# Travaux dirigés pour l'introduction au logiciel R

Christophe Ambroise

05/09/2018

# Débuts avec Rstudio et Rmarkdown

# Ecrire des lignes de code et les sauvegarder

1. Ecriver des lignes de codes qui affecte des valeurs réelles aux variables $a$, $b$, $c$ et $x$.
2. Calculer et afficher la valeur de

$$y = ax^2 + bx + c$$

   pour $a = 1$, $b = 2$, et $c = -1$
3. Sauver vos lignes de codes dans un fichier R (`td1.R`)
4. Changer les valeur de a, b, c dans le fichier et tester la commande 'source('td1.R')

# Solution

```
a<-1
b<-2
c<- -1
x<-4
y<- a*x^2 + b^x + c
y
```

```
## [1] 31
```

# Chercher de l'aide et écrire une fonction

1. Chercher comment écrire une fonction (`function en anglais`)
2. Ecriver une fonction `trinome` qui calcule $f(x) = a * x^2 + b^x + c$
3. Sauvegarder votre fonction de `td1.R` avec des commentaires
4. Executer la fonction pour différentes valeur de $x$

# Solution

```
trinome<-function(x){
  a<-1
  b<-2
  c<- -1
  y<- a*x^2 + b^x + c   # Attention seule cette dernière va
}

print(trinome(4))
```

```
## [1] 31
```

# Calculer le déterminant du trimome

1. Ecrire une fonction qui calcule le déterminant du trinome
2. Ecrire une fonction qui donne les racines du trinome et utilise la fonction précédente

# Solution

```
determinant <- function(a,b,c){
  Delta<- b^2 -4*a*c
  # si vous voulez résoudre le trinome dans les complexes
  #  il vous faudrait mettre as.complex(Delta)  en dernièr
}
determinant(1,2,-1)
```

```
roots<-function(a,b,c){
  x1<- (-b+sqrt(determinant(a,b,c)))/(2*a)
  x2<- (-b-sqrt(determinant(a,b,c)))/(2*a)
  print(x1)
  print(x2)
}

roots(1,2,-1)
```

```
## [1] 0.4142136
## [1] -2.414214
```

# De l'aide encore

1. Trouver la fonction `R` qui fait le travail de votre fonction `roots`

# Solution

```
??root
polyroot(c(-1,2,1))
```

```
## [1]  0.4142136+0i -2.4142136-0i
```

# Illustration

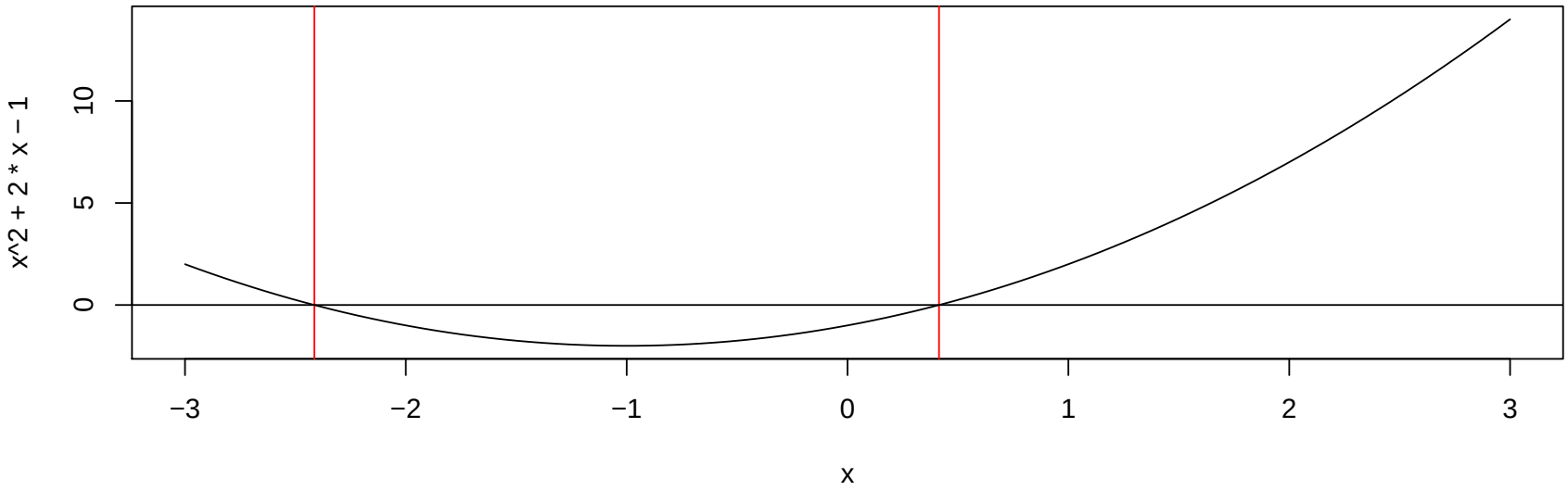
1. Chercher comment tracer une courbe en R base (`curve` en anglais)
2. Tracer votre trinome
3. Indiquer les zéros par des lignes verticales rouges (fonction `abline`)

# Solution

```
curve(x^2+2*x-1,-3,3)
abline(v = as.numeric(polyroot(c(-1,2,1))),col="red")
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf =
## parties imaginaires sont perdues lors de la conversion a
```

```
abline(h=0)
```



# Ecrire un fichier Rmarkdown reproducible

1. Transformer votre fichier R en un fichier Rmarkdown
2. Produiser un fichier html
3. Produiser un fichier pdf

# Introduction aux objets R

# Génération de vecteurs I

Où l'on se familiarise avec la création de vecteurs (commandes `c()`, `seq()`, `rep()`, `paste()` et leurs options).

1. Créer un vecteur contenant la suite des entiers de 1 à 12 de deux manières différentes.
2. Créer le vecteur `c(0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0)` de trois manières différentes.
3. Créer un vecteur contenant tous les multiples de 2 compris entre 1 et 50.
4. Créer un vecteur contenant tous les nombres de 1 à 100 qui ne sont pas des multiples de 5.
5. Créer un vecteur contenant 3 fois chacun des 10 chiffres.
6. Créer un vecteur contenant une fois la lettre A, deux fois la lettre B, etc., 26 fois la lettre Z. Quelle est la longueur de cette suite ? (Utiliser la chaîne `LETTERS` prédéfinie).

# Génération de vecteurs II

7. Créer le vecteur `c("individu 1", "individu 2", ...,"individu 100")`.

# Génération de vecteurs (Solution) I

1.

```
1:12
seq(from=1,to=12,by=1)
```

2.

```
seq(from=0.5,to=5,len=10)
seq(from=0.5,to=5,by=0.5)
1:10/2
```

3. L'opérateur %% correspond au reste de la division entière.

```
v <- 1:50
v[v %% 2 == 0]
```

# Génération de vecteurs (Solution) II

4.

```
v <- 1:100
v[v %% 5 != 0]
```

5.

```
rep(c(1:10),each=3)
```

6.

```
rep(LETTERS,c(1:length(LETTERS)))
```

7.

```
paste(rep("individu",100),1:100)
```

# Manipulation de séquences I

Où l'on se familiarise avec la manipulation de vecteurs

1. Quels sont les entiers divisibles par 3 parmi les 100 premiers et combien y-en a-t-il ? Que vaut leur somme ? Leur produit ?
2. Générer une séquence d'ADN de $n$ bases. Compter le nombre d'occurrences de chaque lettre (d'abord sans puis avec la fonction `table`). Renvoyer les indices de la séquence où l'on trouve la lettre `"t"`.
3. Créer un vecteur contenant les 100 premiers entiers échantillonnés aléatoirement. À partir de ce vecteur, créer les vecteurs `x` et `y` des 100 premiers entiers ordonnés dans l'ordre croissant et décroissant. Concatenez `x` et `y`, enlever le seul nombre apparaissant deux fois de suite en le repérant à l'aide de la commande `diff`.

# Manipulation de séquences II

4. On rappelle que
$$e^x = \sum_{k \geq 0} \frac{x^k}{k!}.$$
Créer dans un vecteur `exp2` les 20 premiers termes de cette suite. Supprimer toutes les valeurs inférieures à $10^{-8}$. En déduire une approximation de $e^2$ et comparer avec la valeur `exp(2)`.
5. Créer un vecteur `couleurs` contenant 10 couleurs de votre choix sous forme de chaînes de caractères. Créer un échantillon aléatoire de taille 4 parmi ces couleurs ainsi qu'un vecteur `primaires` contenant les 3 couleurs primaires. Tester combien et quelles sont les couleurs primaires présentes dans votre échantillon.

# Manipulation de séquences (Solution) I

1.

```
integers <- 1:100
div.by.3 <- integers[integers %% 3 == 0]
cat("Nombre:",length(div.by.3), "somme:",sum(div.by.3),
    "produit:",prod(div.by.3))
```

```
## Nombre: 33 somme: 1683 produit: 4.827109e+52
```

2. On peut utiliser l'opérateur `sum` sur un vecteur de booléens:

# Manipulation de séquences (Solution) II

```r
adn <- sample(c("a","c","g","t"),50,rep=TRUE)
nbA <- sum(adn == "a")
nbC <- sum(adn == "c")
nbG <- sum(adn == "g")
nbT <- sum(adn == "t")
cat(nbA,"A,", nbC,"C,", nbG,"G,", nbT, "T.")
```

```
## 12 A, 14 C, 10 G, 14 T.
```

Cependant, la fonction `table` fait directement le travail :

```r
table(adn)
```

```
## adn
##  a  c  g  t
## 12 14 10 14
```

# Manipulation de séquences (Solution) III

```r
which(adn == "t")
```

```
##  [1]  2  6  7  8  9 15 17 19 22 29 33 34 38 42
```

3.

```r
v <- sample(1:100)
x <- sort(v)
y <- rev(x)
v <- c(x,y)
v <- v[-which(diff(x) == 0)]
```

4.

# Manipulation de séquences (Solution) IV

```
exp2 <- 2^(0:20) / factorial(0:20)
exp2 <- exp2[exp2 >= 1e-8]
sum(exp2)
```

```
## [1] 7.389056
```

```
exp(2)
```

```
## [1] 7.389056
```

5.

# Manipulation de séquences (Solution) V

```
couleurs  <- c("rouge","bleu","vert","marron","rose","jaune
echantillon <- sample(couleurs,4)
primaires <- c("rouge","jaune","bleu")
sum(primaires %in% echantillon)
```

```
## [1] 2
```

```
intersect(primaires,echantillon)
```

```
## [1] "rouge" "bleu"
```

# Jeux de hasard I

1. On veut mimer un jeu de pile ou face. On notera 1 pour pile et 0 pour face. Simulez 1000 lancers de pièces. On gagne 1 euro si c'est pile et on perd 1 euro si c'est face. Combien d'argent avez-vous gagné à l'issue des 1000 lancers ?
2. Un ami vous propose le jeu suivant. On lance un dé. Si le résultat est 5 ou 6, on gagne 3 euros, si le résultat est 4 on gagne 1 euro et si c'est 3 ou moins on perd 2.5 euros. Avant d'accepter la partie, vous essayez de simuler ce jeu, pour voir si vous avez des chances de vous enrichir. Conclusion ?

# Jeux de hasard (Solution) I

1.

```
tirages <- sample(c("pile","face"),1000,rep=TRUE)
gains <- sum(tirages == "pile") - sum(tirages == "face")
gains
```

```
## [1] 24
```

2.

```
tirages  <- sample(c(1:6),100000,rep=TRUE)
esp.emp  <- 3 * sum(tirages >= 5) + sum(tirages == 4) - 2.5
  sum(tirages <= 3)
esp.theo <- (2/6 * 3 + 1/6 * 1 - 3/6 * 2.5) * 100000
cat("Espérance théorique:",esp.theo, "estimée:",esp.emp)
```

# Jeux de hasard (Solution) II

```
## Espérance théorique: -8333.333 estimée: -8414
```

# Tableaux de données I

On suppose que la taille et le poids des individus en France se répartissent selon des lois normales de paramètres suivants:

- la taille des femmes est en moyenne de 165 cm, avec un écart-type de 6 cm.
- la taille des hommes est en moyenne de 175 cm, avec un écart-type de 7 cm.
- le poids des femmes est en moyenne de 60 kg, avec un écart-type de 2 kg.
- le poids des hommes est en moyenne de 75 kg, avec un écart-type de 4 kg.

# Tableaux de données II

Créer les vecteurs `taille.homme`, `taille.femme`, `poids.homme`, `poids.femme` contenant la taille et le poids de $n = 257$ hommes et $m = 312$ femmes générés selon les lois ci-dessus (on supposera que poids et taille sont deux variables indépendantes ce qui est bien entendu faux!).

Créer un tableau `donnees` à $n + m$ lignes et 3 colonnes telles que

- la première colonne contienne la variable taille,
- la deuxième colonne contienne la variable poids,
- la troisième colonne soit un facteur indiquant le sexe de l'individu.

Placer l'objet `donnees` dans l'itinéraire de recherche à l'aide de la commande `attach`. Puis, à l'aide de la commande `by`,

- déterminer simultanément le plus petit poids chez les femmes et le plus petit poids chez les hommes, ainsi que le numéro des individus correspondant,

# Tableaux de données III

- de même pour la taille,
- faites un résumé statistique de chacun des deux groupes (commande `summary`).

Y a-t-il des hommes de plus d'un mètre quatre vingt dix et de moins de soixante-quinze kilos ? Si oui, combien ? Y a-t-il des femmes de moins d'un mètre soixante et de plus de soixante kilos ? Combien ?

# Tableaux de données (Solution) I

Commençons par créer les vecteurs composant l'échantillon. Deux décimales suffisent:

```
n <- 257
m <- 312
taille.f <- round(rnorm(m,165,6),2)
taille.h <- round(rnorm(n,175,7),2)
poids.f  <- round(rnorm(m,60,2),2)
poids.h  <- round(rnorm(n,75,4),2)
```

Passons à la création du tableau de données:

```
taille <- c(taille.f,taille.h)
poids  <- c(poids.f,poids.h)
sexe   <- rep(c("femme","homme"),c(m,n))
donnees <- data.frame(cbind(taille,poids,sexe))
```

# Tableaux de données (Solution) II

La commande `head` est pratique pour vérifier le format des données, en affichant les premières éléments d'un objet selon son type:

```
head(donnees)
```

```
##   taille poids  sexe
## 1    163 62.66 femme
## 2 162.82 59.05 femme
## 3 171.43 58.82 femme
## 4 160.31 62.16 femme
## 5 166.01  61.3 femme
## 6 157.49 61.14 femme
```

Plaçons les données dans l'itinéraire de recherche pour les manipuler plus aisément:

```
attach(donnees)
```

# Tableaux de données (Solution) III

```
## The following objects are masked _by_ .GlobalEnv:
##
##     poids, sexe, taille
```

Les commandes by ou tapply appliquent une fonction selon un facteur:

```
tapply(taille,sexe,min)
```

```
##  femme  homme
## 147.95 151.81
```

```
tapply(taille,sexe,which.min)
```

```
## femme homme
##   159   182
```

# Tableaux de données (Solution) IV

```
tapply(poids,sexe,min)
```

```
## femme homme
## 54.38 62.95
```

```
tapply(poids,sexe,which.min)
```

```
## femme homme
##   262   110
```

On peut donc obtenir un résumé statistiques selon chaque groupe en combinant avec `summary`:

```
summary(donnees)
```

# Tableaux de données (Solution) V

```
##      taille          poids          sexe
##  161.06 :  3   60.43  :  4   femme:312
##  170.2  :  3   58.42  :  3   homme:257
##  176.45 :  3   58.89  :  3
##  177.4  :  3   60.04  :  3
##  158.77 :  2   60.08  :  3
##  159.4  :  2   60.48  :  3
##  (Other):553   (Other):550
```

```
by(taille,sexe,summary)
```

```
## sexe: femme
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   147.9   161.5   165.6   165.5   169.1   184.2
## ------------------------------------------------
## sexe: homme
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

# Tableaux de données (Solution) VI

```
##   151.8   170.8   175.0   175.1   179.6   196.8
```

```
by(poids,sexe,summary)
```

```
## sexe: femme
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   54.38   58.69   60.04   60.04   61.27   64.74
## ------------------------------------------------
## sexe: homme
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   62.95   72.42   74.81   74.86   77.91   84.73
```

# Population de bactéries I

On souhaite modéliser la croissance d'une population bactérienne mise en culture dans une boîte de Petri. À cet effet, on distingue deux types de bactérie:

1. des bactéries jeunes et "immatures", notées $a$, qui ne se divisent pas ;
2. des bactéries "matures, notées $b$, susceptibles de se diviser par mitose.

On suppose que la reproduction a lieu à intervalles de temps discrets ; les bactéries $b$ se divisent d'un instant à l'autre en une bactérie $a$ et une bactérie $b$ ; enfin, toute bactérie $a$ devient mature d'un pas de temps à l'autre.

**Questions**

# Population de bactéries II

1. On note $n_a(t)$ et $n_b(t)$ le nombre de bactéries de chaque type à l'instant $t$. Écrire le système de deux équations décrivant l'évolution de $n_a(t+1)$ et $n_b(t+1)$ en fonction de $n_a(t)$ et $n_b(t)$.
2. Écrire une fonction `PopBacteries(n0,T)` qui renvoie trois vecteurs de taille $T+1$ contenant l'évolution des deux catégories de bactérie de l'instant initial au temps $T$ ainsi que l'évolution de la population totale. Le paramètre $n_0$ est le nombre $n_a(0)$, et l'on suppose que $n_b(0) = 0$.
3. Pour $T = 20$ et $n_0 = 1$, générer la population bactérienne correspondante et calculer le taux d'accroissement de la population totale. Représenter graphiquement ces résultats (fonction `plot`).
4. On souhaite maintenant introduire de l'aléa dans la dynamique bactérienne. À cet effet, on suppose qu'une bactérie de type $b$ a une probabilité $p$ d'accomplir une mitose en $a+b$. Modifier la fonction `PopBacteries(n0,T,p)` en ajoutant le paramètre $p$.

# Population de bactéries III

5. Étudier l'évolution de la population et son taux de croissance totale pour diverses valeurs de $p$.

# Population de bactéries (Solution) I

Commençons par la fonction générant la population (y compris avec le facteur aléatoire)

```
PopBacteries <- function(n0,T,p) {

  Na <- rep(0,T)
  Nb <- rep(0,T)
  Na[1] <- n0

  for (t in 1:T) {
    Na[t+1] <- sum(runif(Nb[t]) <=p)
    Nb[t+1] <- Na[t] + Nb[t]
  }

  return(list(Na=Na,Nb=Nb,N=Na+Nb))
}
```

# Population de bactéries (Solution) II

Voyons quelques exemples

# Population de bactéries (Solution) III

```
T <- 20
Pop1 <- PopBacteries(1,T,1)
Pop2 <- PopBacteries(2,T,0.35)
Pop3 <- PopBacteries(5,T,0.15)

par(mfrow=c(1,3))
plot(0:T,Pop1$N, type="l", col="black",
     xlab="temps", ylab="# bactéries")
lines(0:T,Pop1$Na, col="blue", lty=2)
lines(0:T,Pop1$Nb, col="red", lty=3)
legend("topleft",c("total","Na","Nb"),
       col=c("black","blue","red"), lty=c(1,2,3))

plot(0:T,Pop2$N, type="l", col="black",
     xlab="temps", ylab="# bactéries")
lines(0:T,Pop2$Na, col="blue", lty=2)
lines(0:T,Pop2$Nb, col="red", lty=3)
legend("topleft",c("total","Na","Nb"),
```