TP1 Correction : Introduction à R

Sonia Tieo

15/10/2020

1.Entrer, sortir et trouver de l'aide avec R

R ouvre une session à chaque entrée. Par defaut (sous Windows) la session est toujours dans le même répertoire. Pour changer de répertoire, utiliser le menu.

Pour avoir de l'aide sur la fonction mean, il suffit de taper:

```
help(mean)
```

Une aide html est disponible avec :

```
help.start()
```

```
## starting httpd help server ... done
```

```
## If the browser launched by 'xdg-open' is already running, it is *not*
##     restarted, and you must switch to its window.
## Otherwise, be patient ...
```

2.R comme machine à calculer

2-1. Calculs simples

Exécutez et commentez :

```
2+2
```

```
## [1] 4
```

```
#blabla (blabla sans # ne marche pas)

2+2 # ceci est une addition
```

```
## [1] 4
```

```
pi
```

```
## [1] 3.141593
```

```
exp(2)
```

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

```
log(10)
```

```
## [1] 2.302585
```

```
sin(5*pi)
```

```
## [1] 6.123234e-16
```

```
(1+3/5)*5
```

```
## [1] 8
```

2-2.Calculs sur plusieurs valeurs

Si nous voulons faire une moyenne de notes, il faut pouvoir manipuler plusieurs valeurs ensemble (donc un vecteur). Effectuons la moyenne (mean) de 4, 10 et 16 :

```
mean(c(4,10,16))
```

```
## [1] 10
```

En utilisant la ligne suivante:

```
mean(c(4,10,16))
```

```
## [1] 10
```

En utilisant la ligne suivante:

```
c(4,10,16)
```

```
## [1]  4 10 16
```

Trouvez à quoi sert la fonction c().

Exercice d'application:
1. Calculer la moyenne de 2,3,9,5,4
2. Calculer la somme de 4,6,20
3. Calculer la médiane de 4,6,20

```
mean(c(2,3,9,5,4))
```

```
## [1] 4.6
```

```
sum(c(4,6,20))
```

```
## [1] 30
```

```
median(c(4,6,20))
```

```
## [1] 6
```

2-3.Mettre en mémoire plusieurs valeurs

Nous souhaitons stocker un vecteur pour le réutiliser. Nous devons donc affecter des valeurs à un nom. Executer dans cet ordre et commenter ces ordres :

```
x <- pi
print(x)
```

```
## [1] 3.141593
```

```
x
```

```
## [1] 3.141593
```

```
objects()
```

```
## [1] "x"
```

```
y=pi
objects()
```

```
## [1] "x" "y"
```

```
y
```

```
## [1] 3.141593
```

```
x <- c(4,10,16)
print(x)
```

```
## [1]  4 10 16
```

```
x
```

```
## [1]  4 10 16
```

Que remarquez-vous ?

Exercices d'application :
x = c(32,48,10,2)
1. Calculer le max (max) de x.
2. Calculer le min (min) de x.
3. Calculer la moyenne (mean) de x.
4. Calculer la longueur (length) de x.
5. Calculer le résumé numérique (summary) de x.

```
x = c(32,48,10,2)
max(x)
```

```
## [1] 48
```

```
min(x)
```

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

```
mean(x)
```

```
## [1] 23
```

```
length(x)
```

```
## [1] 4
```

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       2       8      21      23      36      48
```

3.R manipule des vecteurs

3-1.Calcul vectoriel

Additionnons 2 vecteurs, commentez:

```
y=c(-1,5,0,2)
x
```

```
## [1] 32 48 10  2
```

```
y
```

```
## [1] -1  5  0  2
```

```
x+y
```

```
## [1] 31 53 10  4
```

```
-y
```

```
## [1]  1 -5  0 -2
```

Commenter

```
x+2
```

```
## [1] 34 50 12  4
```

```
abs(y)
```

```
## [1] 1 5 0 2
```

```
x*y
```

```
## [1] -32 240   0   4
```

```
x/y
```

```
## [1] -32.0   9.6   Inf   1.0
```

Commenter

```
1:3
```

```
## [1] 1 2 3
```

```
-1:5
```

```
## [1] -1  0  1  2  3  4  5
```

```
-(1:5)
```

```
## [1] -1 -2 -3 -4 -5
```

3-2.Vecteur de logiques

Les logiques sont soit TRUE soit FALSE.

```
w <- c(TRUE,FALSE,FALSE)
sum(w)
```

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

```
all(w)
```

```
## [1] FALSE
```

```
!w
```

```
## [1] FALSE  TRUE  TRUE
```

```
(TRUE)&(FALSE)
```

```
## [1] FALSE
```

```
(TRUE)|(FALSE)
```

```
## [1] TRUE
```

```
(TRUE)|(TRUE)
```

```
## [1] TRUE
```

3-3. Valeurs spéciales et calculs

- NA est la valeur manquante
- NaN est la valeur « Not a Number »
- Inf est l'infini

```
log(0)
```

```
## [1] -Inf
```

```
log(Inf)
```

```
## [1] Inf
```

```
1/0
```

```
## [1] Inf
```

```
0/0
```

```
## [1] NaN
```

```
max(c( 0/0,1,10))
```

```
## [1] NaN
```

```
max(c(NA,1,10))
```

```
## [1] NA
```

```
max(c(-Inf,1,10))
```

```
## [1] 10
```

```
is.finite(c(-Inf,1,10))
```

```
## [1] FALSE  TRUE  TRUE
```

```
is.na(c(NA,1,10))
```

```
## [1]  TRUE FALSE FALSE
```

```
is.nan(c(NaN,1,10))
```

```
## [1]  TRUE FALSE FALSE
```

3-4. Créer des vecteurs

1. Créer le vecteur d'entiers de 5 à 23.
2. Créer le vecteur de 6 à 24 allant de 2 en 2.
3. Créer le vecteur de 100 valeurs régulièrement espacées entre 0 et 1.
4. Créer le vecteur suivant
   [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

5. Créer le vecteur suivant
   [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5

6. Créer le vecteur suivant
   1 1 2 2 2 3 3 3 3

```
#1
5:23
```

```
##  [1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
```

```
#2
seq(6,24,by = 2)
```

```
##  [1]  6  8 10 12 14 16 18 20 22 24
```

```
#3
seq(0,100,length.out = 100)
```

```
##   [1]   0.000000   1.010101   2.020202   3.030303   4.040404   5.050505
##   [7]   6.060606   7.070707   8.080808   9.090909  10.101010  11.111111
##  [13]  12.121212  13.131313  14.141414  15.151515  16.161616  17.171717
##  [19]  18.181818  19.191919  20.202020  21.212121  22.222222  23.232323
##  [25]  24.242424  25.252525  26.262626  27.272727  28.282828  29.292929
##  [31]  30.303030  31.313131  32.323232  33.333333  34.343434  35.353535
##  [37]  36.363636  37.373737  38.383838  39.393939  40.404040  41.414141
##  [43]  42.424242  43.434343  44.444444  45.454545  46.464646  47.474747
##  [49]  48.484848  49.494949  50.505051  51.515152  52.525253  53.535354
##  [55]  54.545455  55.555556  56.565657  57.575758  58.585859  59.595960
##  [61]  60.606061  61.616162  62.626263  63.636364  64.646465  65.656566
##  [67]  66.666667  67.676768  68.686869  69.696970  70.707071  71.717172
##  [73]  72.727273  73.737374  74.747475  75.757576  76.767677  77.777778
##  [79]  78.787879  79.797980  80.808081  81.818182  82.828283  83.838384
##  [85]  84.848485  85.858586  86.868687  87.878788  88.888889  89.898990
##  [91]  90.909091  91.919192  92.929293  93.939394  94.949495  95.959596
##  [97]  96.969697  97.979798  98.989899 100.000000
```

```
#4
rep(1:5,times=3)
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
#5
rep(x=1:5,each=3)
```

```
##  [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
```

```
#
c(rep(1,each=2),rep(2,each=3),rep(3,each=4))
```

```
## [1] 1 1 2 2 2 3 3 3 3
```

3-5. Selectionner des vecteurs

Par leur numéro:

```
x[1]
```

```
## [1] 32
```

```
x[2]
```

```
## [1] 48
```

```
x[c(1,2,3)]
```

```
## [1] 32 48 10
```

```
x[1:3]
```

```
## [1] 32 48 10
```

```
x[c(2,2,1,3)]
```

```
## [1] 48 48 32 10
```

```
x[c(1:3,2,1)]
```

```
## [1] 32 48 10 48 32
```

```
x[-1]
```

```
## [1] 48 10  2
```

```
x[-c(1,2)]
```

```
## [1] 10  2
```

```
x[-(1:2)]
```

```
## [1] 10  2
```

Par des logiques:

```
objects()
```

```
## [1] "w" "x" "y"
```

```
vec1<-c(3,NA,4)
objects()
```

```
## [1] "vec1" "w"    "x"    "y"
```

```
vec2<-c(FALSE,TRUE,FALSE)
objects(pattern="vec*")
```

```
## [1] "vec1" "vec2"
```

```
vec2
```

```
## [1] FALSE  TRUE FALSE
```

```
vec1
```

```
## [1]  3 NA  4
```

```
vec1[vec2]
```

```
## [1] NA
```

```
is.na(vec1)
```

```
## [1] FALSE  TRUE FALSE
```

Tapez l'ordre suivant sans essayer de l'interpréter (dans un premier temps) :

```
vec1<-runif(20)
vec1
```

```
##  [1] 0.05640999 0.21390517 0.56110749 0.26592275 0.17884796 0.95882679
##  [7] 0.39902923 0.87843310 0.59140888 0.58817070 0.65532806 0.41810522
## [13] 0.01578423 0.25796975 0.58112815 0.63762519 0.47568257 0.94288264
## [19] 0.82664225 0.97496487
```

```
vec1[vec1>0.5]<-NA
vec1
```

```
##  [1] 0.05640999 0.21390517         NA 0.26592275 0.17884796         NA
##  [7] 0.39902923         NA         NA         NA         NA 0.41810522
## [13] 0.01578423 0.25796975         NA         NA 0.47568257         NA
## [19]         NA         NA
```

En utilisant le groupe d'ordres précédents, remplacez les valeurs manquantes de vec1 par 0 ; retournez sur la ligne précédente et l'interpréter.

3-6. Chaînes de caractères (pour aller plus loin)

Exécutez et commentez

```
z=c("aze","fds")
z[1]
```

```
## [1] "aze"
```

```
paste("m",1:3)
```

```
## [1] "m 1" "m 2" "m 3"
```

```
paste("m",1:3,sep="")
```

```
## [1] "m1" "m2" "m3"
```

```
c(paste("m",1:3,sep=""),paste("p",1:4,sep="."),z)
```

```
## [1] "m1"  "m2"  "m3"  "p.1" "p.2" "p.3" "p.4" "aze" "fds"
```

4.R est un (super) tableur

1. Importez les données du tableau contenu dans le fichier tab1.ods dans l'objet R que vous appelerez don1. Le resultat de l'affichage à l'écran doit être:

	V1	V2
1	1	2
2	0	2
3	3	1

2. Importez les données du tableau contenu dans le fichier tab2.xls dans l'objet R que vous appelerez don2. Le resultat de l'affichage à l'écran doit être:

	variable1	variable2
1	-1.0	0
2	2.0	-2

	variable1	variable2
3	3.1	4

3. Importer les données du tableau contenu dans le fichier tab3.xls dans l'objet R que vous appelerez don3. Le resultat de l'affichage à l'écran doit être :

	sexe	taille
aang	masculin	172,9
katara	feminin	175,2
sokka	masculin	180,5

```
#ATTENTION TRANSFORMER LE ODS EN CSV
library(readxl)
don1 <- read.csv("~/Bureau/COURS_R_FINANCES/TP1/tab1.csv", header=FALSE)
don2 <- read_excel("~/Bureau/COURS_R_FINANCES/TP1/tab2.xls")
don3 <- read.csv("~/Bureau/COURS_R_FINANCES/TP1/tab3.csv")
```

4.1 Les noms des variables (colonnes) et des individus (lignes)

Exécuter et commenter :

```
rownames(don3)
```

```
## [1] "1" "2" "3"
```

```
names(don3)
```

```
## [1] "nomligne" "sexe"     "taille"
```

```
colnames(don3)
```

```
## [1] "nomligne" "sexe"     "taille"
```

```
colnames(don1)[2]
```

```
## [1] "V2"
```

```
colnames(don1)[2] <- "var2"
colnames(don1)
```

```
## [1] "V1"   "var2"
```

```
colnames(don1) <- colnames(don2)
colnames(don1) <- c("VAR1" , "VAR2")
```

4.2 Sélection dans des tableaux

Exécuter et commenter : TABLEAU[LIGNE, COLONNE]

```
don1[1,]
```

```
##   VAR1 VAR2
## 1    1    2
```

```
don3[,"sexe"]
```

```
## [1] masculin feminin  masculin
## Levels: feminin masculin
```

```
don3$sexe
```

```
## [1] masculin feminin  masculin
## Levels: feminin masculin
```

```
don3[,2]
```

```
## [1] masculin feminin  masculin
## Levels: feminin masculin
```

```
don3[,c(FALSE,TRUE)]
```

```
## [1] masculin feminin  masculin
## Levels: feminin masculin
```

```
don3[,c("taille","sexe")]
```

```
##   taille     sexe
## 1  172,9 masculin
## 2  175,2  feminin
## 3  180,5 masculin
```

```
don1[1,2]
```

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

```
don1[,1:2]
```

```
##   VAR1 VAR2
## 1    1    2
## 2    0    2
## 3    3    1
```

```
don1[-1,]
```

```
##   VAR1 VAR2
## 2    0    2
## 3    3    1
```

```
don1[c(2,3),c(2,1)]
```

```
##   VAR2 VAR1
## 2    2    0
## 3    1    3
```

```
don1[c(TRUE,FALSE,TRUE),]
```

```
##   VAR1 VAR2
## 1    1    2
## 3    3    1
```

```
don1[don1[,1]>0,]
```

```
##   VAR1 VAR2
## 1    1    2
## 3    3    1
```

```
don1[-(1),]
```

```
##   VAR1 VAR2
```

```
## 2    0    2
## 3    3    1
```

```
don1[,c(2,1)]
```

```
##   VAR2 VAR1
## 1    2    1
## 2    2    0
## 3    1    3
```

4.3 Sélection dans des tableaux

Exécuter et commenter :

```
don1[,1]+don1[,2]
```

```
## [1] 3 2 4
```

```
exp(don1[,1])
```

```
## [1]  2.718282  1.000000 20.085537
```

```
don3[1,]+don3[2,]
```

```
## Warning in Ops.factor(left, right): '+' not meaningful for factors

## Warning in Ops.factor(left, right): '+' not meaningful for factors

## Warning in Ops.factor(left, right): '+' not meaningful for factors
```

```
##   nomligne sexe taille
## 1       NA   NA     NA
```

4.4 Fusion des tableaux

1. Importez les données du tableau contenu dans le fichier tab3.xls dans l'objet R que vous appelerez avatar1. Le resultat de l'affichage à l'écran doit être:

nomligne	sexe	taille
aang	masculin	172,9
katarA	feminin	175,2
sokka	masculin	180,5

2. Importez les données du tableau contenu dans le fichier tab4.xls dans l'objet R que vous appelerez avatar2. Le resultat de l'affichage à l'écran doit être:

nomligne	sexe	taille
zuko	masculin	172
toph	feminin	150,2
iroh	masculin	160,5
azula	feminin	176

3. Importez les données du tableau contenu dans le fichier tab4.xls dans l'objet R que vous appelerez avatar2. Le resultat de l'affichage à l'écran doit être:

nomlign	element
aang	air
katara	eau
sokka	NA
zuko	feu
toph	terre
iroh	feu
azula	feu

```
avatar1 <- read.csv("~/Bureau/COURS_R_FINANCES/TP1/tab3.csv")
avatar2 <- read.csv("~/Bureau/COURS_R_FINANCES/TP1/tab4.csv")
avatar3 <- read.csv("~/Bureau/COURS_R_FINANCES/TP1/tab5.csv")
```

4. Fusionner les données avatar1 et avatar2 par lignes, appeler ce nouveau tableau all.avatar

```
all.avatar <- rbind(avatar1,avatar2)
all.avatar <- rbind(avatar2,avatar1)
```

5. Que se passe t'il dans les cas suivant ? Commentez Réponse: ça ne marche pas

```
cbind(avatar1,avatar2)

colnames(avatar1)[1] <-'prenom'
rbind(avatar1,avatar2)
colnames(avatar1)[1] <-'nomligne'
```

6. Fusionner le tableau all.avatar avec avatar3, et nommer ce nouveau tableau allfusion.avatar:

```
allfusion.avatar <- merge(all.avatar, avatar3, by="nomligne")
```

4.5 Elimination des valeurs manquantes

Lorsque l'on souhaite connaître les valeurs manquantes (NA) d'un vecteur :

```
which(is.na( allfusion.avatar[,"element"]))
```

```
## [1] 5
```

Lorsque l'on souhaite connaître les valeurs manquantes (NA) de tout un tableau :

```
which(is.na(allfusion.avatar),arr.ind=TRUE)
```

```
##      row col
## [1,]   5   4
```

Enlevez du tableau allfusion.avatar la ligne avec cette valeur manquante:

```
na.omit(allfusion.avatar)
```

```
##   nomligne     sexe taille element
## 1     aang masculin  172,9     air
## 2    azula  feminin    176     feu
## 3     iroh masculin  160,5     feu
## 4   katara  feminin  175,2     eau
## 6     toph  feminin  150,2   terre
## 7     zuko masculin    172     feu
```