Aide à l'utilisation de **R** pour représenter graphiquement les données

Marie Laure Delignette-Muller

26 mars 2018

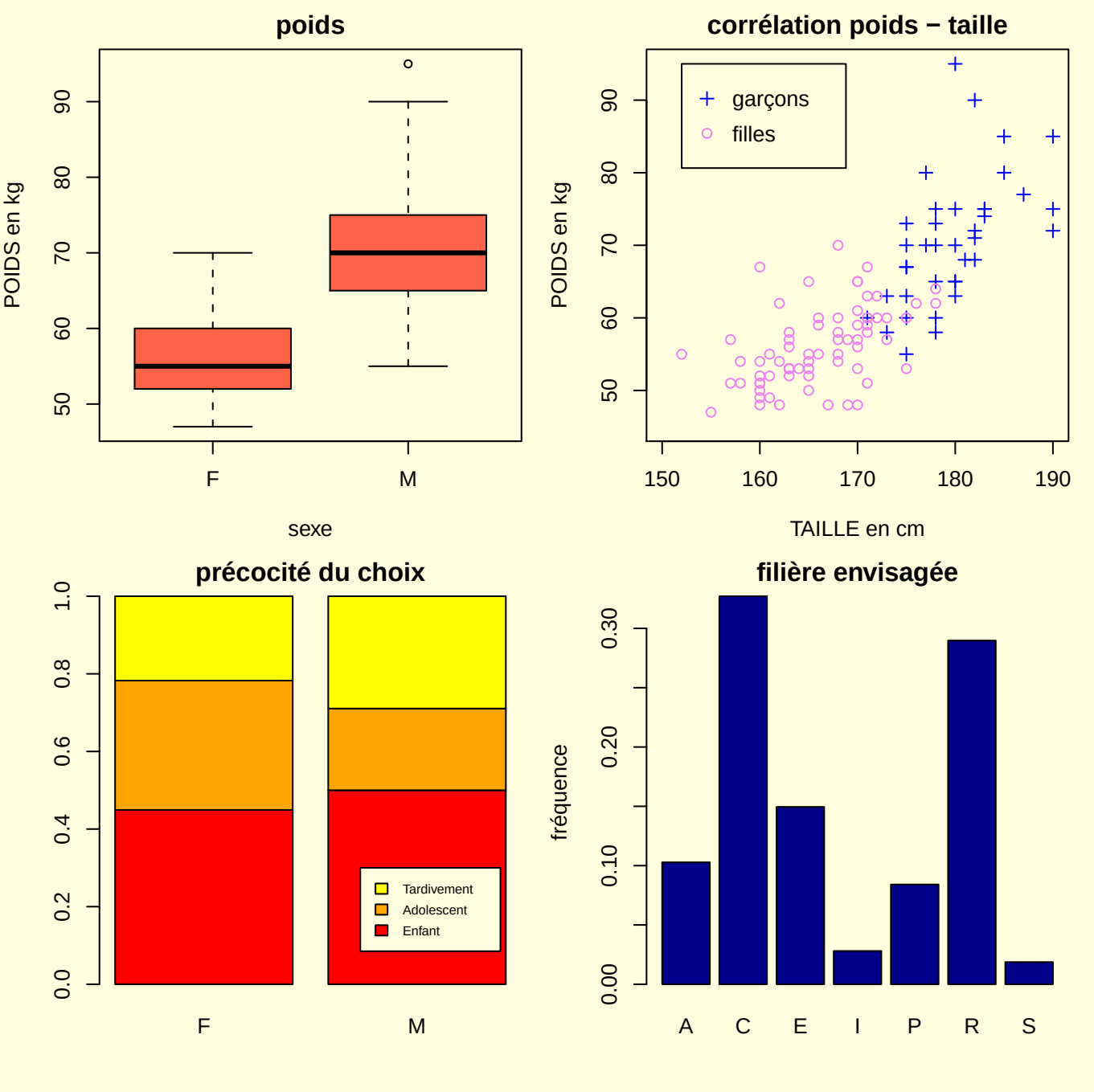


Table des matières

1 Introduction **3**
- 1.1 Objectifs de ce document . . . 3
- 1.2 Rappels sur la gestion des graphes . . . 3
- 1.3 Un premier exemple . . . 3

2 Quelques représentations graphiques classiques **4**
- 2.1 L'histogramme de fréquence (une variable quantitative continue) . . . 4
- 2.2 Le diagramme des fréquences cumulées (une variable quantitative continue) . . . 5
- 2.3 Le diagramme en boîte ou boîte à moustaches et le `stripchart` (une variable quantitative continue assortie éventuellement d'une variable qualitative) . . . 6
- 2.4 La droite de Henry ou " normal Q-Q plot " (une variable quantitative continue) . . . 8
- 2.5 Le diagramme en secteurs (une variable qualitative) . . . 9
- 2.6 Le diagramme en bâtons (une variable qualitative ou une variable quantitative discrète) assortie éventuellement d'une variable qualitative) . . . 10
- 2.7 Les représentations en barres ou en bandes (2 variables qualitatives) . . . 11
- 2.8 Le diagramme de dispersion (deux variables quantitatives continues) . . . 12
- 2.9 Les diagrammes de dispersion des variables prises 2 à 2 (n variables quantitatives continues) . . . 13
- 2.10 Les diagrammes permettant de représenter plus de 2 variables sur un même graphe . . . 14

3 La personnalisation des graphes **15**
- 3.1 Pour modifier le type de tracé dans la fonction `plot` . . . 15
- 3.2 Pour modifier les étiquettes des axes . . . 16
- 3.3 Pour ajouter un titre ou un commentaire sur un graphe . . . 17
- 3.4 Pour intégrer, dans un titre, un commentaire ou une étiquette des symboles, indices ou exposants . . . 17
- 3.5 Pour changer le type de point ou de ligne et la couleur . . . 18
- 3.6 Pour fixer la zone de tracé (min et max sur les axes) . . . 20
- 3.7 Pour représenter 2 courbes sur la même figure et ajouter une légende . . . 20
- 3.8 Pour superposer 2 courbes sur la même figure avec des axes des ordonnées différents et ajouter un axe à droite . . . 21
- 3.9 Pour changer les marges . . . 22
- 3.10 Pour changer la taille des points tracés et du texte . . . 23
- 3.11 Pour changer le fond de la fenêtre graphique . . . 24
- 3.12 Comment intégrer tout cela pour créer un joli graphe . . . 25

1 Introduction

1.1 Objectifs de ce document

R offre la possibilité de créer une très grande variété de graphes, plus ou moins facilement suivant la complexité du graphe désiré. Pour avoir une idée des possibilités et des exemples de codes **R** correspondant, il suffit de taper la commande : `demo(graphics)`. L'objectif de ce document n'est pas de répertorier toutes les possibilités offertes (c'est mission impossible en quelques pages !) mais de vous familiariser avec la création et la gestion de quelques graphes classiques et de vous indiquer comment modifier un certain nombre d'options pour personnaliser vos graphes. L'aide en ligne vous permettra ensuite d'aller plus loin. Rappelons que vous pouvez en savoir plus sur une fonction graphique ou tout autre fonction en tapant le code suivant : `help(nomdelafonction)` ou de façon équivalente `?nomdelafonction` Une aide en ligne est aussi disponible en tapant le code : `help.start()`. Par ailleurs, un manuel introductif fourni par les auteurs du logiciel à l'adresse http://www.r-project.org/ dans la rubrique " Documentation " ainsi que le manuel rédigé en français par Emmanuel Paradis disponible dans la même rubrique apportent de nombreux renseignements concernant la réalisation de graphes avec **R**. La lecture de ce type de manuels est indispensable si l'on vise une réelle maîtrise de R. L'objectif du présent document est bien moindre. Il est de vous rendre capable de réaliser quelques représentations graphiques pour vos besoins propres même si vous ne maîtrisez complètement ni les bases de la programmation (algorithmie) ni le langage `R`.

*Ce document représente un complément au document " Aide à l'utilisation de **R** pour réaliser les tests paramétriques et non paramétriques ". Il suppose donc connues les notions décrites dans ce premier document.*

1.2 Rappels sur la gestion des graphes

Pour ouvrir une nouvelle fenêtre graphique on tape généralement la commande `x11()`. Sous Windows on peut aussi utiliser la commande `windows()` et sous Mac OS il faut en principe utiliser la commande `quartz()`. Cette fenêtre peut éventuellement être partitionnée pour y créer plusieurs graphes. La commande `par(mfrow=c(n1,n2))` permet par exemple de partitionner la fenêtre en n1*n2 graphes, avec un découpage en n1 graphes sur la verticale et en n2 graphes sur l'horizontale.

Une fois le ou les graphes réalisé(s), la fenêtre graphique peut être sauvée sous différents formats. La sauvegarde sous format " EPS " est de bonne qualité et souvent recommandée pour la publication d'articles. Pour sauver la fenêtre dans ce format et stocker le fichier correspondant sous le nom " nomfic.eps ", il suffit de taper la commande `dev.copy2eps(file="nomfic.eps")` suivie de `dev.off()`. On peut aussi sauver la fenêtre en format " JPEG " en tapant `dev.copy(device=jpeg,file="nomfic.jpg")` suivie de `dev.off()` ou en format "PDF" en tapant `dev.copy(device=pdf,file="nomfic.pdf")`. D'autres formats de sauvegarde sont disponibles. On peut consulter l'aide de la fonction `dev.copy()` pour les connaître.

Une autre façon d'exporter une figure consiste ouvrir directement le fichier dans lequel on veut exporter la figure, à l'aide de l'une des fonctions `jpeg`, `bmp`, `pdf`, etc. On trace ensuite la figure et on ferme le fichier avec la fonction `dev.off`, comme dans l'exemple suivant :

```
jpeg("toto.jpg",quality=100,width=20,height=20,units="cm",
    pointsize=12,res=300))
plot(.....)
dev.off()
```

1.3 Un premier exemple

Il est tout d'abord nécessaire d'importer à l'aide de la fonction `read.table` le jeu de données comportant les variables que l'on souhaite représenter (prenez garde à bien indiquer comme répertoire courant de **R** celui contenant ces fichiers pour que ces commandes marchent). Il peut ensuite être pratique d'utiliser la fonction `attach` qui permet d'appeler chaque variable uniquement par le nom de la colonne correspondante (ce que nous ferons ici) bien que ce ne soit pas recommandé si l'on travaille sur plusieurs jeux de données avec certains noms de colonnes identiques.

```
> ds <- read.table("datasaucissons.txt",header=TRUE)
> attach(ds)
```

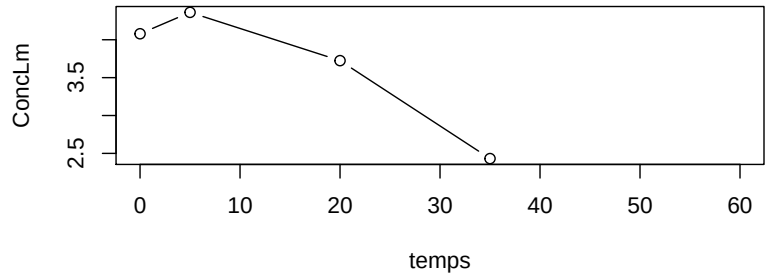
Il est important de vérifier la structure du jeu de données ce qui peut être fait rapidement à l'aide de la fonction `str`. Il convient notamment de vérifier que les variables qualitatives sont bien codées numériquement et que les variables qualitatives sont bien considérées comme des facteurs (sinon utilisez la fonction `as.factor` pour les transformer en facteurs).

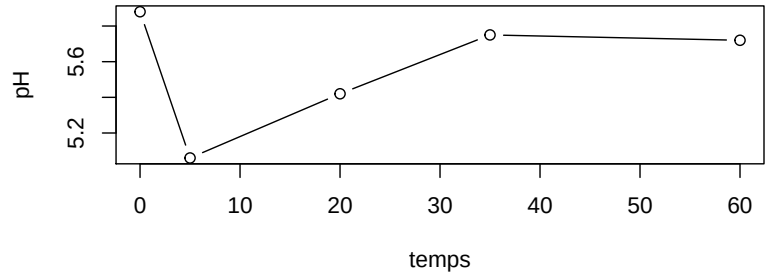
```
> str(ds)
```

```
'data.frame':        5 obs. of  5 variables:
 $ temps : int  0 5 20 35 60
 $ ConcLm: num  4.08 4.36 3.72 2.43 NA
 $ poids : int  400 352 286 271 NA
 $ pH    : num  5.88 5.06 5.42 5.75 5.72
 $ aw    : num  0.96 0.945 0.915 0.815 0.76
```

On peut ensuite représenter les variables qui nous intéressent dans le tableau de données.

```
> par(mfrow=c(2,1)) # partage de la fenêtre en 2 lignes et 1 colonne
> plot(temps,ConcLm,type='b') # type='b'signifie qu'on veut des points reliés par des lignes
> plot(temps,pH,type='b')
```





2 Quelques représentations graphiques classiques

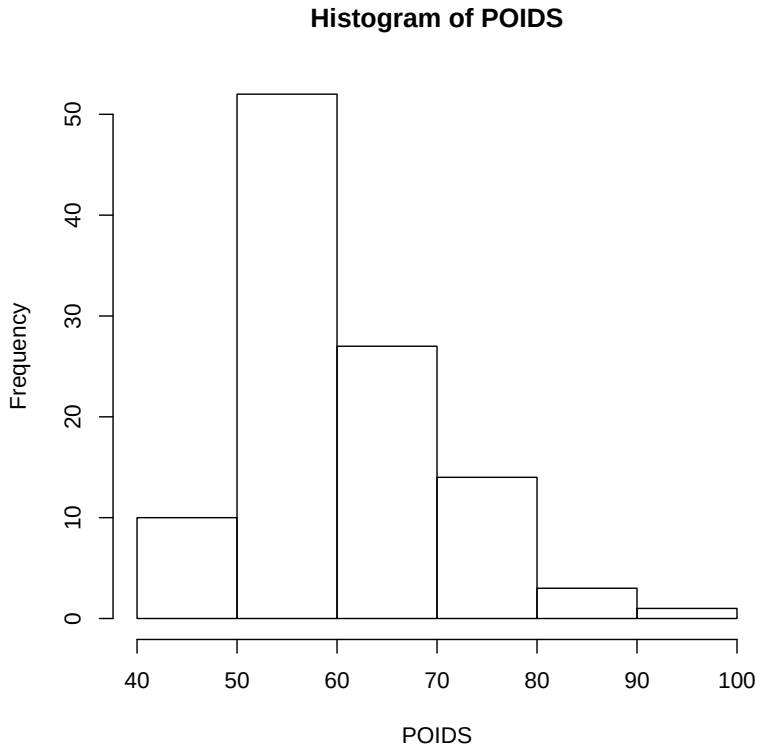
La quasi-totalité des exemples présentés dans cette partie utilise le même fichier de données "ENQ9697.txt" qu'il est nécessaire d'importer avant de tester les exemples (cf. commandes suivantes) :

```
> d <- read.table("ENQ9697.txt",header=TRUE)
> attach(d)

> str(d)

'data.frame':        107 obs. of  11 variables:
 $ SEXE      : Factor w/ 2 levels "F","M": 1 2 1 2 1 1 2 1 1 1 ...
 $ AGE       : int  22 21 19 20 19 21 21 19 20 22 ...
 $ POIDS     : int  53 67 63 60 48 58 77 61 52 70 ...
 $ TAILLE    : int  175 175 172 175 167 171 187 170 161 168 ...
 $ CADRE     : Factor w/ 2 levels "C","V": 2 1 2 2 2 1 1 1 1 1 ...
 $ DECISION  : Factor w/ 3 levels "A","E","T": 2 1 1 1 1 2 1 1 2 2 ...
 $ ENTOURAGE : Factor w/ 2 levels "N","O": 2 1 2 2 1 1 1 1 1 1 ...
 $ STAGE     : Factor w/ 2 levels "N","O": 2 2 1 2 2 2 1 2 2 2 ...
 $ FORMATION : Factor w/ 2 levels "A","C": 2 2 2 2 2 2 2 2 2 2 ...
 $ MOTIVATION: Factor w/ 5 levels "C","D","L","R",..: 2 4 1 3 5 1 5 1 5 2 ...
 $ FILIERE   : Factor w/ 7 levels "A","C","E","I",..: 6 6 3 2 6 1 2 1 6 6 ...
```

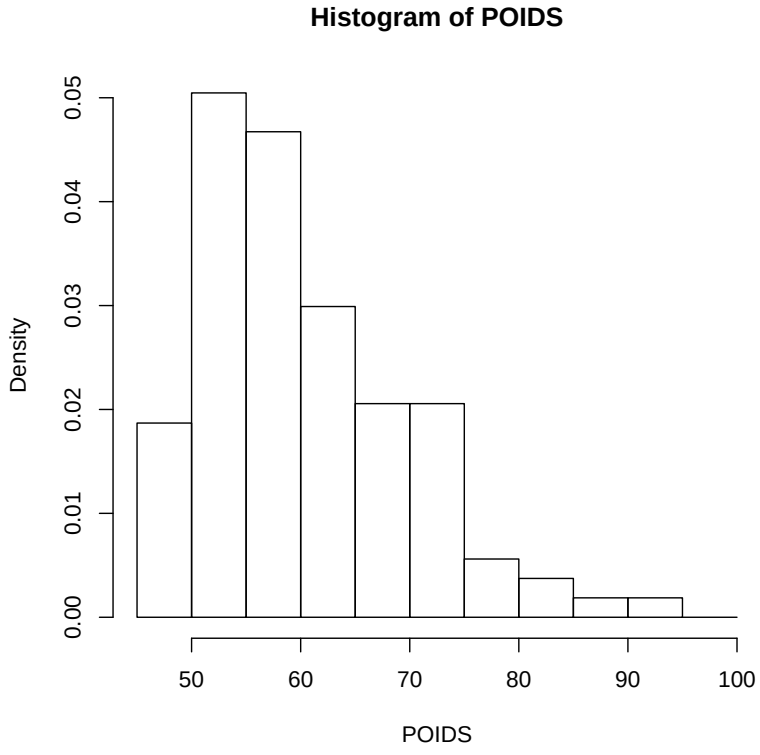
2.1 L'histogramme de fréquence (une variable quantitative continue)

La fonction `hist` permet de tracer directement un histogramme de fréquences pour un vecteur correspondant à une variable quantitative continue. Cette fonction peut être paramétrée par le nombre de classes (`breaks=n`) ou directement par un vecteur contenant les valeurs numériques délimitant les classes (`breaks=c(lim1,lim2,lim3,`$\cdots$`,limk)`). Par défaut, l'histogramme est représenté en effectifs. On peut le représenter en fréquences relatives en paramétrant la fonction par `freq=FALSE`. Par défaut chaque classe est définie en incluant la limite supérieure ($]lim_i; lim_{i+1}]$). Voici deux exemples de tracés d'histogrammes.

```
> hist(POIDS,breaks=6)
```




```
> hist(POIDS,breaks=c(45,50,55,60,65,70,75,80,85,90,95,100),freq=F)
```

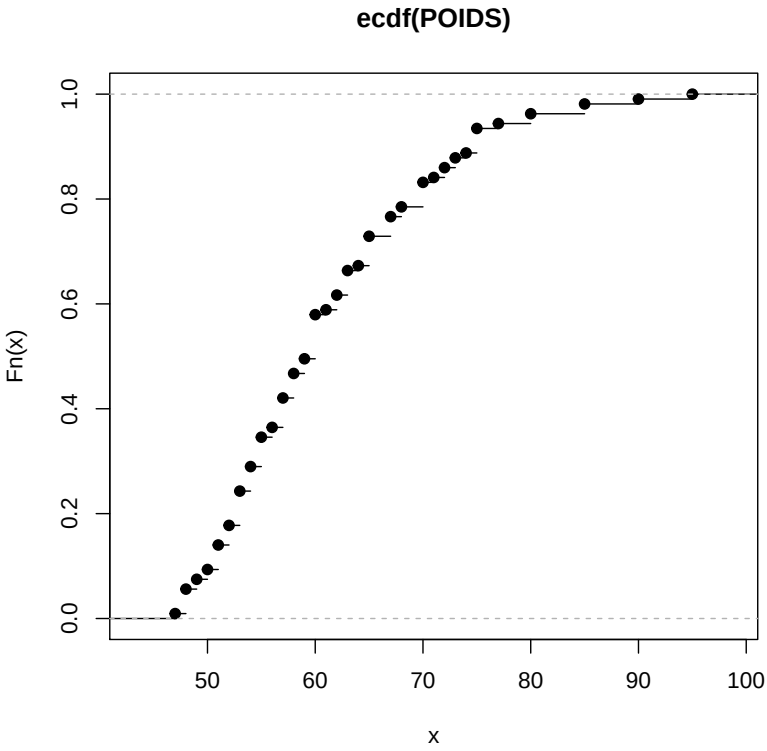


2.2 Le diagramme des fréquences cumulées (une variable quantitative continue)

Au moins deux représentations peuvent être proposées en utilisant la fonction `ecdf` qui calcule les fréquences cumulées :

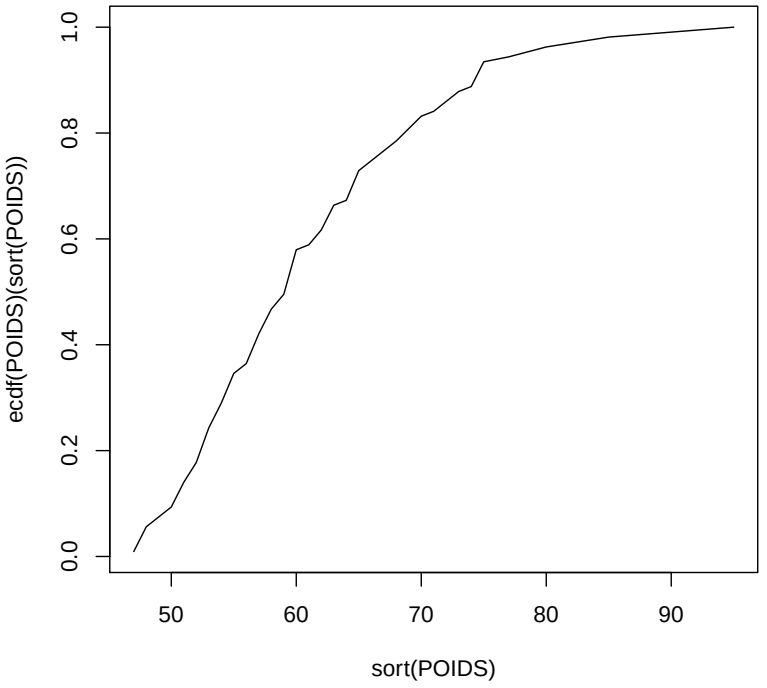
— soit la représentation fournie par défaut :

```
> plot(ecdf(POIDS))
```



— soit une représentation avec interpolation linéaire qui nécessite l'utilisation de la fonction `sort` de tri des observations par ordre croissant :

```
> plot(sort(POIDS),ecdf(POIDS)(sort(POIDS)),type="l")
```

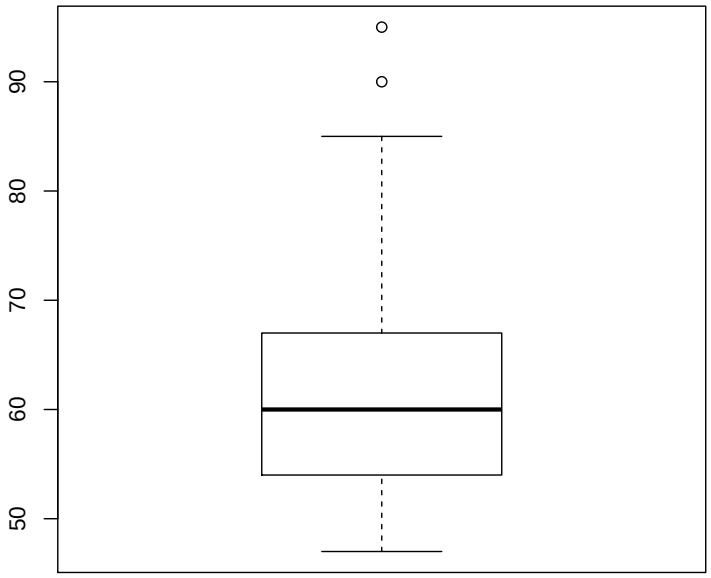


2.3 Le diagramme en boîte ou boîte à moustaches et le `stripchart` (une variable quantitative continue assortie éventuellement d'une variable qualitative)

La fonction boxplot() permet de tracer un ou plusieurs diagramme(s) en boîte (cf. exemples ci-dessous). Les points extrêmes (définis par défaut dans cette fonction comme ceux étant à une distance de plus de 1.5 fois la taille de la boîte d'un des bords de la boîte) sont représentés en points isolés. Il est possible de modifier cette définition des points extrêmes en changeant la valeur de l'argument `range` de la fonction.

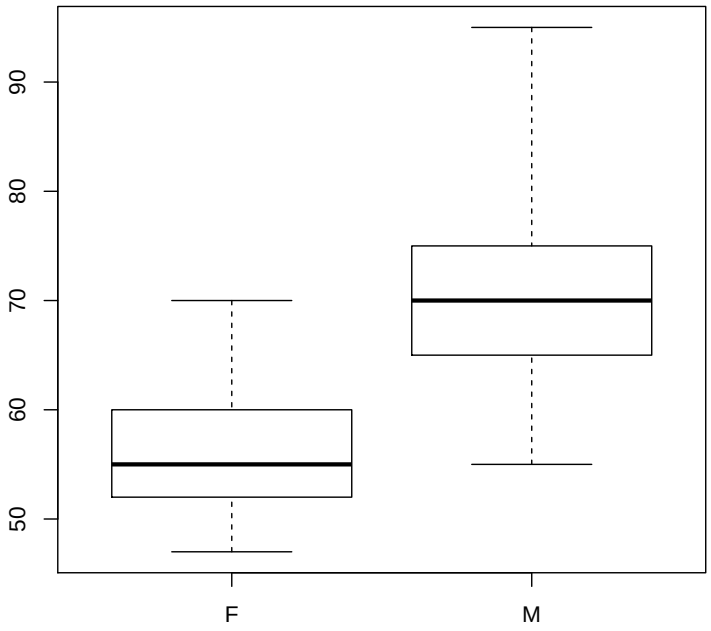
— Voici le diagramme en boîte du poids des étudiants.

```
> boxplot(POIDS)
```



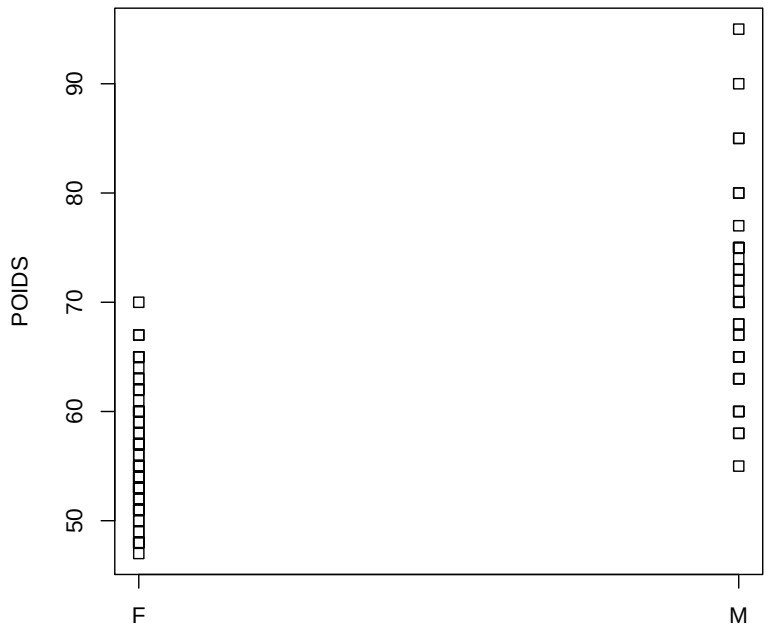
— Voici les diagrammes en boîte du poids des étudiants en fonction de leur sexe, avec l'argument `range` augmenté de façon à intégrer tous les points observés dans les moustaches.

```
> boxplot(POIDS~SEXE,range=2)
```



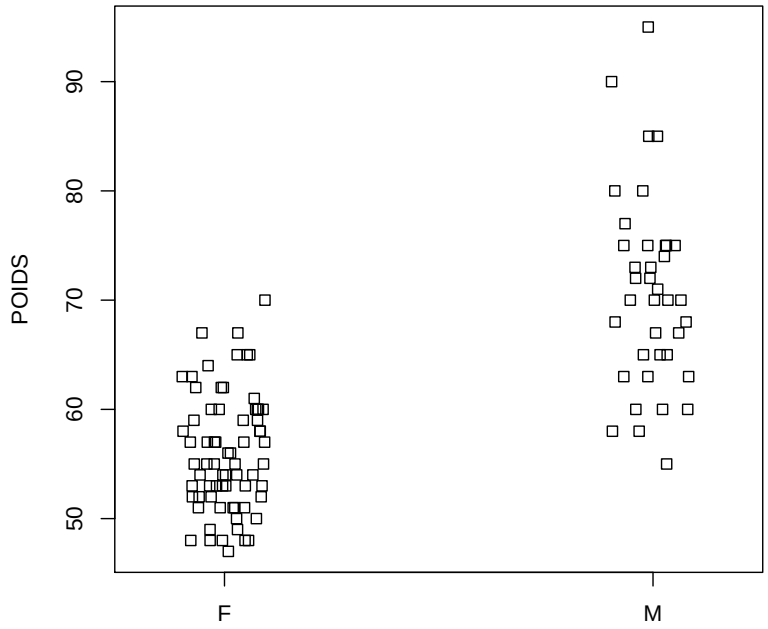
— Il peut aussi être intéressant de reporter toutes les valeurs observées, notamment lorsque les effectifs sont très petits.

```
> stripchart(POIDS~SEXE,vertical=TRUE)
```



— Enfin il est possible de représenter les ex-aequos sous forme d'amas de points.

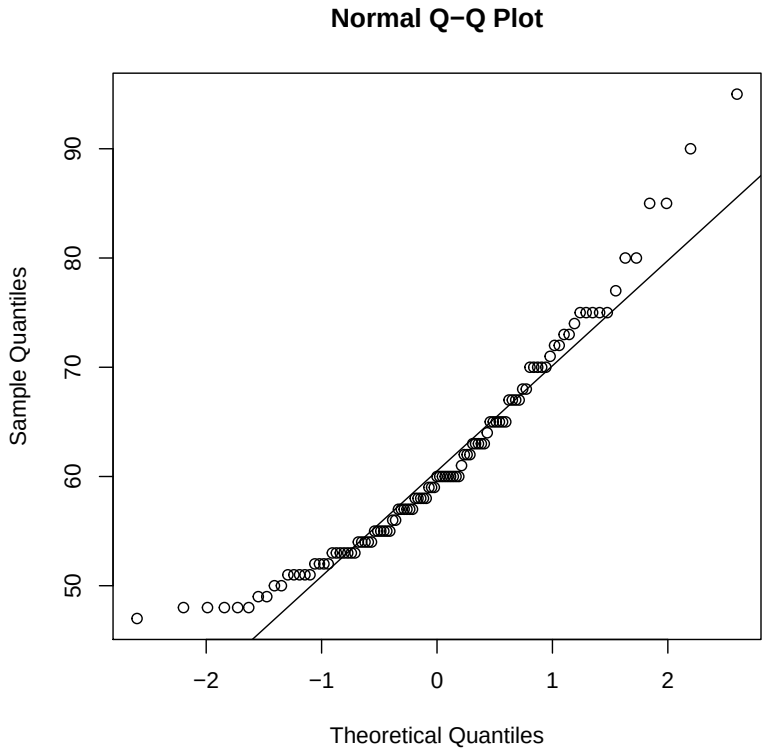
```
> stripchart(POIDS~SEXE,vertical=T,method="jitter")
```



2.4 La droite de Henry ou " normal Q-Q plot " (une variable quantitative continue)

La fonction `qqnorm` permet de tracer les quantiles observés en fonction des quantiles de la loi normale $N(0,1)$. Rappelons que pour une distribution observée normale les points obtenus sont quasiment alignés. Ainsi cette représentation est souvent utilisée pour juger graphiquement de la normalité d'une distribution observée. Voici un exemple de tracé de la droite de Henry sur le poids, ici tous sexes confondus. La fonction `qqline` peut être utilisée pour ajouter à ce graphe une ligne qui passe par les premier et troisième quartiles afin d'aider à juger de la linéarité des points du graphe.

```
> qqnorm(POIDS)
> qqline(POIDS)
```



2.5 Le diagramme en secteurs (une variable qualitative)

Il est important de noter que l'utilisation cette représentation n'est pas toujours recommandée. Elle est en effet jugée mauvaise par de nombreux statisticiens considérant que l'oeil humain sait mieux juger d'une mesure linéaire que d'une mesure angulaire ou de surface. Néanmoins, si l'on veut tout de même réaliser un tel diagramme, on peut utiliser la fonction `pie`. Si l'on part des données brutes, il est tout d'abord nécessaire de créer la table des effectifs par classe en utilisant la fonction `table` (cf. exemple ci-dessous sur le moment du choix professionnel des étudiants vétérinaires).

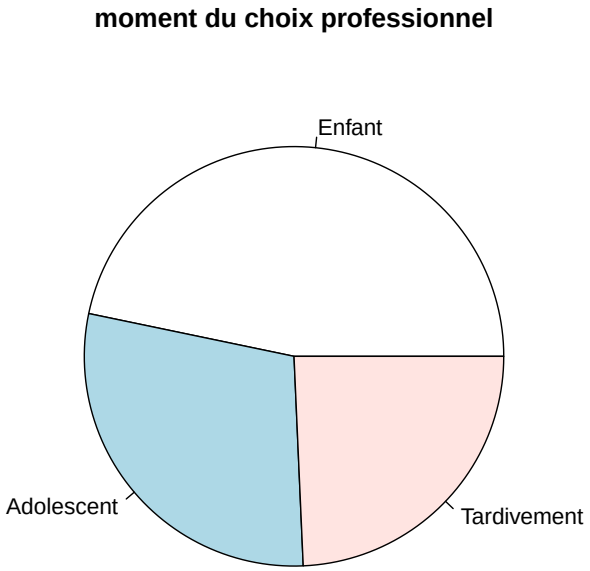
```
> tdec<-table(DECISION)
> tdec

DECISION
 A  E  T
31 50 26
```

Dans cet exemple les modalités A, E et T correspondent respectivement à Adolescent, Enfant et Tardivement. On peut remarquer que les modalités du facteur `DECISION` sont par défaut classées par ordre alphabétique dans la table ce qui ne conviendra pas bien pour une représentation graphique ou l'on attend ici un ordre plus logique E, A et T. On peut alors changer cet ordre en recréant un facteur à partir du premier et en changeant l'ordre des niveaux de la façon suivante :

```
> DECISION2 <- factor(DECISION,levels=c("E","A","T"))
```

On peut d'ailleurs aussi modifier le nom des niveaux pour les rendre plus explicites et enfin reconstruire la table à partir de ce nouveau facteur :

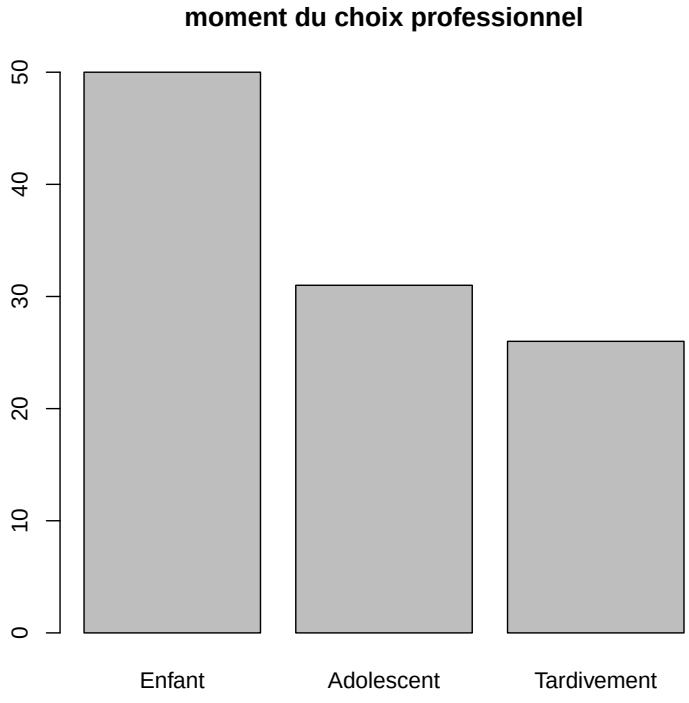
```
> levels(DECISION2) <- c("Enfant","Adolescent","Tardivement")
> tdec2 <- table(DECISION2)
> tdec2

DECISION2
     Enfant  Adolescent Tardivement
         50          31          26

> pie(tdec2,main="moment du choix professionnel")
```



2.6 Le diagramme en bâtons (une variable qualitative ou une variable quantitative discrète) assortie éventuellement d'une variable qualitative)

Le diagramme en bâtons est souvent préféré au diagramme en secteurs, même pour une variable qualitative, considérant que l'oeil humain sait mieux juger d'une mesure linéaire que d'une mesure angulaire. La fonction `barplot` permet de créer un tel graphe. Comme dans le cas précédent, si l'on part des données brutes, il est tout d'abord nécessaire de créer la table des effectifs par classe en utilisant la fonction `table` (cf. exemple précédent).

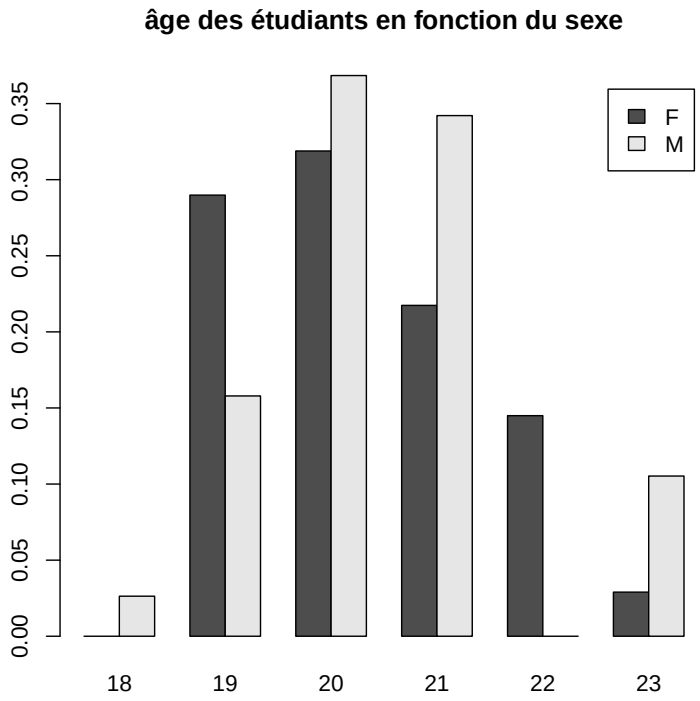
```
> barplot(tdec2,main="moment du choix professionnel")
```



Dans un diagramme en bâtons, peuvent être représentés sur l'axe des ordonnées, les effectifs ou les fréquences. Dans le deuxième cas, il est nécessaire de calculer au préalable ces fréquences par exemple en utilisant la fonction `prop.table` qui calcule les fréquences sur chaque ligne (si `margin` fixé à 1) sou sur chaque colonne (si `margin` fixé à

2) d'une table. Ceci est nécessaire notamment si l'on veut représenter plusieurs diagrammes en bâtons sur le même graphe comme ci-dessous.

```
> tage <- table(SEXE,AGE)
> tage

    AGE
SEXE 18 19 20 21 22 23
   F  0 20 22 15 10  2
   M  1  6 14 13  0  4

> tage2 <-prop.table(tage,margin=1)
> tage2

    AGE
SEXE         18         19         20         21         22         23
   F 0.00000000 0.28985507 0.31884058 0.21739130 0.14492754 0.02898551
   M 0.02631579 0.15789474 0.36842105 0.34210526 0.00000000 0.10526316

> barplot(tage2,beside=TRUE,legend.text=TRUE,main="âge des étudiants en fonction du sexe")
```



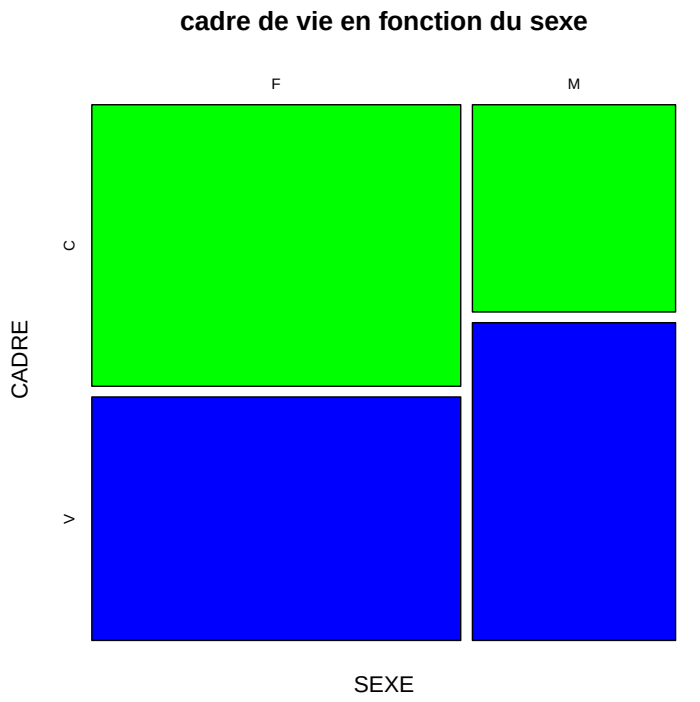
2.7 Les représentations en barres ou en bandes (2 variables qualitatives)

Lorsque l'on souhaite représenter la distribution d'une variable qualitative en fonction d'une deuxième variable qualitative, on peut aussi tracer des représentations en barres (ou en bandes), toujours à l'aide de la fonction barplot(). En partant des données brutes, il faut utiliser au préalable la fonction table() pour calculer les effectifs de la table de contingence associée aux 2 variables qualitatives, puis diviser les effectifs de chaque colonne par l'effectif total de la colonne afin d'obtenir des distributions en fréquences par colonne. Pour cela on peut à nouveau utiliser la fonction `prop.table` mais cette fois en lui mettant un deuxième argument `margin=2` pour qu'il fasse bien le calcul par colonne et non par ligne comme dans l'exemple précédent.

```
> tcadsexe <- table(CADRE,SEXE)
> tcadsexe

     SEXE
CADRE  F  M
    C 37 15
    V 32 23

> tcadsexe2 <- prop.table(tcadsexe,margin=2)
> barplot(tcadsexe2,main="cadre de vie en fonction du sexe",legend.text=TRUE)
```

On peut aussi tout simplement utiliser la représentation par défaut d'une table de contigence qui donne des bandes ou barres de largeur proportionnelle aux effectifs de chaque classe correspondant à chaque bande ou barre. Attention, par défaut `barplot` et `plot` ne représente pas la table dans le même sens, donc il convient de reconstruire la table dans l'autre sens si l'on veut obtenir le même type de graphe. On peut aussi ajouter de la couleur comme ci-dessous.

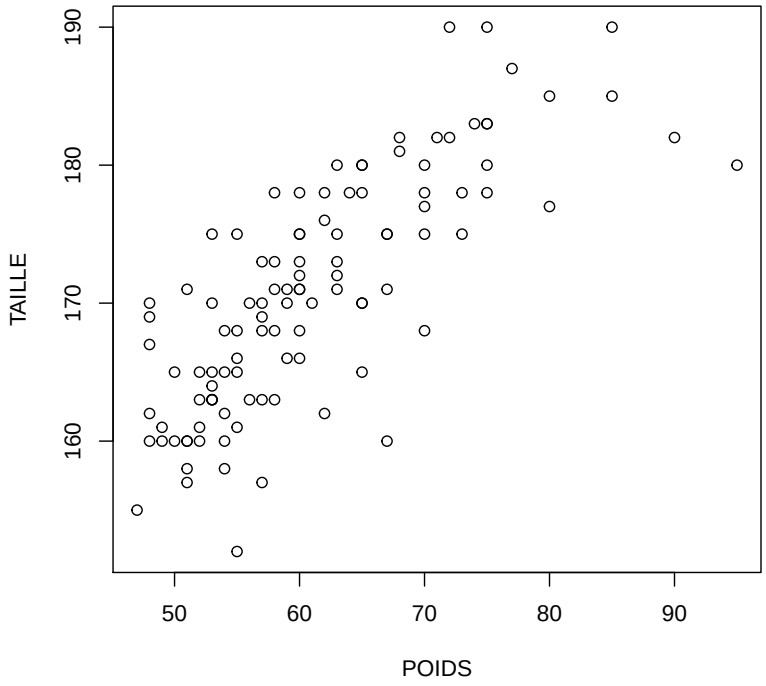
```
> tcadsexe3 <- table(SEXE,CADRE)
> plot(tcadsexe3,main="cadre de vie en fonction du sexe",col=c("green","blue"))
```



2.8 Le diagramme de dispersion (deux variables quantitatives continues)

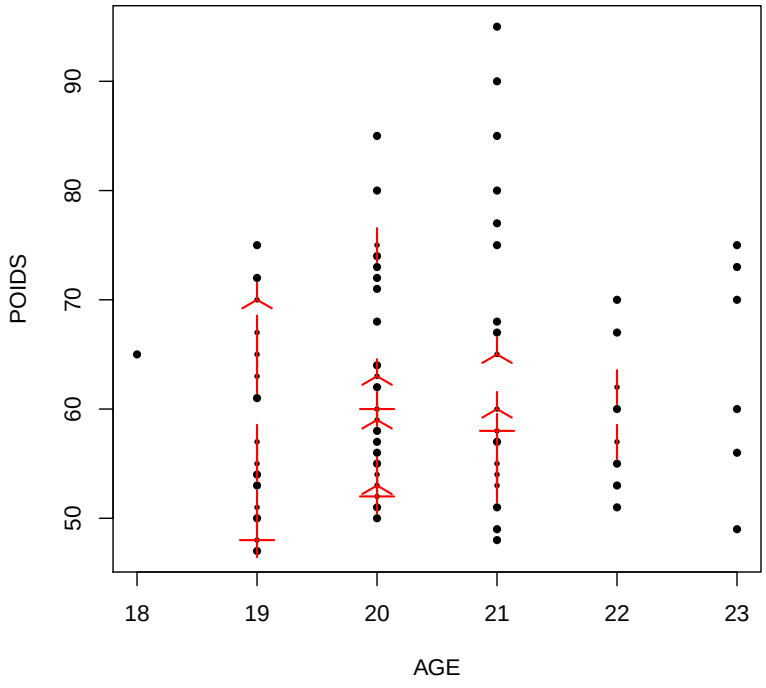
La fonction plot(), que l'on reverra en détail dans le chapitre sur la personnalisation des graphes, permet entre autres de tracer un nuage de points ou diagramme de dispersion.

```
> plot(POIDS,TAILLE)
```



Lorsqu'il y a des ex-aequos, il peut dans certains cas être intéressant d'utiliser la fonction `sunflowerplot` pour représenter les points correspondant à plusieurs observations par des soleils avec autant de rayons que d'observations.

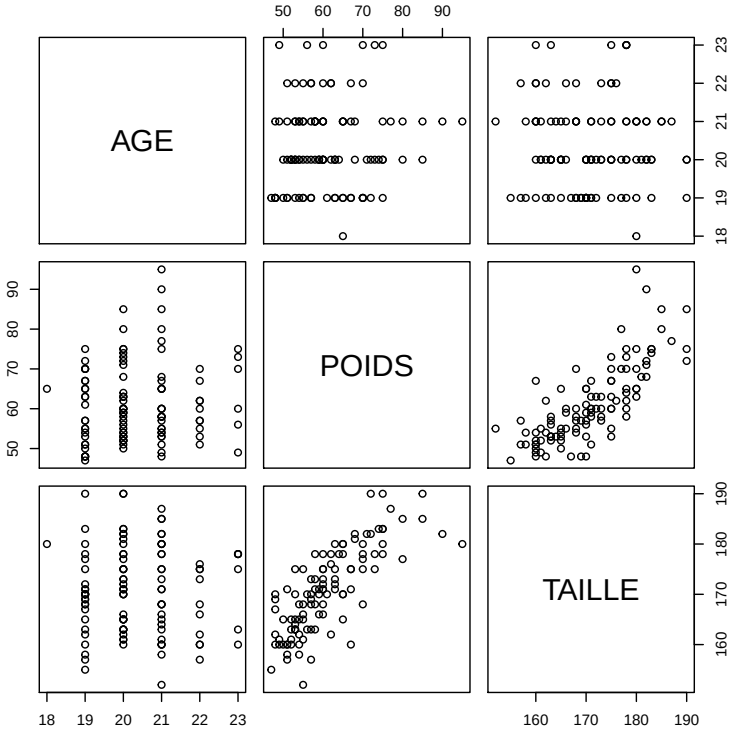
```
> sunflowerplot(AGE,POIDS)
```



2.9 Les diagrammes de dispersion des variables prises 2 à 2 (n variables quantitatives continues)

La fonction `pairs` permet à partir d'un jeu de données (objet de type `data.frame`) de tracer tous les diagrammes de dispersion des variables de ce jeu de données prises 2 à 2,comme ci-dessous sur le sous-jeu de données comprenant les variable AGE, POIDS et TAILLE.

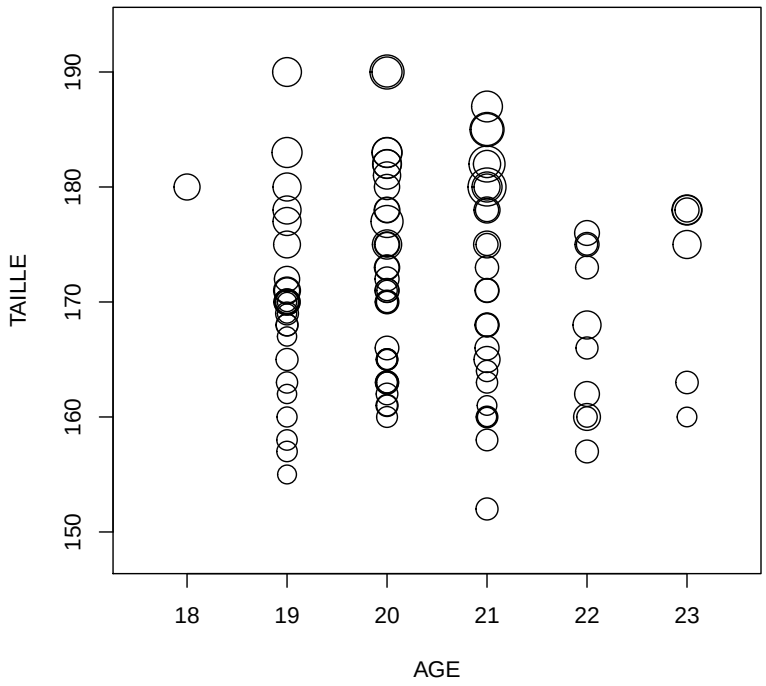
```
> pairs(d[,2:4])
```



2.10 Les diagrammes permettant de représenter plus de 2 variables sur un même graphe

La fonction `symbols` peut parfois être utile lorsque l'on veut représenter plus de deux variables quantitatives observées simultanément sur les mêmes unités d'observation. Elle offre un grand nombre de possibilités décrites dans l'aide en ligne. Voici un exemple, sans grand intérêt ici, mais qui donne une idée des possibilités offertes. Dans cet exemple est représenté le nuage de points de la taille en fonction de l'âge, avec des cercles dont le rayon est proportionnel au poids.

```
> symbols(AGE,TAILLE,circles=POIDS/500,inches=FALSE)
```

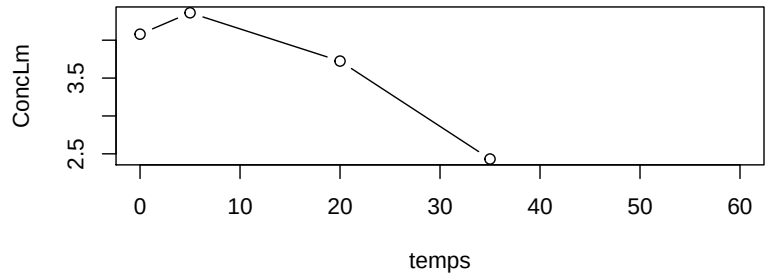


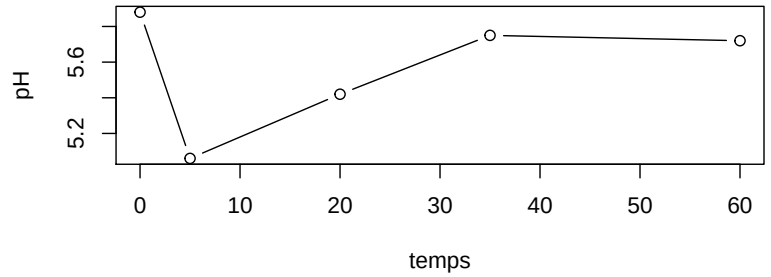
Il est aussi possible de jouer sur la couleur des points et les symboles utilisés pour différencier des observations correspondant à différentes modalités d'une variable qualitative, ce que nous verrons dans le cadre de la deuxième partie de ce document.

3 La personnalisation des graphes

Un nombre très important d'options permet de personnaliser les graphes à souhait. Dans cette partie sont présentées les plus couramment utilisées. Certaines options peuvent être paramétrées dès l'ouverture de la fenêtre graphique à l'aide de la fonction `par`, d'autres peuvent être paramétrées directement dans la fonction graphique (certaines peuvent l'être dans les deux). L'ensemble des paramètres modifiables par la fonction `par` peut être consulté dans l'aide de cette fonction (en tapant `?par`). Nous n'en présenterons que quelques uns. Toutes les options ne sont pas fonctionnelles pour toutes les fonctions graphiques : il faut parfois essayer pour voir ou consulter l'aide de R. Nous prendrons comme exemple de base la représentation de nuages de points et de courbes. Nous reprendrons comme base le premier exemple représenté au début de ce document.

```
> ds <- read.table("datasaucissons.txt",header=TRUE)
> attach(ds)

> par(mfrow=c(2,1)) # partage de la fenêtre en 2 lignes et 1 colonne
> plot(temps,ConcLm,type='b') # type='b'signifie qu'on veut des points reliés par des lignes
> plot(temps,pH,type='b')
```

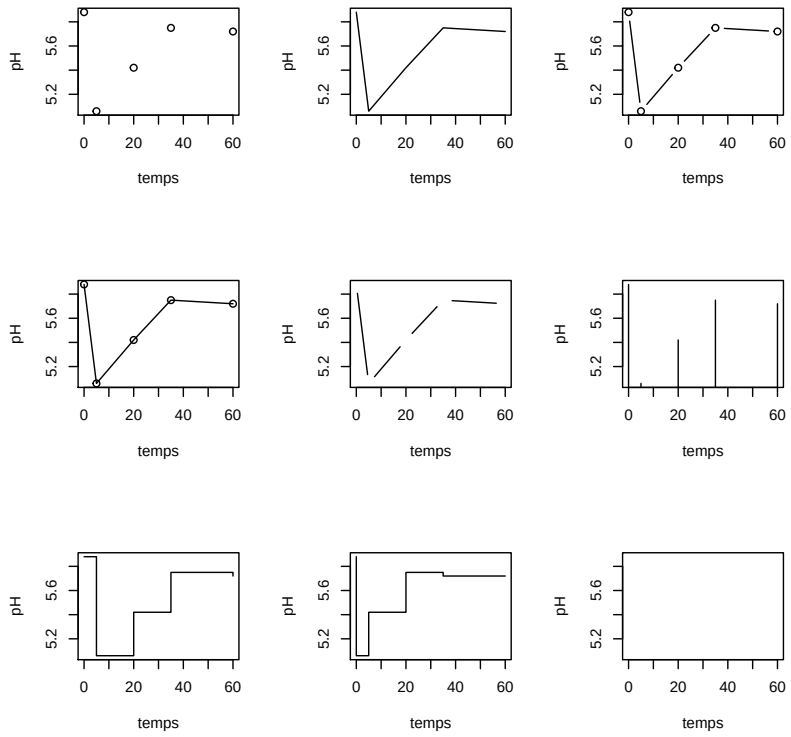




3.1 Pour modifier le type de tracé dans la fonction `plot`

La fonction `plot` dispose d'un argument `type` qui peut prendre différentes valeurs parmi lesquelles : `"p"` pour une représentation des points isolés, `"l"` pour une représentation des lignes reliant les points, `"b"` pour une représentation des 2 ("both"), `"o"` pour une représentation des 2 sans interruption des lignes, `"c"` pour des lignes interrompues au niveau des points, `"h"` pour des lignes verticales, `"s"` pour des marches d'escaliers, `"S"` pour d'autres marches d'escaliers définies différemment, et enfin `"n"` si l'on ne veut rien représenter (peut servir lorsqu'on veut juste représenter les axes) (cf. exemples ci-dessous pour le même jeu de données). Voici une illustration des divers tracés :

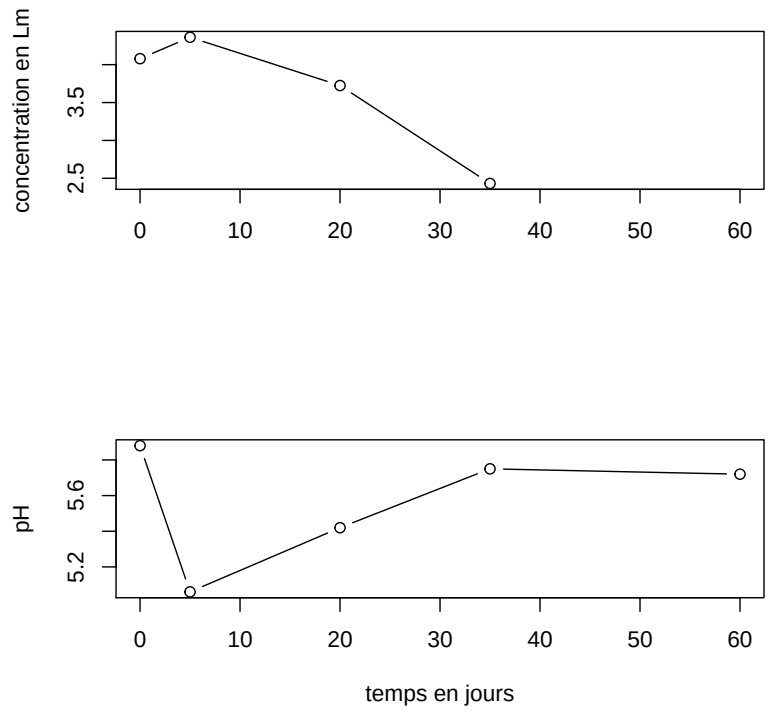
```
> par(mfrow=c(3,3))
> plot(temps,pH,type='p')
> plot(temps,pH,type='l')
> plot(temps,pH,type='b')
> plot(temps,pH,type='o')
> plot(temps,pH,type='c')
> plot(temps,pH,type='h')
> plot(temps,pH,type='s')
> plot(temps,pH,type='S')
> plot(temps,pH,type='n')
```



3.2 Pour modifier les étiquettes des axes

Imaginons que nous souhaitions, sur un graphe, modifier les étiquettes des axes qui sont mises par défaut par R à partir des noms des vecteurs colonnes dans le tableau de données initial. Il suffit alors de spécifier l'étiquette de l'axe des x (resp. des y) dans l'argument `xlab` (resp. `ylab`) dans la fonction `plot`, en mettant le texte souhaité entre guillemets (cf. exemple ci-dessous). Si l'on ne souhaite pas d'étiquette pour un axe, on fixe `xlab` (ou `ylab`) à `""`.

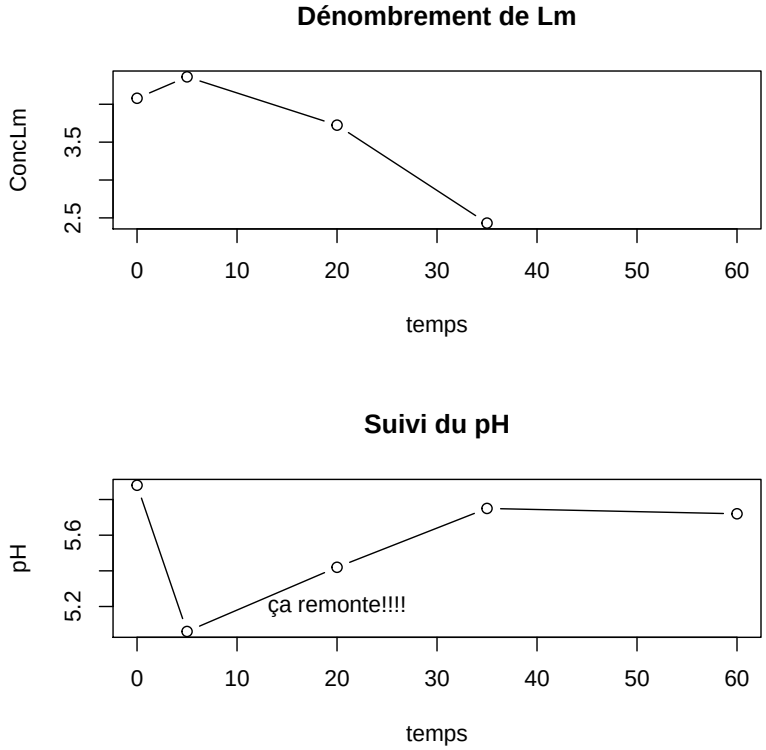
```
> par(mfrow=c(2,1))
> plot(temps,ConcLm,type="b",xlab="",ylab="concentration en Lm")
> plot(temps,pH,type="b",xlab="temps en jours",ylab="pH")
```



3.3 Pour ajouter un titre ou un commentaire sur un graphe

Pour ajouter un titre à un graphe, il suffit d'ajouter l'argument `main` dans la fonction `plot` et de le fixer au titre souhaité entre guillemets (cf. exemple ci-dessous). Si l'on souhaite ajouter un texte n'importe où sur la figure, on utilise la fonction `text` que l'on paramètre dans l'ordre par l'abscisse et l'ordonnée du point d'écriture et par le texte entre guillemets (cf. exemple ci-dessous).

```
> par(mfrow=c(2,1))
> plot(temps,ConcLm,type="b",main="Dénombrement de Lm")
> plot(temps,pH,type="b",main="Suivi du pH")
> text(20,5.2,"ça remonte!!!!")
```



Si une des échelles (x ou y) est non numérique on peut utiliser le code suivant pour connaître les limites utilisées sur l'échelle des x et des y.

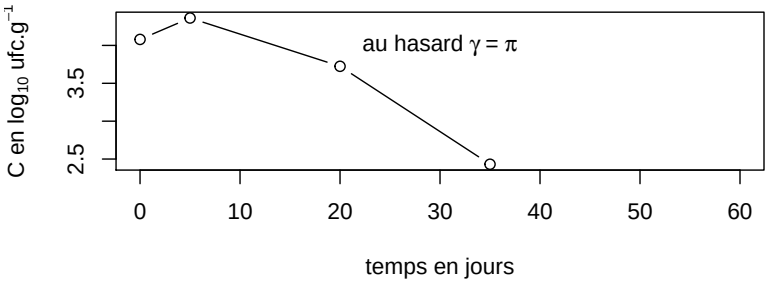
```
> barplot(tcadsexe2)
> par("usr")
```

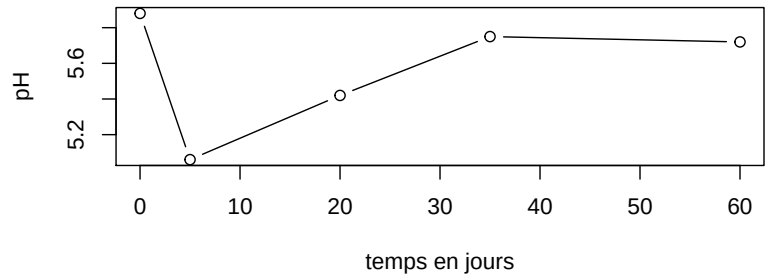
```
[1]  0.112  2.488 -0.010  1.000
```

3.4 Pour intégrer, dans un titre, un commentaire ou une étiquette des symboles, indices ou exposants

On peut dans un titre, un commentaire, une étiquette d'axe ou encore une légende (cf. plus loin) intégrer des symboles mathématiques ou des indices, des exposants, etc. Il est alors nécessaire d'utiliser de façon imbriquée 2 fonctions, la fonction `expression` et la fonction `paste`. La fonction `expression` permet d'interpréter, d'après des règles prédéfinies, des expressions de type formulation mathématique, et de les transcrire sous le format graphique souhaité. La fonction `paste` permet de définir un texte avec des morceaux qui seront interprétés par `expression` (ceux qui n'apparaissent pas entre guillemets) et d'autres qui ne le seront pas et seront retranscrits tels quels (ceux qui apparaissent entre guillemets). Pour comprendre, regardez attentivement l'exemple ci-dessous. Ensuite, pour savoir comment écrire les différentes formulations mathématiques afin qu'elles soient interprétables par `expression`, vous pouvez consulter les exemples accessibles sous **R** en tapant `demo(plotmath)` ou encore consulter l'aide de `plotmath` en tapant `?plotmath`.

```
> par(mfrow=c(2,1))
> plot(temps,ConcLm,type="b",xlab="temps en jours",ylab=expression(paste("C en ",log[10]," ufc.",g^-1)))
> text(30,4,expression(paste("au hasard ",gamma," = ",pi)))
> plot(temps,pH,type="b",xlab="temps en jours",ylab="pH")
```

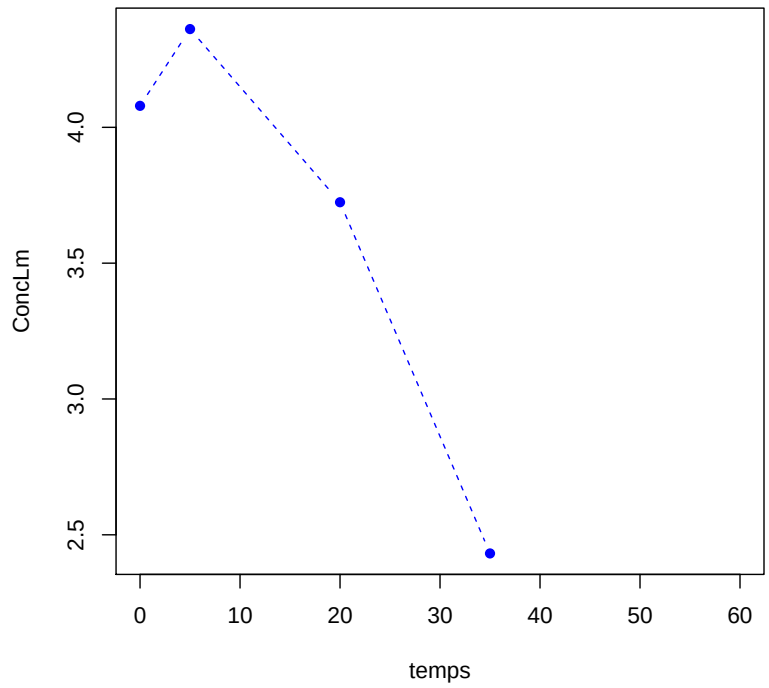




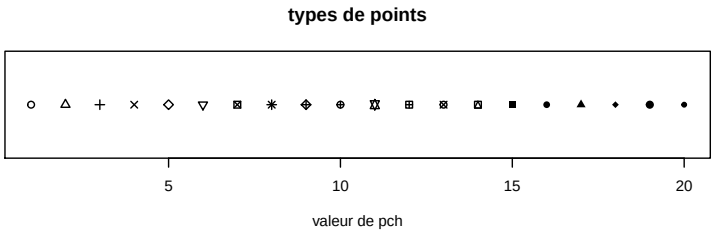
3.5 Pour changer le type de point ou de ligne et la couleur

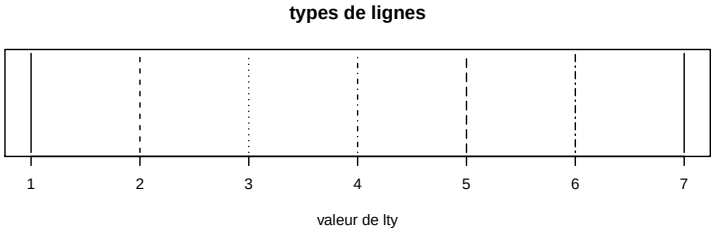
Dans la fonction `plot`, on peut paramétrer le type de point en fixant le paramètre `pch` à différentes valeurs allant de 1 à 20, le type de ligne en fixant le paramètre `lty` à différentes valeurs allant de 1 à 6 et la couleur en fixant le paramètre col à différentes valeurs allant par défaut de 1 à 8 (cf.exemple ci-dessous).

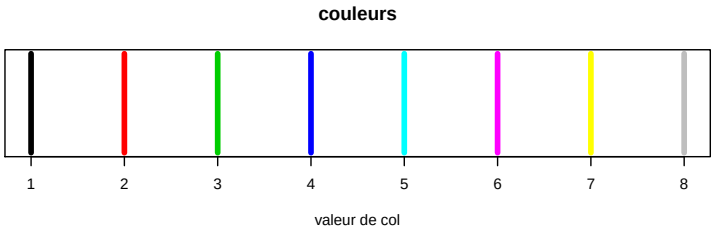
```
> plot(temps,ConcLm,type="b",pch=16,lty=2,col=4)
```



Les types de points, de lignes et les couleurs associés aux différentes valeurs de ces paramètres sont donnés dans la figure ci-dessous.

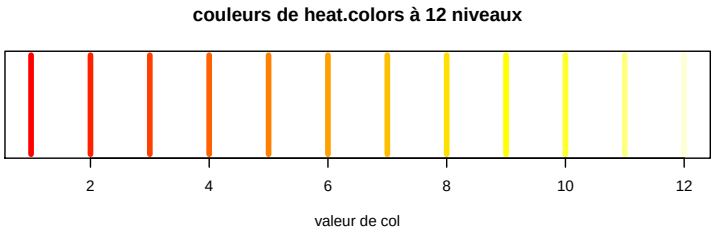


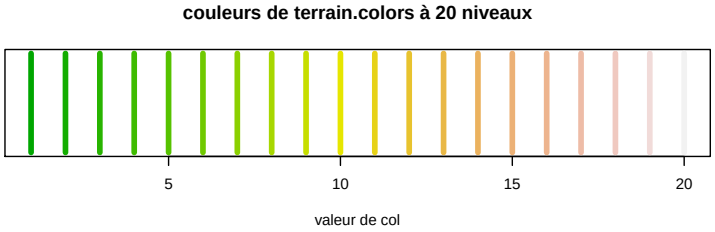


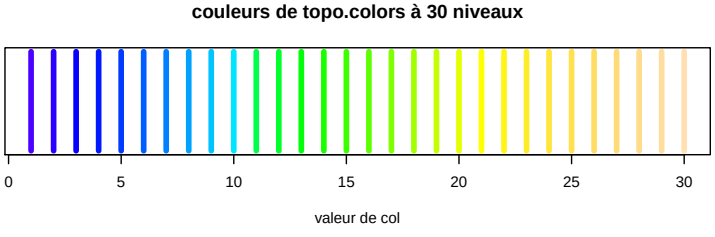


La palette des couleurs affectées par défaut aux nombres entiers peut être modifiée en utilisant la fonction `palette`. Les codes suivants remplacent la palette par défaut par différentes palettes définies dans **R** en changeant le nombre de niveaux, et le dernier code correspond au retour à la palette de défaut. Les couleurs des trois palettes de l'exemple sont représentées sur la figure suivante.

```
> palette(heat.colors(12))
> palette(terrain.colors(20))
> palette(topo.colors(30))
> palette("default")
```





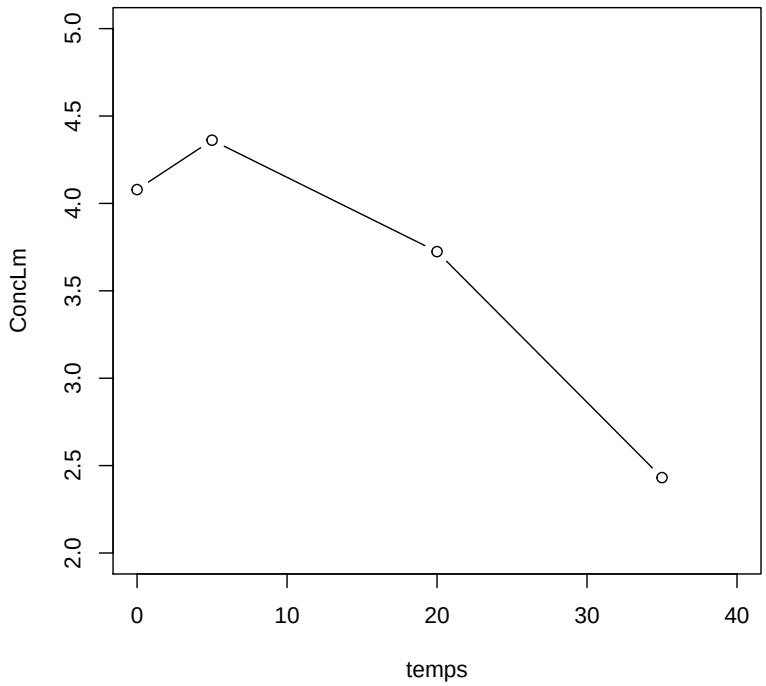


Le paramètre col peut aussi être fixé au nom entre guillemets d'une des très nombreuses couleurs prédéfinies dans R et dont on peut voir tous les noms en tapant `colors()` (exemple : `col="white"` ou `col="yellowgreen"` etc.) et la palette peut être définie avec comme argument de la fonction `palette` un vecteur de chaîne de caractères correspondant à des couleurs existantes (ex :. `palette(c("grey1","tomato2","paleturquoise2",···))`).

3.6 Pour fixer la zone de tracé (min et max sur les axes)

Il est possible de choisir la zone de tracé en fixant les paramètres xlim et ylim chacun au vecteur formé de sa valeur minimum et de sa valeur maximum (ex. : `xlim=c(xmin,xmax)`).

```
> plot(temps,ConcLm,type="b",xlim=c(0,40),ylim=c(2,5))
```

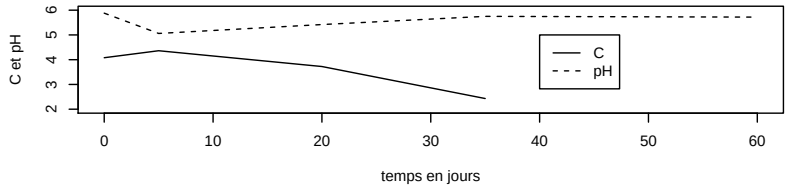


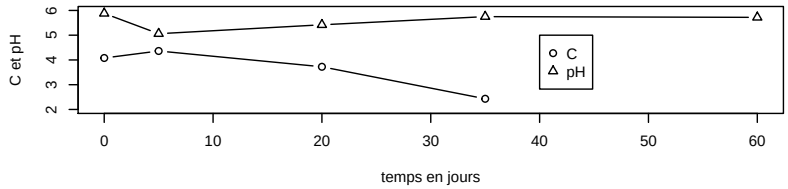
3.7 Pour représenter 2 courbes sur la même figure et ajouter une légende

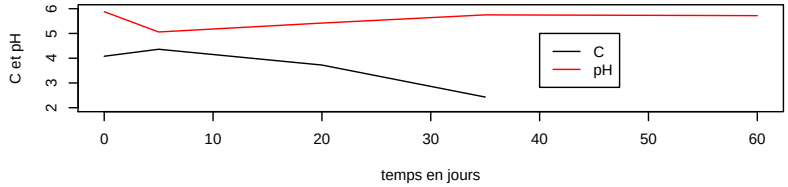
Lorsqu'on utilise la fonction `plot` à partir d'une fenêtre graphique sur laquelle une courbe a déjà été réalisée, l'ancienne courbe est effacée et remplacée par la nouvelle. Si l'on ne veut pas effacer l'ancienne courbe, mais superposer une nouvelle sur l'ancienne, il ne faut donc pas utiliser la fonction `plot`, mais la fonction `points` qui elle n'écrase pas ce qui est déjà sur le graphe (cf. exemple ci-dessous). Cette fonction s'utilise à peu près comme la fonction `plot` mis à part qu'elle ne gère pas les paramètres généraux du graphe (comme `xlab`, `main` etc.) qui doivent être paramétrés lors de la réalisation du premier graphe. Afin de distinguer les deux (ou n) courbes représentées, il est judicieux de les tracer avec des types de points ou de lignes ou des couleurs différentes. On peut ensuite ajouter au graphe une légende mettant en lien ces différentes caractéristiques et ce qu'elles représentent.

La fonction `legend` doit alors être utilisée. Celle-ci est paramétrée tout d'abord par l'abscisse et l'ordonnée du point sur le graphe où l'on veut faire démarrer cette légende. Le premier argument `x` de la fonction `legend` peut aussi être fixé à `"bottomleft", "topright", "center", ...` pour indiquer qualitativement où l'on souhaite mettre cette légende. L'argument `legend` (généralement troisième) correspond au vecteur contenant les étiquettes associées à chaque type de points ou de ligne ou chaque couleur. Enfin, suivant le type de différenciation entre courbes, un argument `pch` (type de points) ou `lty` (type de lignes), `col` (couleur de tracé) ou `fill` (couleur de remplissage) doit être affecté au vecteur contenant dans le même ordre les valeurs de pch, lty ou col employés. Regardez les trois exemples suivants pour mieux comprendre.

```
> par(mfrow=c(3,1))
> plot(temps,ConcLm,type="l",ylim=c(2,6),xlab="temps en jours",ylab="C et pH")
> points(temps,pH,type="l",lty=2)
> legend(40,5,legend=c("C","pH"),lty=c(1,2))
> plot(temps,ConcLm,type="b",pch=1,ylim=c(2,6),xlab="temps en jours",ylab="C et pH")
> points(temps,pH,type="b",pch=2)
> legend(40,5,legend=c("C","pH"),pch=c(1,2))
> plot(temps,ConcLm,type="l",col=1,ylim=c(2,6),xlab="temps en jours",ylab="C et pH")
> points(temps,pH,type="l",col=2)
> legend(40,5,legend=c("C","pH"),lty=1,col=c(1,2))
```



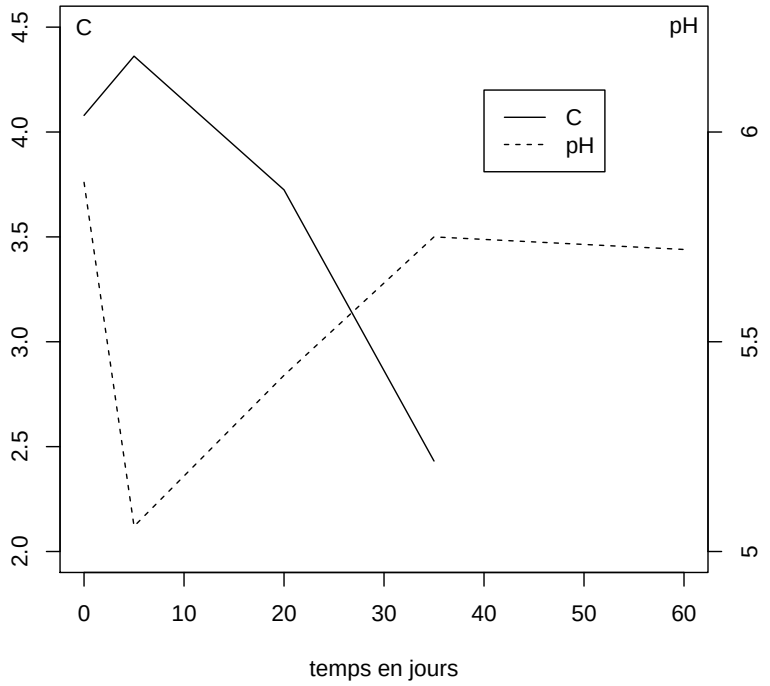




3.8 Pour superposer 2 courbes sur la même figure avec des axes des ordonnées différents et ajouter un axe à droite

Lorsque les courbes que l'on veut représenter sur le même graphe ne correspondent pas à la même variable mesurée, et qui plus est lorsque les ordres de grandeur de ces variables sont différents, il est préférable de faire une transformation des valeurs tracées pour la seconde courbe afin de se trouver dans la même zone de tracé. Une transformation affine adaptée de type `vmodifie <- a+b*v` (cf. exemple ci-dessous pour le pH) peut généralement être choisie de façon empirique. Il est ensuite souhaitable d'associer à cette nouvelle courbe un axe que l'on peut par exemple ajouter à droite du graphe. La fonction `axis` permet d'ajouter un axe. Cette fonction doit être paramétrée en premier par side, le côté sur lequel on veut ajouter l'axe (`side=1` pour en bas, `2` pour à gauche, `3` pour en haut et `4` pour à droite). Il faut ensuite affecter le paramètre `at` au vecteur contenant les ordonnées des marques du nouvel axe puis le paramètre `labels` au vecteur contenant les marques associées en se servant de la transformation affine choisie pour faire la correspondance entre les deux (cf. exemple suivant). Il est possible d'affecter ces marques soit à des valeurs numériques soit à des caractères ou mots à définir dans ce deuxième cas entre guillemets (ex. : `labels=c("A","B","C")`). Il peut être utile de savoir que l'on peut enlever l'axe représenté par défaut par la fonction `plot` en utilisant l'argument `xaxt="n"` (pour enlever l'axe des x) ou `yaxt="n"` (pour enlever l'axe des y).

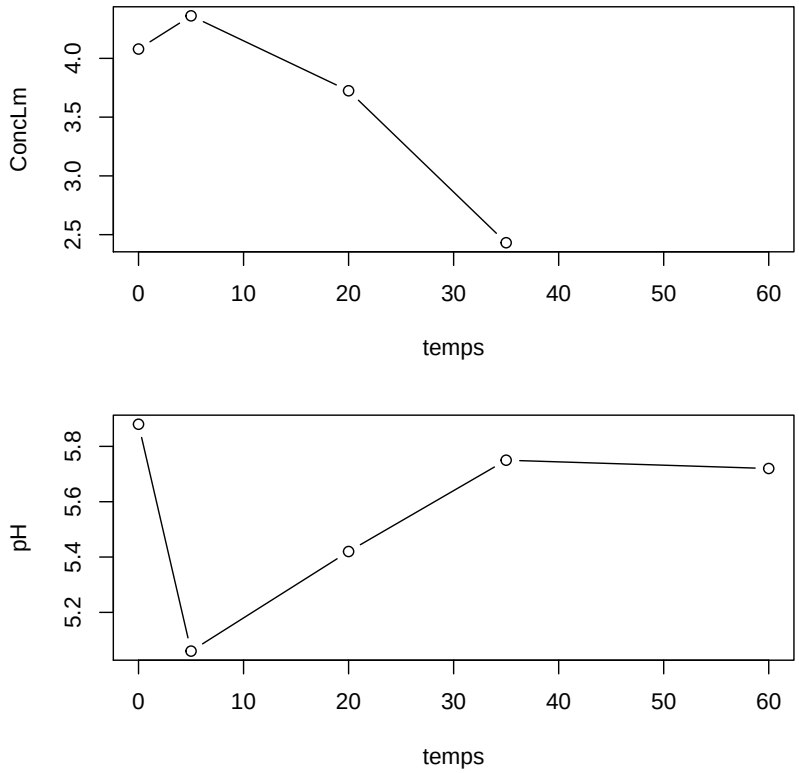
```
> pHmodifie<-pH*2-8
> plot(temps,ConcLm,type="l",ylim=c(2,4.5),xlab="temps en jours",ylab="",lty=1)
> points(temps,pHmodifie,type="l",lty=2)
> axis(side=4,at=c(2,3,4),labels=c(5,5.5,6))
> legend(40,4.2,c("C","pH"),lty=c(1,2))
> text(0,4.5,"C")
> text(60,4.5,"pH")
```



3.9 Pour changer les marges

Il est souvent utile de modifier les marges autours des graphes, notamment lorsqu'on partage la fenêtre graphique en plusieurs graphes et que les marges définies par défaut sont par défaut trop grandes. Ces marges doivent être paramétrées juste après la création de la fenêtre graphique, par la fonction `par` qui permet de modifier un grand nombre de paramètres de la fenêtre graphique (cf. `?par` pour plus de détails). L'argument correspondant aux marges est `mar`. Par défaut `mar` est fixé à `c(5.1,4.1,4.1,2.1)`, ces 4 valeurs représentant respectivement les marges **en bas, à gauche, en haut et à droite**. Il suffit donc d'augmenter ou de diminuer ces valeurs pour modifier les marges (cf. exemple).

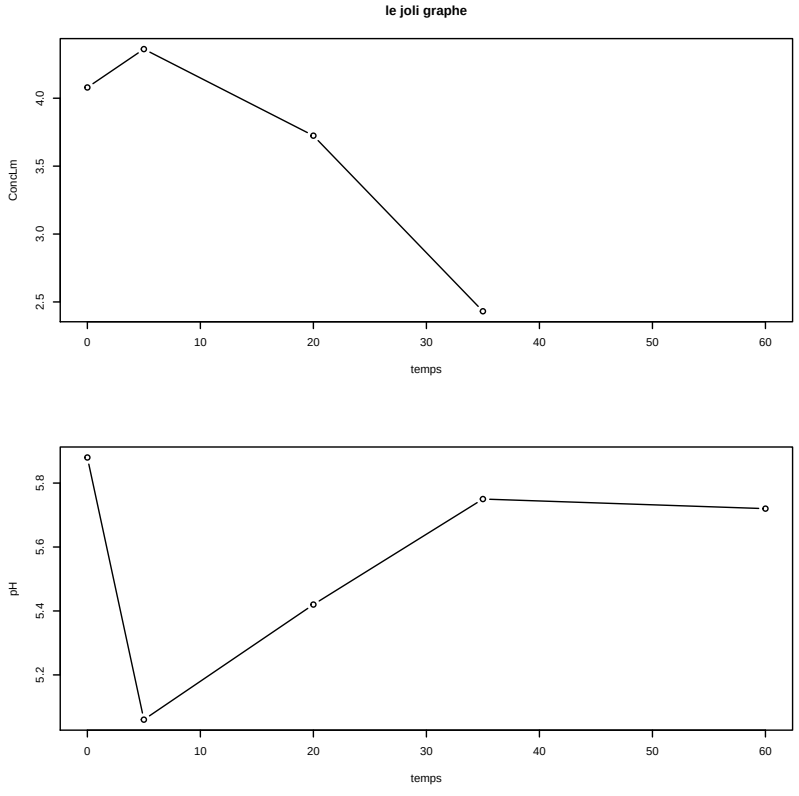
```
> par(mfrow=c(2,1))
> par(mar=c(5,4,1,1))
> plot(temps,ConcLm,type="b")
> plot(temps,pH,type="b")
```



3.10 Pour changer la taille des points tracés et du texte

On peut modifier globalement la taille de tous les éléments graphiques (points représentés, texte dans le titre et les étiquettes des axes, marques des axes et marges) en changeant la valeur de l'argument `cex` de la fonction `par` (sa valeur par défaut est 1) (cf. exemple ci-dessous).

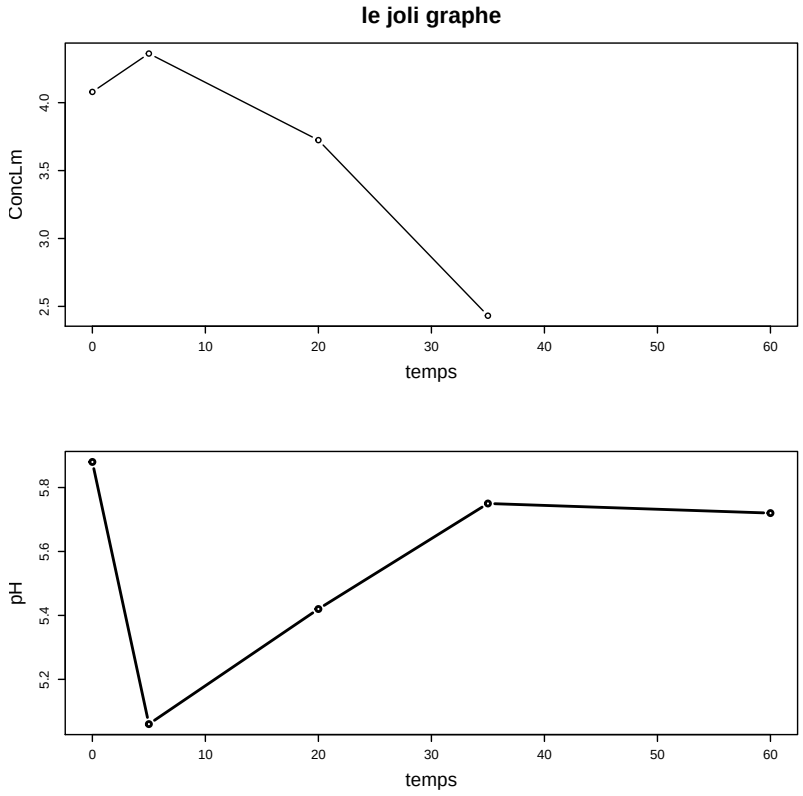
```
> par(mfrow=c(2,1))
> par(cex=0.5)
> plot(temps,ConcLm,type="b",main="le joli graphe")
> plot(temps,pH,type="b")
```



Mais il se peut que l'on veuille par exemple uniquement réduire la taille des points représentés mais pas celle des autres éléments graphiques ou pas avec le même rapport. Il faut alors modifier la taille courante avec `par(cex=correction)`

puis modifier les autres éléments par rapport à cette nouvelle taille courante : `cex.main` (la taille du titre), `cex.axis` (taille des marques des axes) et `cex.lab` (taille des étiquettes des axes) (cf. exemple suivant). Par ailleurs, il faut savoir que la largeur de la ligne lorsque l'on trace une courbe avec `plot` ou `points` peut être fixée en modifiant le paramètre `lwd` dans la fonction (valeur par défaut à 1) (cf. exemple suivant).

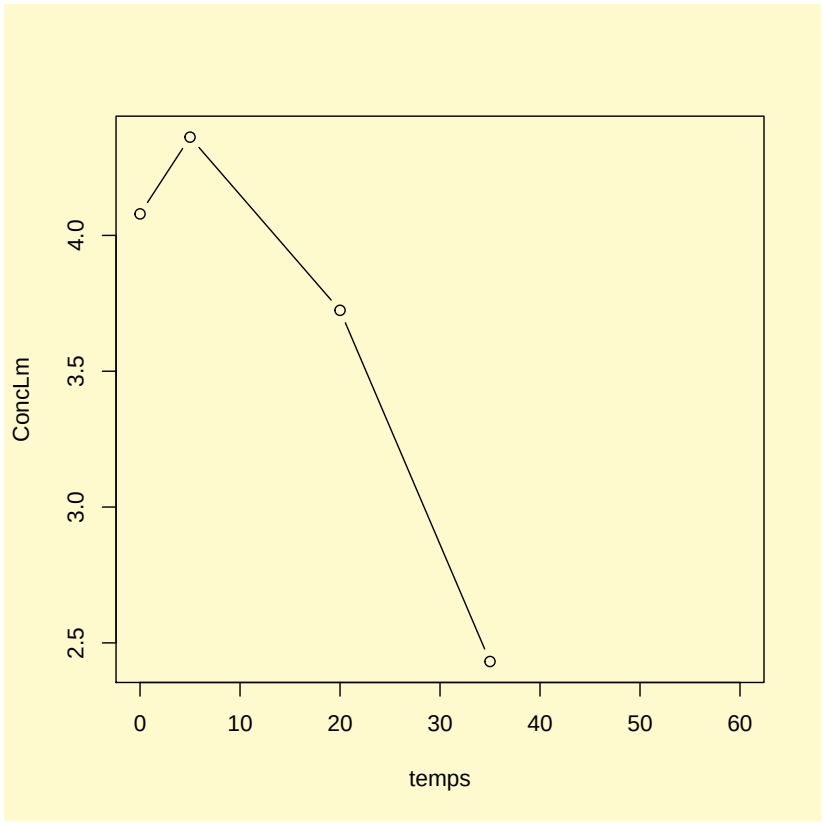
```
> par(mfrow=c(2,1))
> par(cex=0.5)
> # diminue toutes les tailles de moitié y compris les marges
> par(cex.main=2)
> # agrandit d'un facteur 2 le titre
> par(cex.axis=1.2)
> # diminue d'un facteur 0.8 les marques des axes
> par(cex.lab=1.7)
> # agrandit d'un facteur 1.7 les étiquettes des axes
> plot(temps,ConcLm,type="b",main="le joli graphe")
> plot(temps,pH,type="b",lwd=2)
> # tracé du pH avec une ligne 2 fois plus épaisse
```



3.11 Pour changer le fond de la fenêtre graphique

Il est parfois utile de changer la couleur de fond de la fenêtre graphique notamment lorsqu'on veut enregistrer le graphe dans un fichier JPEG (il faut le fixer à `"white"` sinon le fond par défaut peut apparaître gris dans certains cas). Il suffit pour cela de modifier l'argument `bg` avec la fonction `par` et de le fixer à la couleur souhaitée.

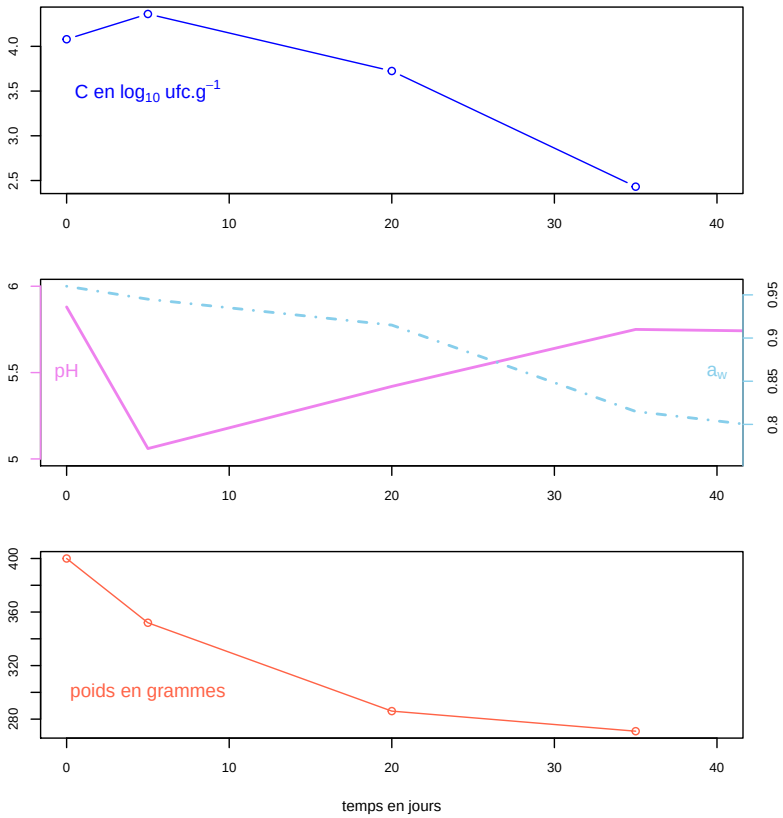
```
> par(bg="lemonchiffon1")
> plot(temps,ConcLm,type="b")
```



3.12 Comment intégrer tout cela pour créer un joli graphe

Lors de la réalisation d'un graphe, il faut donc tout d'abord ouvrir une fenêtre graphique (si vous ne voulez pas écraser la précédente), puis définir à l'aide de la fonction `par` les différents paramètres globaux de ce graphe puis utiliser les fonctions graphiques souhaitées en fixant à l'intérieur de ces fonctions les différents paramètres modifiables. N'oubliez pas que vous pouvez à tout moment consulter l'aide pour en savoir plus sur une fonction et sur ces paramètres (`?nomdelafonction`). Voici une représentation de l'exemple étudié utilisant une grande partie des outils de personnalisation des graphes présentés dans ce document.

```
> par(bg="white")
> par(mfrow=c(3,1))
> par(mar=c(4,3,0.5,3))
> par(cex=0.7)
> par(cex.axis=0.8)
> plot(temps,ConcLm,type="b",col="blue",xlab="",ylab="",xlim=c(0,40))
> text(5,3.5,expression(paste("C en ",log[10]," ufc.",g^-1)),col="blue",cex=1.2)
> plot(temps,pH,type="l",lwd=2,col="violet",xlab="",yaxt="n",xlim=c(0,40),ylim=c(5,6))
> awmodifie<-aw*5+1.2
> points(temps,awmodifie,type="l",lwd=2,col="skyblue",lty=4)
> axis(side=2,at=c(5,5.5,6),labels=c(5,5.5,6),col="violet")
> text(0,5.5,"pH",cex=1.2,col="violet")
> axis(side=4,at=c(4.95,5.2,5.45,5.7,5.95),labels=c(0.75,0.80,0.85,0.90,0.95),col="skyblue")
> text(40,5.5,expression(a[w]),cex=1.2,col="skyblue")
> plot(temps,poids,type="o",xlab="temps en jours",ylab="",xlim=c(0,40),col="tomato")
> text(5,300,"poids en grammes",col="tomato",cex=1.2)
```



Maintenant, pour vous entraîner, vous pouvez soit tenter de réaliser la figure illustrant la première page de ce document (si possible sans regarder le code correspondant qui figure à la dernière page de ce document), soit représenter vos propres données. N'hésitez pas à demander de l'aide et à explorer l'aide en ligne des fonctions utilisées, ce document étant loin d'être exhaustif sur les immenses possibilités offertes par chacune d'elles. Et ne perdez pas patience : il est souvent nécessaire de s'y prendre en plusieurs fois pour trouver les valeurs idéales des divers arguments des fonctions pour obtenir une figure claire et agréable à regarder.

Code de réalisation du graphe de la première page de ce document

```
d <- read.table("ENQ9697.txt",header=TRUE)
x11()
par(mfrow=c(2,2))
par(cex=0.8)
par(bg="lightyellow")
par(mar=c(4.2,4,2,1))

# Diagrammes boîte du POIDS en fonction du SEXE
##############################################
boxplot(POIDS~SEXE,col="tomato",xlab="sexe",ylab="POIDS en kg",main="poids")

# Diagramme de dispersion du POIDS et de l'âge en différenciant les 2 sexes
# par la couleur et le motif des points
#################################################################
dM<-subset(d,SEXE=="M") # création d'un jeu de données avec uniquement les garçons
dF<-subset(d,SEXE=="F") # création d'un jeu de données avec uniquement les filles
plot(dM$TAILLE,dM$POIDS,pch=3,col="blue",ylab="POIDS en kg",xlab="TAILLE en cm",
     ylim=c(45,95),xlim=c(150,190),main="corrélation poids - taille")
points(dF$TAILLE,dF$POIDS,pch=1,col="violet")
legend(152,95,c("garçons","filles"),pch=c(3,1),col=c("blue","violet"))

# Représentations en barres de la distribution de l'âge du choix professionnel
# en fonction du sexe
#############################################################################
DECISION2 <- factor(DECISION, levels = c("E", "A", "T"))
levels(DECISION2) <- c("Enfant", "Adolescent", "Tardivement")
t<-table(DECISION2,SEXE)
t2<-prop.table(t,margin=2)
barplot(t2,col=c("red","orange","yellow"),legend.text= TRUE,
args.legend=list(y=0.3,cex=0.6),main="précocité du choix")

# Diagramme en bâtons de la filière envisagée
#############################################
t3<-table(FILIERE)
barplot(prop.table(t3),main="filière envisagée",ylab="fréquence",col="darkblue")
```