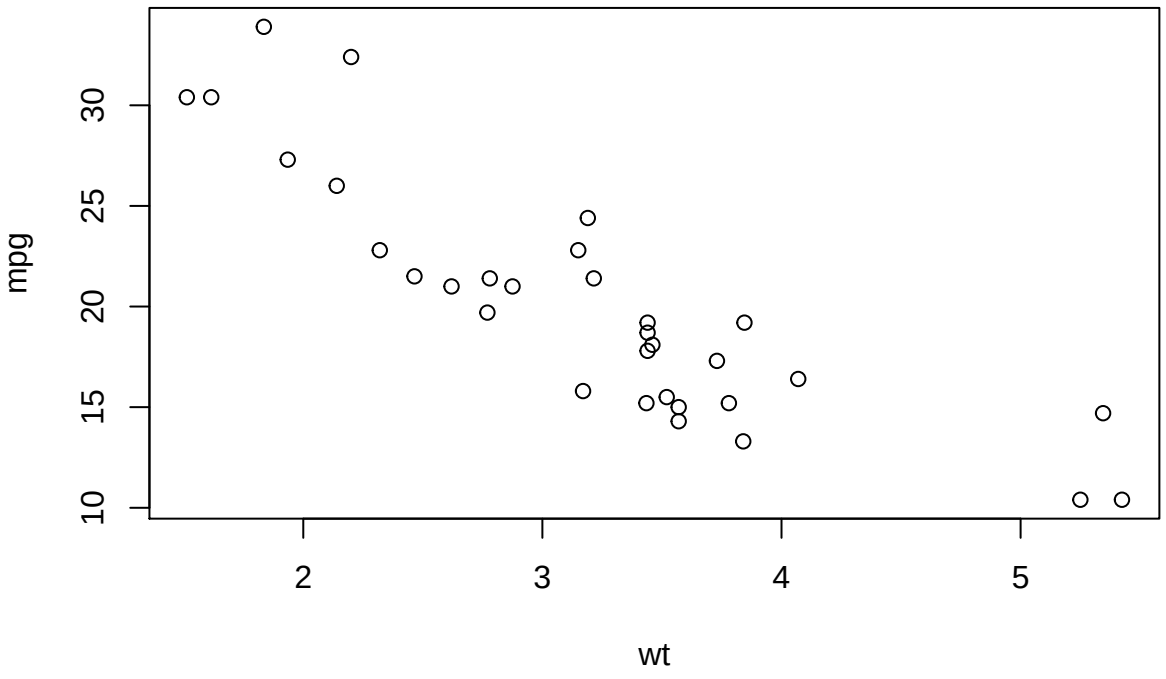
La corrélation de Spearman mesure la corrélation linéaire entre les *rangs* des valeurs de deux variables :

```
cor(mpg, wt, method='spearman')
```

```
## [1] -0.886422
```

Pour représenter graphiquement deux variables quantitatives on utilisera un nuage de point, ou *scatter plot :*

```
plot(x=wt, y=mpg) # x,y = coordonnées des points.
```



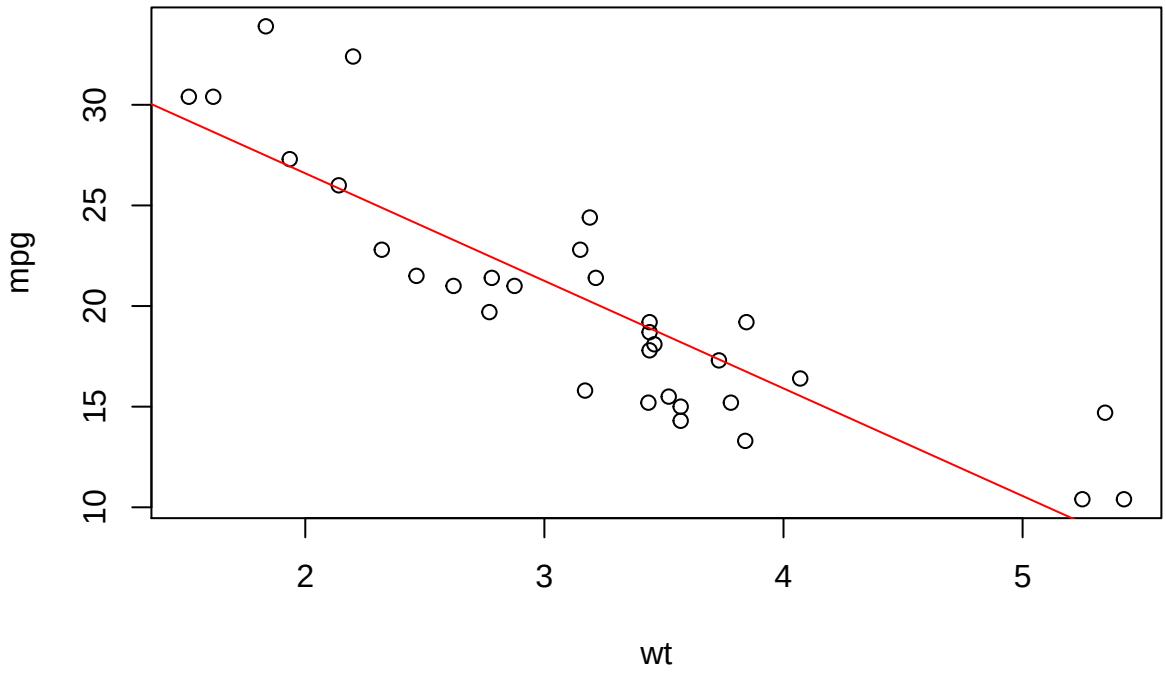
```r
# Alternative: plot(wt ~ mpg) où le ~ indique "wt en fonction de mpg"
```

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

```r
m <- lm(mpg ~ wt)
summary(m)
```

```
##
## Call:
## lm(formula = mpg ~ wt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```r
# essayer:
# pente <- sum((wt-mean(wt))*(mpg-mean(mpg)))/sum((wt-mean(wt))^2)
# intercept <- mean(mpg) - pente * mean(wt)
plot(mpg ~ wt)
abline(m, col = "red")
```



3.2 Variable qualitative vs variable qualitative

On considère deux variables qualitatives mesurées sur les mêmes individus.

On fait des tables de contingences et on calcule les fréquences totales et marginales (en lignes et en colonnes) :

```
t <- table(cyl,am)
# cyl et am sont des vécteurs numériques,
# mais pour construire la table de contingence
# table() compte leurs modalités.
print(t)
```

```
##    am
## cyl  0  1
##   4  3  8
##   6  4  3
##   8 12  2
```

```
prop.table(t) # P(cyl,am)
```

```
##    am
## cyl       0       1
##   4 0.09375 0.25000
##   6 0.12500 0.09375
##   8 0.37500 0.06250
```

```
prop.table(t, margin = 1) # P(am | cyl=4), P(am | cyl=6), P(am | cyl=8)
```

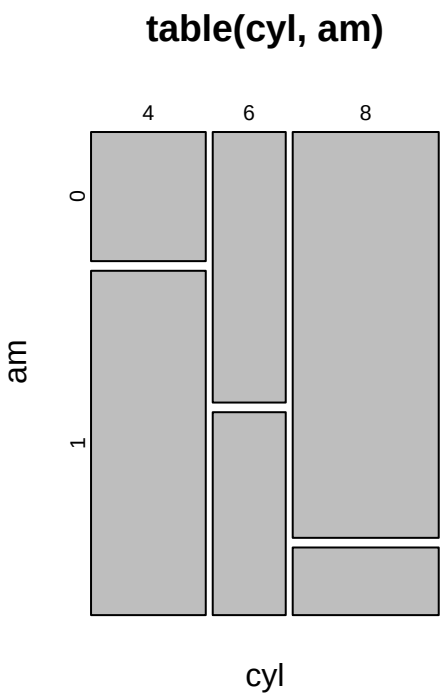
```
##    am
## cyl         0         1
##   4 0.2727273 0.7272727
##   6 0.5714286 0.4285714
##   8 0.8571429 0.1428571
```

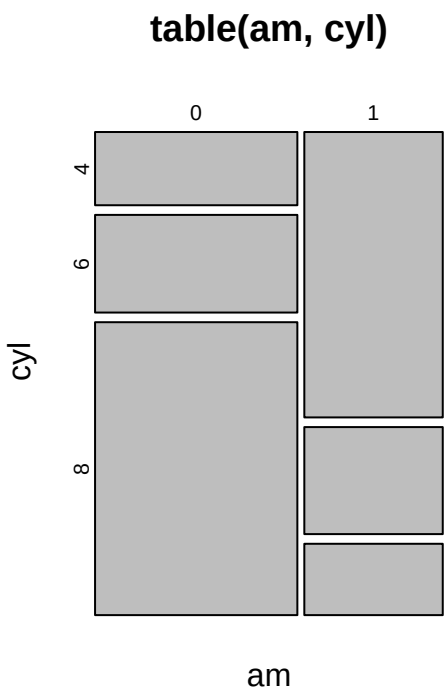
```
prop.table(t, margin = 2) # P(cyl | am=0), P(cyl | am=1)
```

```
##    am
## cyl         0         1
##   4 0.1578947 0.6153846
##   6 0.2105263 0.2307692
##   8 0.6315789 0.1538462
```

Le graphique correspondant est le *mosaic plot* :

```
par(mfrow=c(1,2)) # pour mettre deux graphiques côte à côte
mosaicplot(table(cyl,am)) # P(am | cyl=4), P(am | cyl=6), P(am | cyl=8)
mosaicplot(table(am,cyl)) # P(cyl | am=0), P(cyl | am=1)
```





3.3 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` regroupées selon les niveaux d'une variable qualitative `x` :

```
tapply(mtcars$mpg, mtcars$cyl, mean)
```

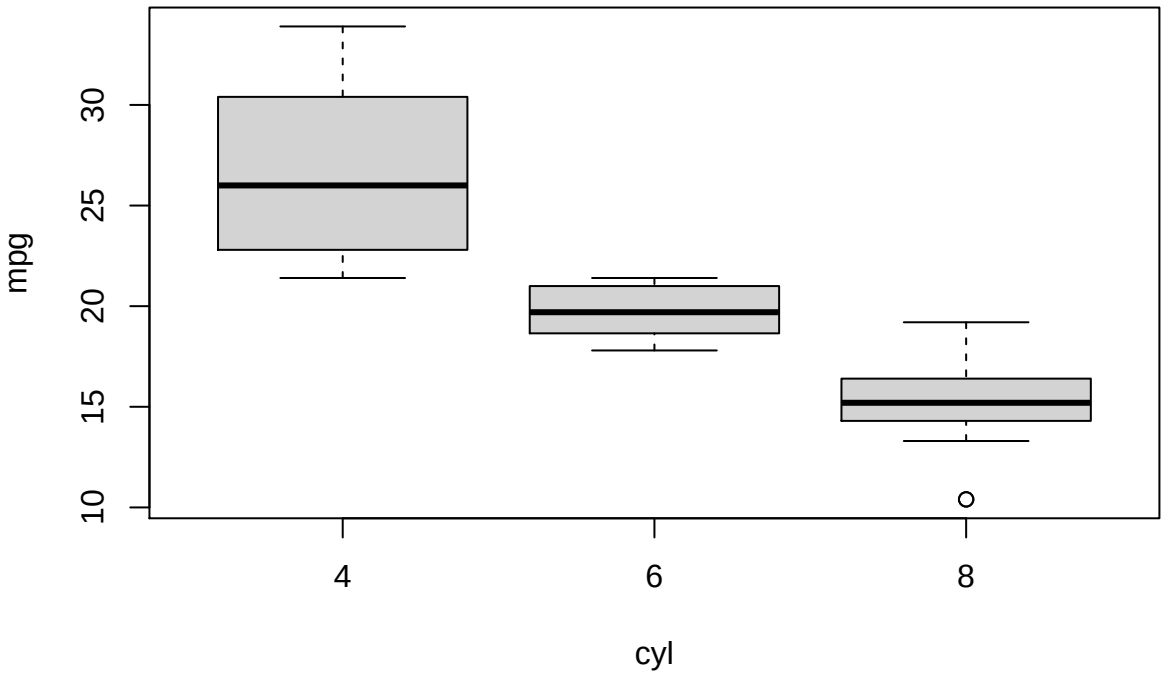
```
##        4        6        8
## 26.66364 19.74286 15.10000
```

```
tapply(mtcars$mpg, mtcars$cyl, summary)
```

```
## $`4`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   21.40   22.80   26.00   26.66   30.40   33.90
##
## $`6`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17.80   18.65   19.70   19.74   21.00   21.40
##
## $`8`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   14.40   15.20   15.10   16.25   19.20
```

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

```r
boxplot(mpg ~ cyl, data = mtcars)
```

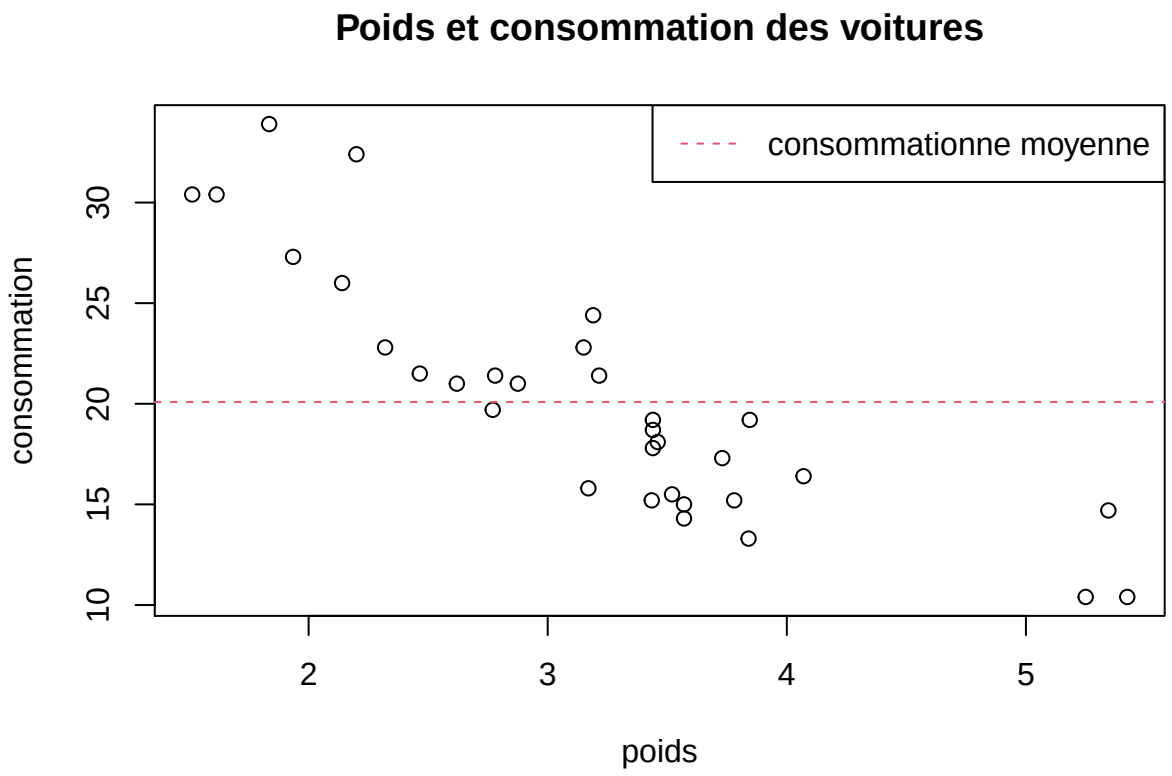


4. Détails sue les graphiques de base

4.1 Constructions de base et scatterplots

En R les graphiques sont crées 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, et ajouter une légende.

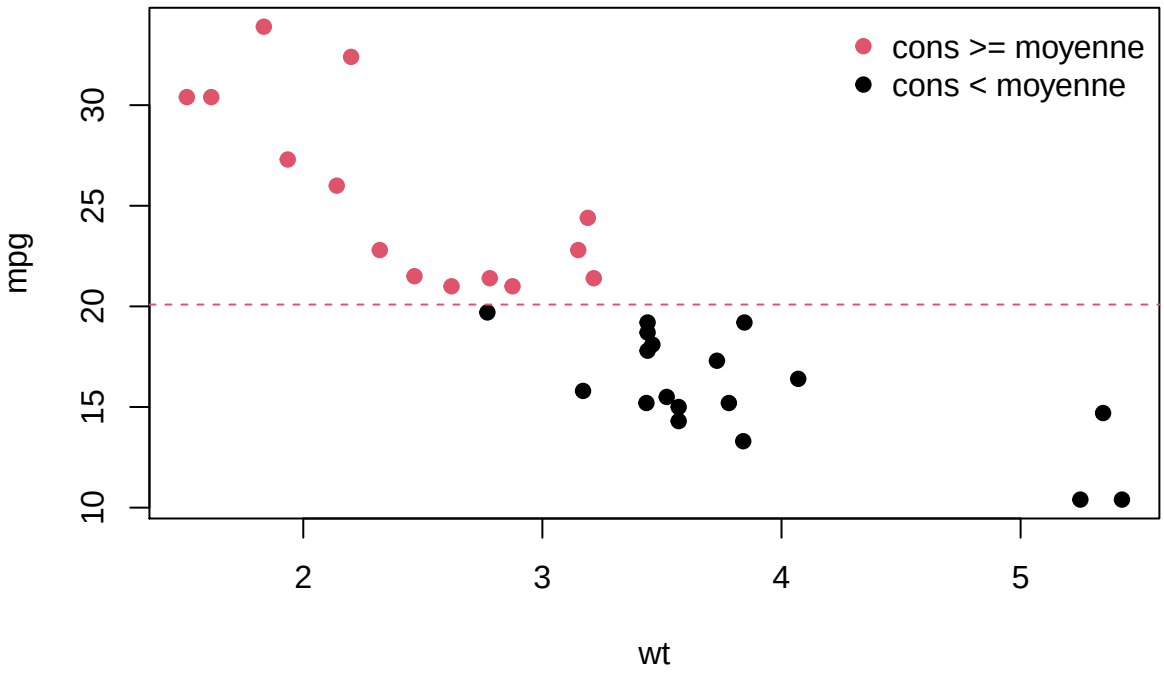
```
plot(wt, mpg,
     main='Poids et consommation des voitures',
     xlab='poids',ylab='consommation') # xlab=label axe x
abline(h=mean(mpg),col=2,lty=2) # h=droite horizontale, col=couleur, lty=type de ligne
legend(x='topright',lty=2,col=2,legend='consommationne moyenne')
```




```
# x=emplacement légende
```

Nous pouvons aussi ajouter une troisième dimension en coloriant les points selon une condition :

```
plot(wt , mpg,
     pch=19, # pch=type de point
     col=(mpg>=mean(mpg))+1 # 1=noir, 2=rouge
     )
abline(h=mean(mpg),col=2,lty=2) # h=droite horizontale, col=couleur, lty=type de ligne
legend(x='topright',
       pch=19,
       col=2:1,
       legend=c('cons >= moyenne','cons < moyenne'), # legend=vecteur avec les labels
       bty='n' #pas de cadre autour de la légende
       )
```

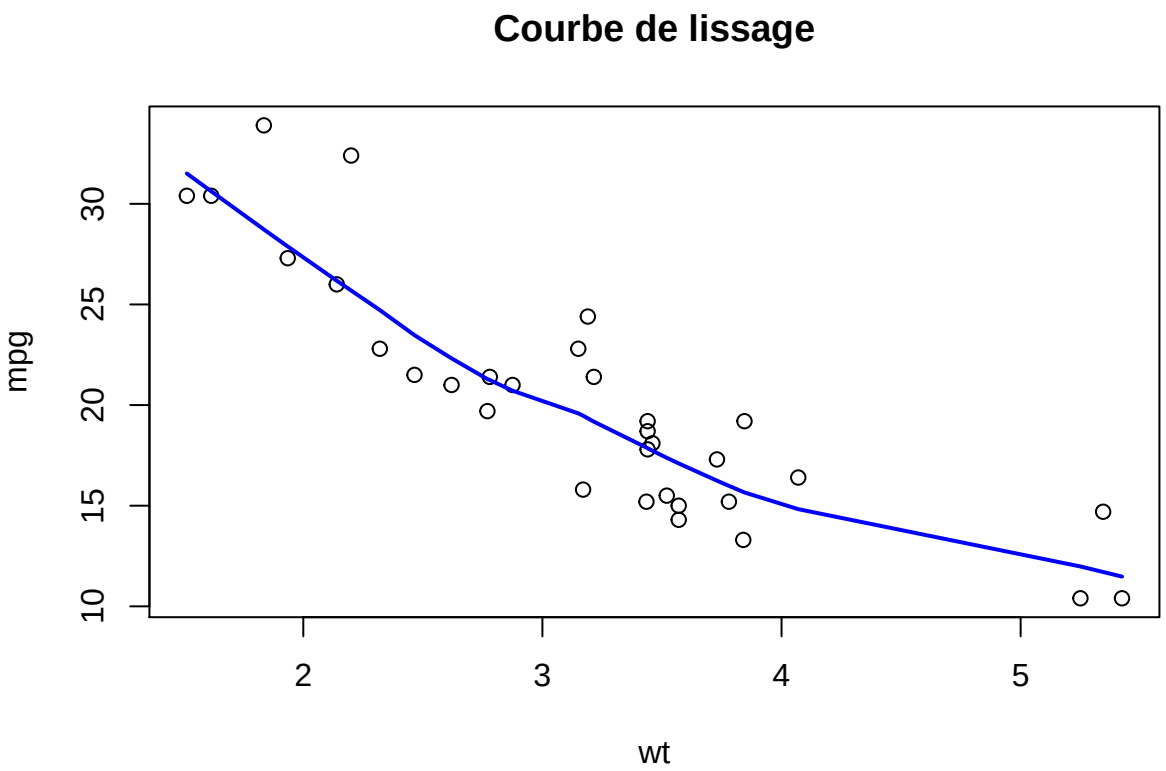


En général, `abline(a,b)` permet d'ajouter à un plot existant une droite d'intercept `a` et pente `b`. Par exemple, une façon alternative d'ajouter la droite de régression est la suivante :

```
intercept=coef(m)[1]
pente=coef(m)[2]
plot(wt , mpg, main='Droite des moindres carrés')
abline(a=intercept, b=pente) # abline() pour ajouter une droite
abline(lowess(wt,mpg))
```

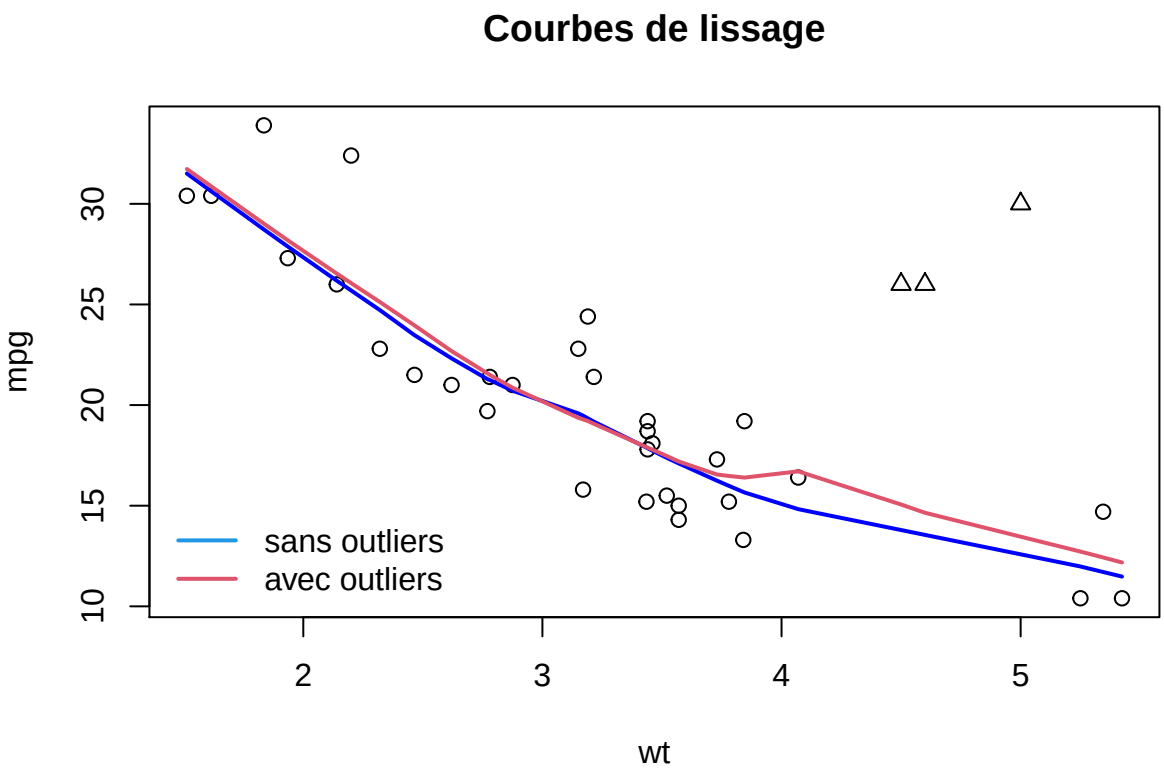
Pour ajouter une ligne quelconque (pas nécessairement une droite) on utilise la fonction `lines()`. Par exemple, pour ajouter la courbe de lissage *lowess* (locally-weighted polynomial regression) :

```
plot(wt, mpg, main='Courbe de lissage')
lines(lowess(mpg~wt), # lines(): pour ajouter une courbe
      col='blue', # même chose que col=4
      lwd=2) # lwd=epaisseur ligne
```



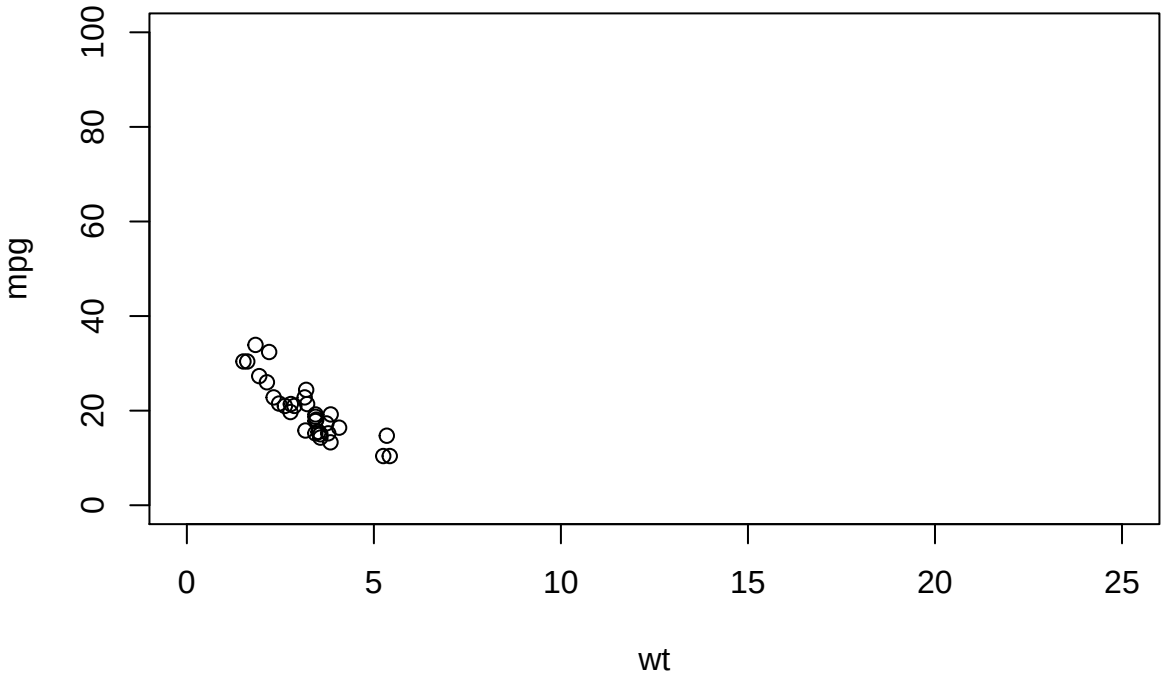
On peut ajouter de nouveaux points à l'aide de `points()` :

```
plot(wt , mpg, main='Courbes de lissage')
lines(lowess(mpg~wt), col='blue', lwd=2)
outliers=cbind(c(4.5,4.6,5),c(26,26,30))
points(outliers,pch=2) # pch=2 pour des triangles
lines(lowess(c(wt,outliers[,1]),
             c(mpg,outliers[,2])
             ),
      col=2,lwd=2)
legend(x='bottomleft',col=c(4,2),lwd=2,legend=c('sans outliers','avec outliers'),bty='n')
```



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

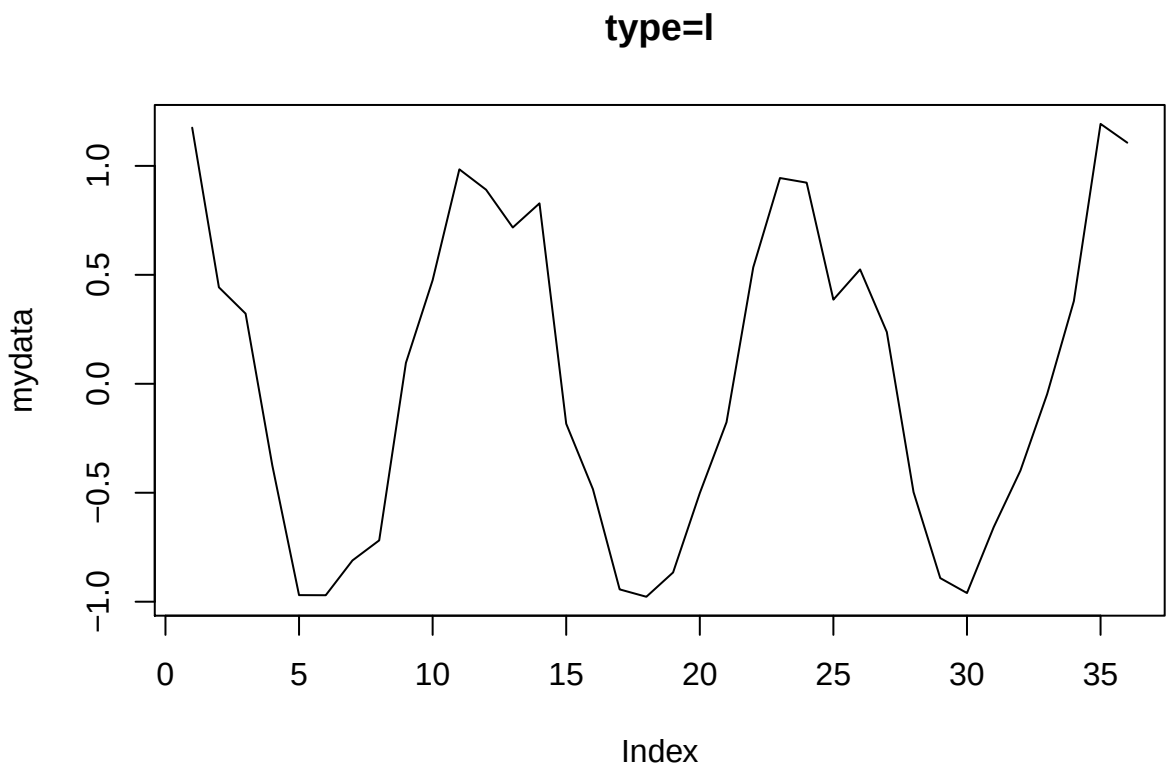
```
plot(wt,mpg, main='Mêmes données avec differente échelle',
     xlim=c(0,25),
     ylim=c(0,100))
```



4.2 Line graphs

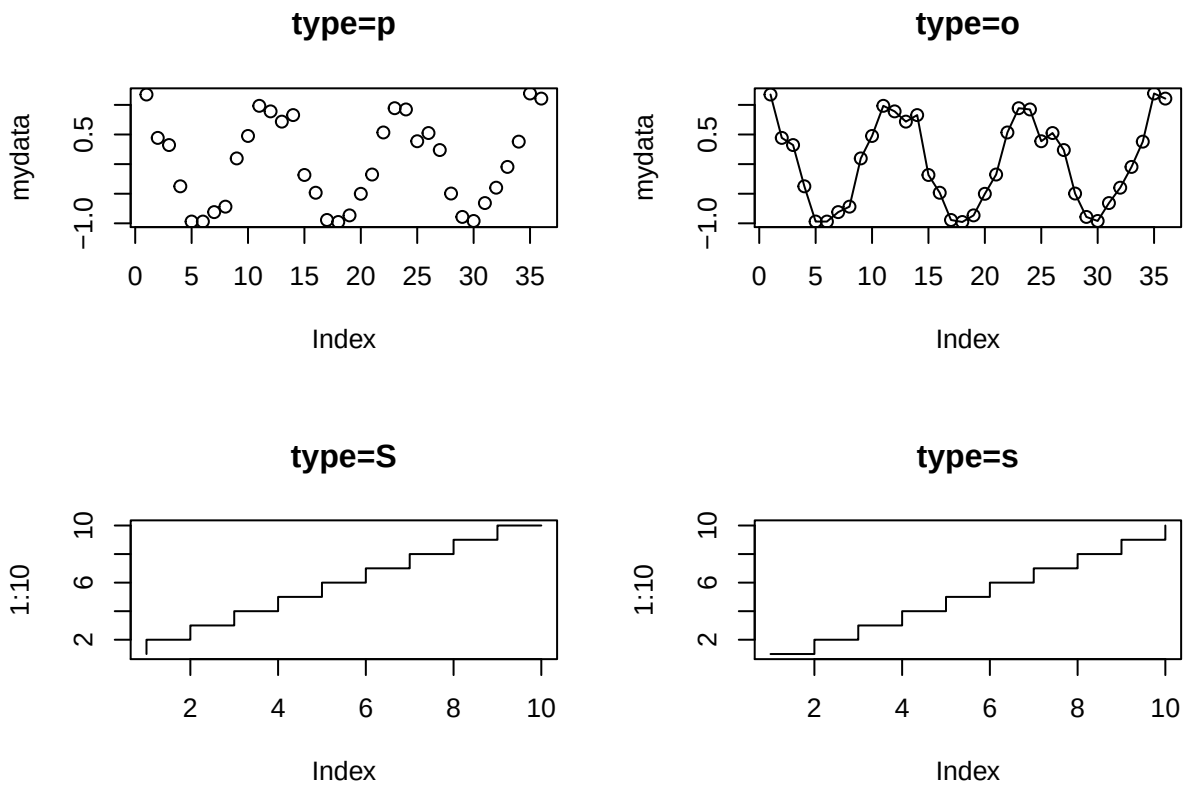
La fonction `lines()` ajoute une ligne à un graphique existant et ne peut pas être utilisée pour créer un nouveau graphique. Pour connecter une ligne reliant les éléments successifs d'un vecteur on utilisera plutôt `plot(x=vecteur,type=l)` :

```
mydata=cos(2*pi/12*(1:36)) + rnorm(36,0,.2) # simulation des données
plot(mydata,type='l',main='type=l')
```



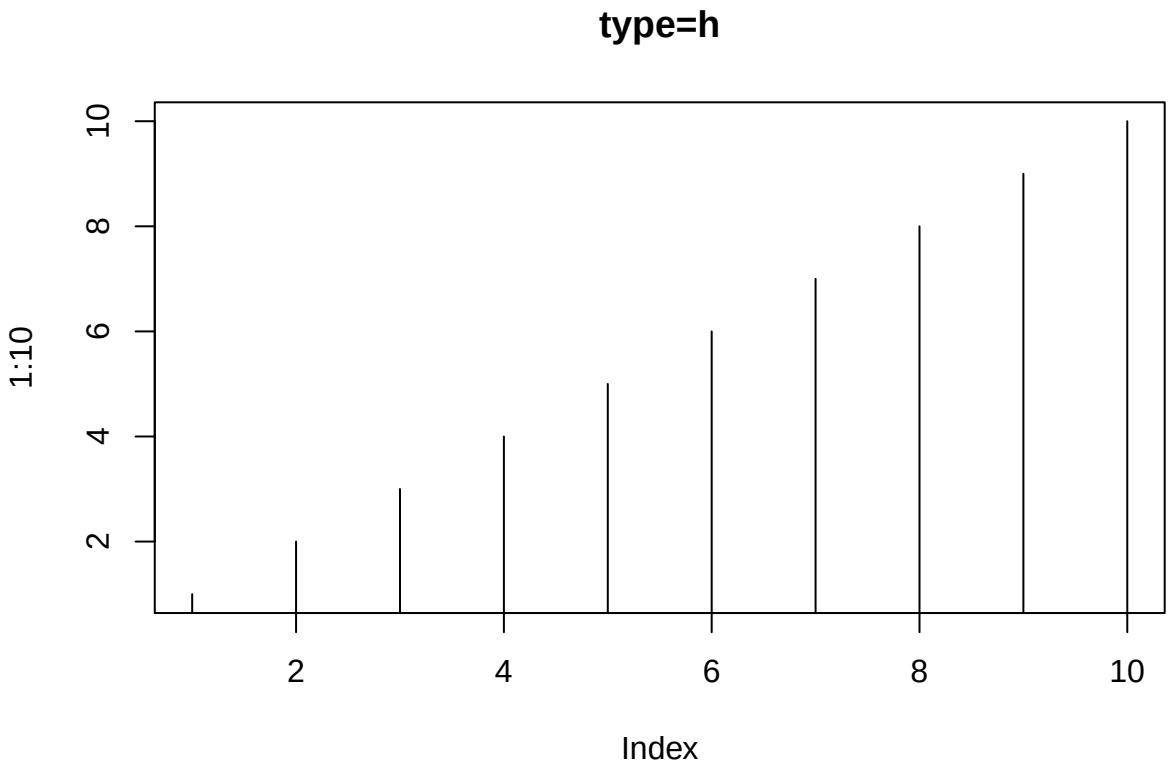
Le paramètre `type` permet de relier (ou pas) les points de façons différentes :

```
par(mfrow=c(2,2)) # expliqué dans la section suivante
plot(mydata,type='p',main='type=p') # défault
plot(mydata,type='o',main='type=o')
plot(1:10,type='S',main='type=S') # escalier 1
plot(1:10,type='s',main='type=s') # escalier 2
```



Avec `plot='h'` on obtient un graphiques à barres :

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



4.3 Paramètres graphiques

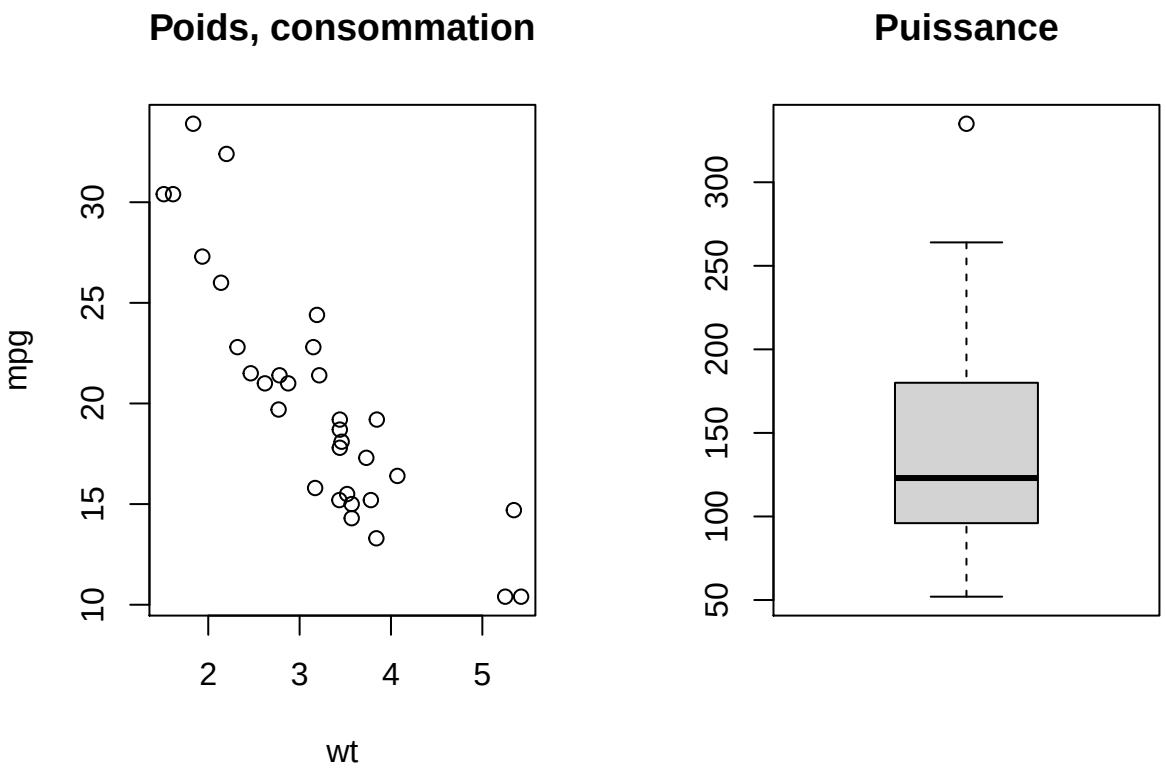
Nous avons déjà vu qu'il est possible de contrôler certains éléments graphiques à l'aide de paramètres :

Élément	Paramètre
point	`pch`
Graphique de type ligne	`type`
couleur	`col`
ligne	`lty`, `lwd`
étiquette axes	`xlab`,`ylab`
dimensions axes	`xlim`,`ylim`
dimension labels	`cex`
orientation labels axes	`las`

Consulter `?par` pour une description des valeurs que ces paramètres peuvent prendre.

Pour imposer des paramètres à la totalité des graphiques produits au cours d'une session on utilisera la fonction `par()`. `par()` est souvent utilisée pour visualiser deux ou plusieurs plot 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 :

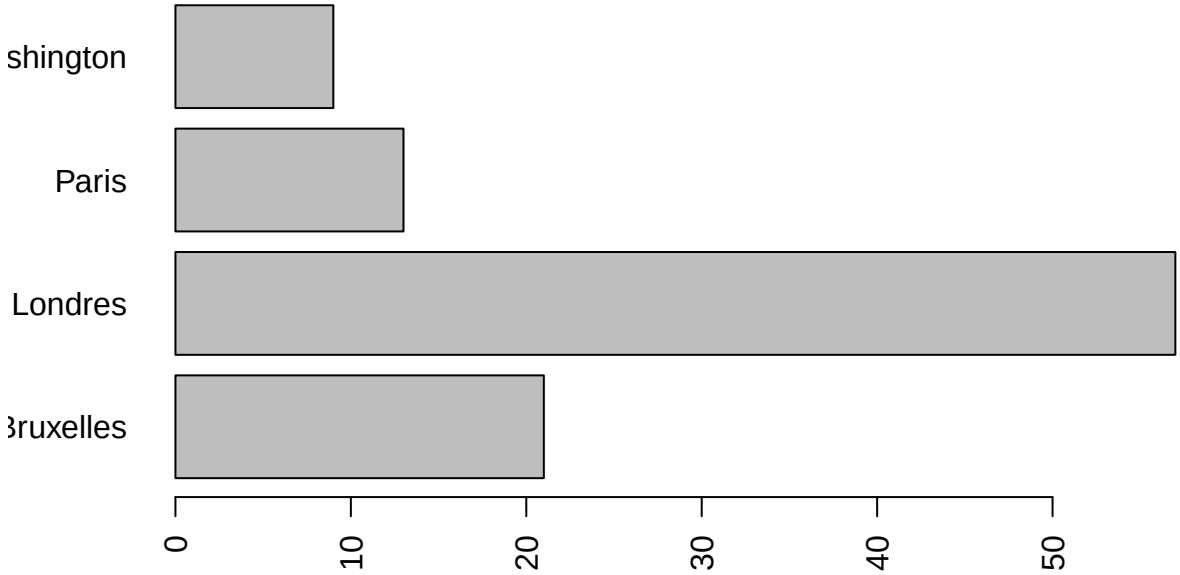
```
# opar = par() # pour pouvoir reinitialiser avec les paramètres initiaux
par(mfrow=c(1,2))
plot(wt,mpg,main='Poids, consommation')
boxplot(hp,main='Puissance')
```



Les changements sont implémentés jusqu'à quand la session est fermée, ou le moteur graphique est réinitialisé par `dev.off()` ou en cliquant sur le balai `Clear all plots` dans `RStudio`.

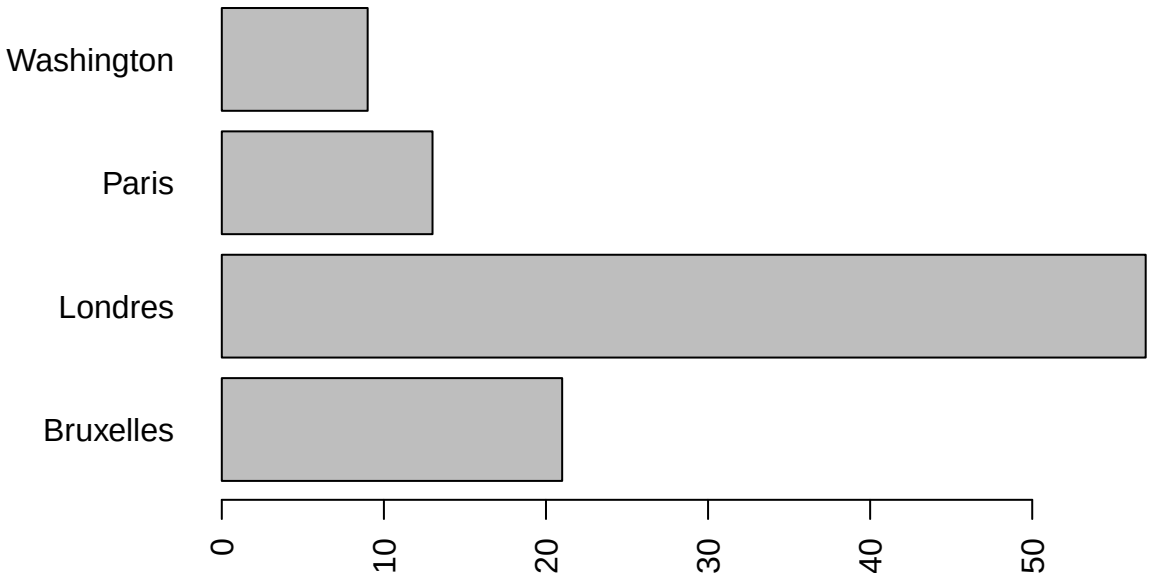
`par()` est aussi utilisée pour changer la dimension des marges. Cela est parfois utile quand les labels sur les axes ne rentrent pas dans la fenêtre, comme dans cet exemple :

```
mydata=as.factor(sample(x=c('Paris','Londres','Bruxelles','Washington'),
                        replace = T,
                        prob=c(.1,.5,.2,.1),
                        size=100)) # simulation d'une variable qualitative
barplot(summary(mydata),
        width=.1,
        horiz=T,
        las=2)
```



On modifie le paramètre qui règle la marge à gauche :

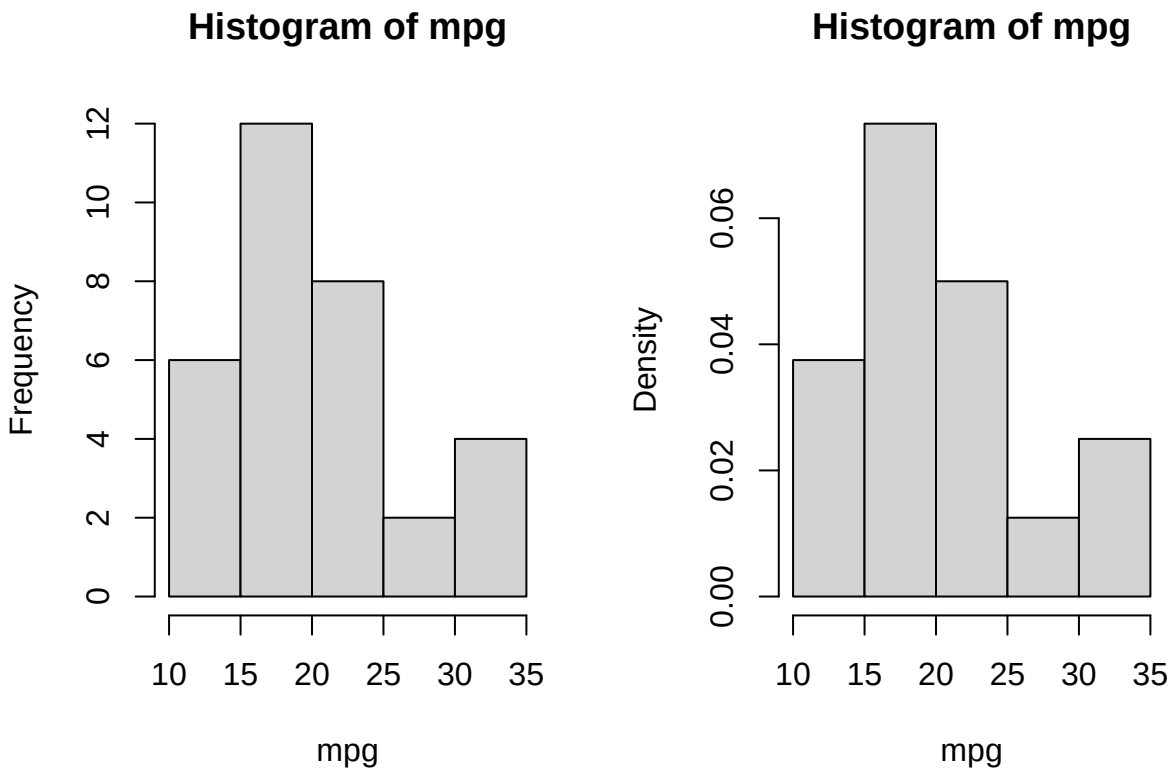
```
par(mar=c(5, 6, 4, 2) + 0.1) # la deuxième composante de mar donne la marge à gauche
# Default est c(5, 4, 4, 2) + 0.1
barplot(summary(mydata),
        width=.1,
        horiz=T,
        las=2)
```


Washington
Paris
Londres
Bruxelles
0
10
20
30
40
50

4.4 Histogrammes

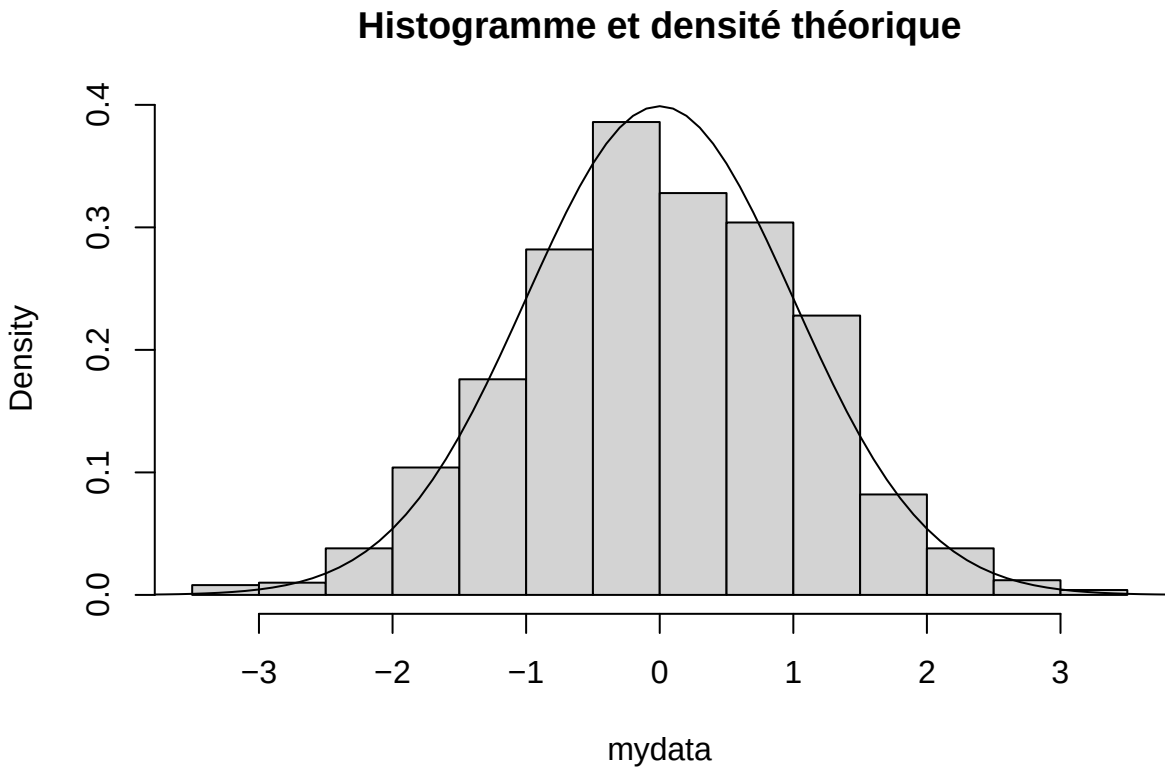
Par défaut, `hist()` affiche l'histogramme avec les comptages, ce qui correspond au paramètre `freq=TRUE` (en anglais *frequency* indique les occurrences). Pour afficher l'histogramme avec la *densité* on utilisera `freq=FALSE`. Dans ce cas, l'aire de chaque rectangle sera égale à la proportion d'observations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

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



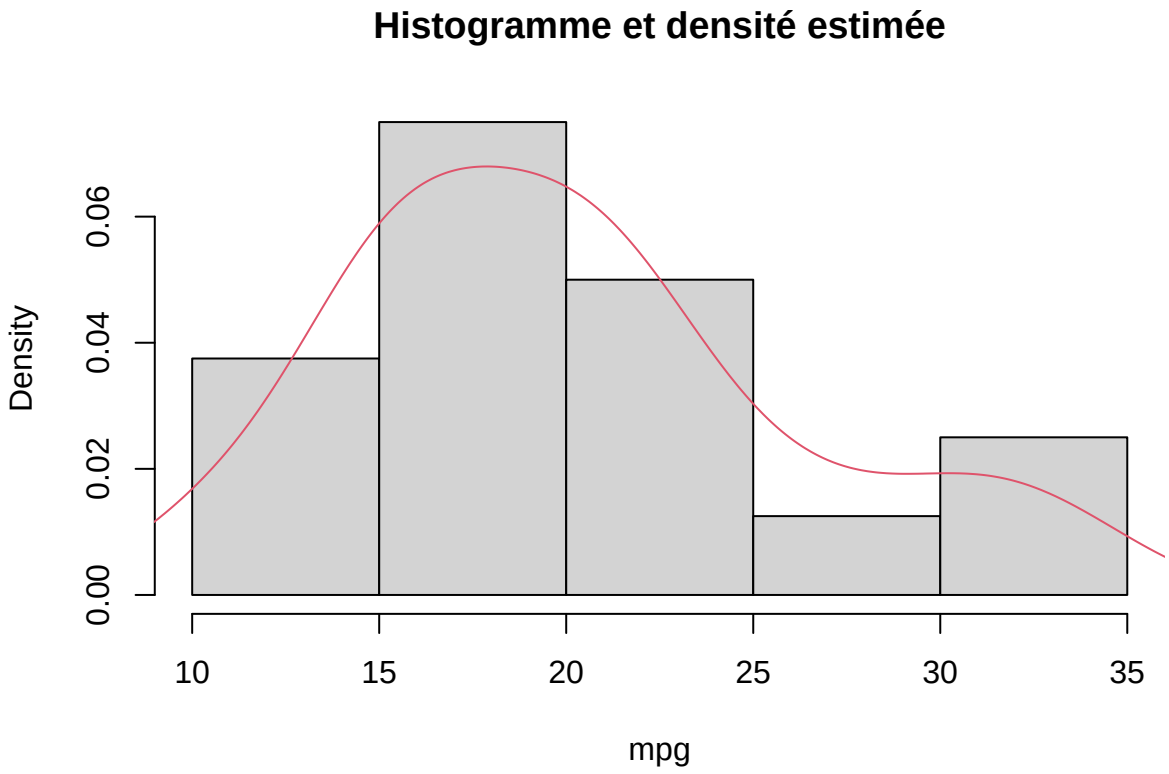
Pour superposer la courbe d'une densité donnée :

```
mydata=rnorm(1000,0,1) # simulation de 1000 observations N(0,1)
hist(mydata,freq=F,main='Histogramme et densité théorique')
curve(dnorm, # la fonction de densité de N(0,1)
      from=-5,to=5, # le range
      add=TRUE) # parce qu'on veut ajouter la densité de N(0,1) à l'histogramme
```



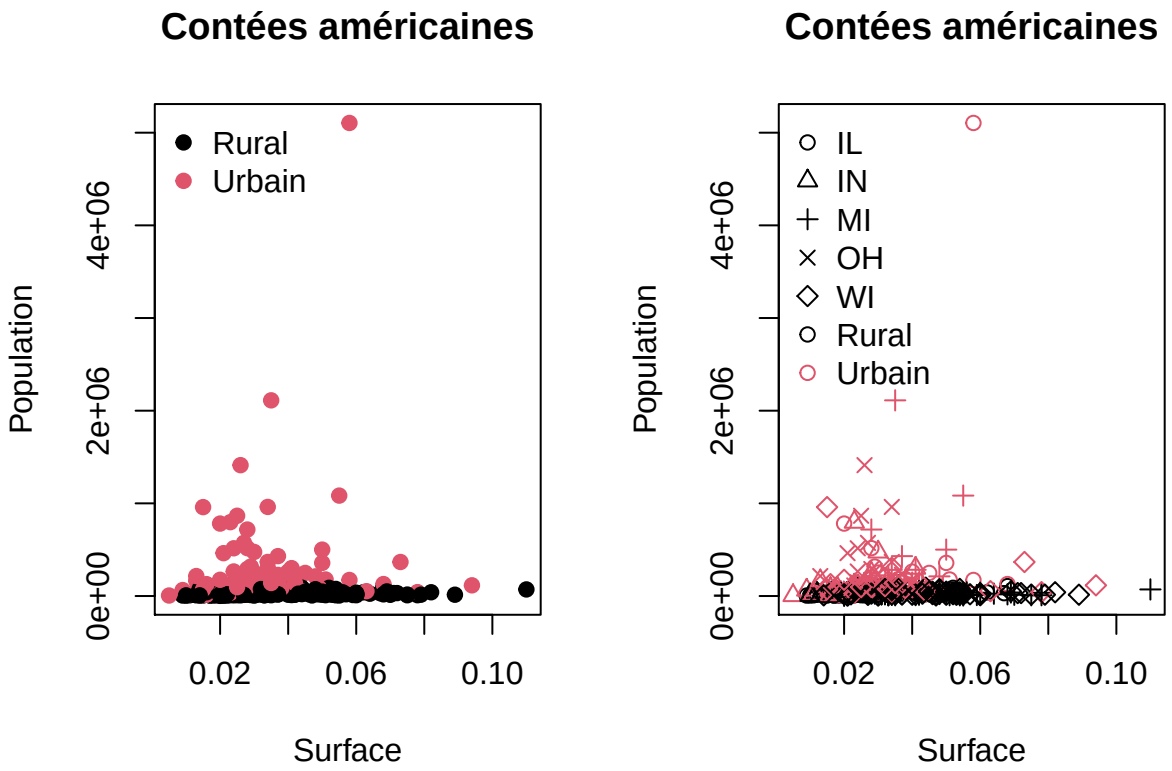
Plutôt que de visualiser l'histogramme des données on peut visualiser la densité estimée par noyau :

```
hist(mpg,freq=F,main='Histogramme et densité estimée')
lines(density(mpg),col=2)
```



5. Exercices

1. On travaille sur les données démographiques `midwest` du package `ggplot2` (un package graphique avancé). Consulter l'aide pour la description des variables. Reproduire les graphiques suivants en utilisant les fonctions graphiques de base



2. On travaille sur le jeu de données `adult` disponible sur le site : archive.ics.uci.edu/ml/datasets/Adult. Il y a 14 colonnes dans cette table. Vous devez commencer par importer les données dans `R` (fichier `adult.data`). Pour les noms des variables, regarder dans le fichier `adult.names`. Répondre aux questions suivantes en rédigeant un rapport `R Markdown`.
   a. Décrire chaque variable de manière appropriée, selon le type de la variable.
   b. Décrire le lien entre les variables `age` et `class`. Il existe plusieurs moyens de représenter chaque lien ne pas hésiter à en essayer plusieurs pour trouver la plus représentative.