Atelier R - Outline

Renato Henriques-Silva

6 février 2019

Module 1

Arithmétique

Montrer brièvement que avec # on écrit des commentaires dans le script

Montrer brièvement tout les différentes types d'opération arithmétique.

- Addition: `+`
- Substraction: `-`
- Multiplication: `*`
- Division: `/`
- Exponent: `^`
- Modulo: `%%`

Variables

Affectation des variables et opérations

Classes de variables

- **numeric**, donc des nombres qui ont des valeurs décimals `3.82`
- **integer**, donc des nombres entier `3` - c'est une sous-classe de "numerics"
- **logical**, donc une variable booléene `TRUE` ou `FALSE`
- **character**, donc du texte (aussi appelé de string) 'r text_spec(" 'le chien' ", background = '#E6E6E6

Verifier les classes de chaque variable avec *class()*

Valeurs Manquantes

Montrer la différence entre NA et NULL

Objets

Montrer les différents objets sur R

- Vecteurs
- Matrices
- Arrays
- Data frames
- Listes

Vecteurs

Montrer comment créer des vecteurs (numeriques, characters, logiques)

- 1 élément
- i:j
- seq(start, end, by)

- c()

Montrer comment ajouter des noms aux vecteurs -> *names()* Montrer comme vérifier la taille du vecteur –> *length()*

Matrices

Montrer comment créer des vecteurs (numeriques, characters, logiques)

- matrix(c(1,2,3,5,6), nrow = 2, ncol = 3)
- matrix(c(1,2,3,5,6), nrow = 3, ncol = 2)
- matrix(seq(1,20,1), nrow=4, ncol=5)
- matrix(seq(1,20,1), nrow=4, ncol=5, byrow = TRUE)
- cbind(1:4,5:8)
- rbind(1:4,5:8)
- matrix(c("DOG","CAT","DOG","DOG"), nrow = 2, ncol = 2)

Montrer comment ajouter des noms aux matrices -> *rownames()* , *colnames()*

Montrer aussi dans la fonction *matrix()*

```
matrix(numeric(), nrow = i, ncol = j, dimnames = list(c("rownmaes"),c("colnames")))
```

Montrer commenr vérifier les dimensions d'une matrice *dim()*

Arrays

Montrer un example d'array

arr <- array(rep(1:20,3), dim=c(4,5,3))

dim(arr)

Facteurs

- Facteurs nominamles *factor()*
- Facteurs ordinales *factor(, order = TRUE, levels = c())*

Montrer un facteur de mâle / femelle (nominale) Montrer un facteur de temperature froid/tiède/chaud (ordinale)

Montrer le comptage des différentes catégories d'un facteur avec *summary()*

Montrer la différence des deux types de facteurs en utilisant des comparaisons *sexe[1] > sexe[2]* et *temp[1] > temp[2]*

Data Frames

Créer trois vecteur différent -> sexe [mâles/femelles], taille[numerique] et especе [Esox Lucius / Barbus Barbus]

- data.frame(especе,sexe,taille)

Créeer un vecteur age[numerique] et le concatener avec df -> cbind(df,age)

Vérifier la classe de df avec *class()*

Montrer comment vérifier la classe de chaque colonne dans un df -> *sapply(df,class)*

Montrer *dim()*, *head()*, *tail()* et *View()*

Listes

Montrer comment créer une liste

liste <- list(vec, mat, df)

Montrer comment nomer les différets compartiments de la liste

liste <- list(nom_a = vec, nom_b = mat, nom_c = df)

Montrer comment ajouter un nouvel élément dans la liste

liste <- c(liste, list(nouvel_elelment))

EXERCICES I

Indexation

L'indexation des différents objets ce fait de la manière suivante:

- vector –> `vec[i]`
- matrix –> `mat[i,j]`
- array –> `arr[i,j,k]`
- data.frame –> `df[i,j]`
- list –> `lis[[i]]`

Indexation des vecteurs

```
vec[1]

vec[i:j]

vec[length(vec)]

vec[c(3,4,5)]
```

Indexation des matrices

```
mat[9]
# ici il faut bien préciser que on peux aller
# chercher élément par élément (et que R cherche en suivant les colonnes)

# ATTENTION, l'operation en haut ne marche pas sur les data.frames

mat[1,2]

mat[3,]

mat[3:4,1:2]

mat[,ncol(mat)]

mat[,"Octobre"]
```

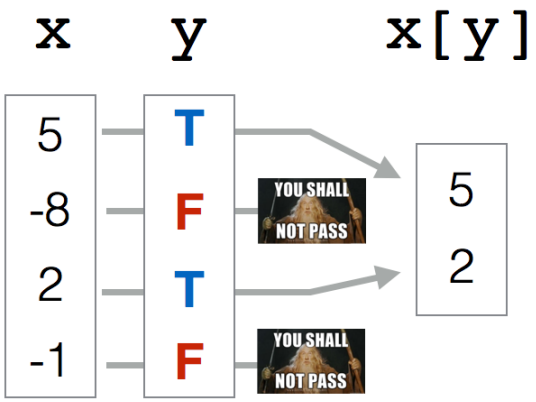
Indexation des data frames

```
df[1,2]

df[c(1,2),4]

df[5,"taille"]

df$espece[3]
```

Indexation des listes

```
liste_b[[1]]

liste_b[[3]][,1]

liste_b[[3]][,"espece"]

liste_b$camp_peche

liste_b$camp_peche$espece
```

Indexation par condition



Montrer une operation en créant à la main un vecteur logique

```
#vec <- c(1,2,3,4,5)

#vec[c(TRUE,FALSE,TRUE,FALSE,FALSE)]
```

Opérateurs de comparaison

- égale à: `==`
- différent de: `!=`
- strictement supérieur à: `>`
- strictement inférieure à: `<`
- supérieure ou égale à: `>=`
- inférieure ou égale à: `<=`

Montrer les operations de comparaision avec les vecteurs de voitures

```
chats.nom <-c("Fluffy","Billy","Minette","Patoune","Maya","Pilou","Luna", "Pacha","Minou","Gribouille")
chats.sexe <-c("F", "M", "F", "F", "F", "M", "F", "M", "M", "M")
chats.poids <- c(3.82,3.55,5.6,4,3.66,4.03,3.76,3.52,3.91,6.48)
chats.age <-c(10,4,3,2,11,1,8,9,7,6)
chats.sommeil <-c(18,13,12,13,18,14,15,17,17,18)
chats.race <-c("Persan","Siamois","Main Coon","Goutière","Siamois","Goutière","Angora","Persan","Persan


chats.sommeil < 15
chats.sexe == 'F'
chats.sexe != 'F'
chats.age[chats.age < 7]
chats.sommeil[chats.age >= 8]
```

Opérateurs logiques

- et `&`
- ou `|`
- n'est pas `!`
- est contenu dans `%in%`

```
chats.poids >= 3.6 & chats.poids < 4

chats.poids[chats.poids >= 3.6 & chats.poids < 4]

chats.sommeil[chats.sexe != 'M' & chats.poids < 4]

chats.nom[chats.sexe != 'M' & chats.poids < 4]

(chats.race == 'Siamois' | chats.race == 'Persan') & chats.age > 7

chats.nom[(chats.race == 'Siamois' | chats.race == 'Persan') & chats.age > 7]

chats.race == "Goutière" | chats.race == 'Main Coon' | chats.race == 'Angora'

chats.race %in% c('Goutière','Main Coon','Angora')
```

Comptages et percentages de vecteurs logiques

```
sum(chats.age >= 8)

 mean(chats.sommeil > 12 &  chats.sommeil < 15)*100

 chats.sommeil > 12 &  chats.sommeil < 15

# R interprète le TRUE comme 1 et le FALSE comme 0, si vous transformez le vecteurs en une variable num
 as.numeric(chats.sommeil > 12 &  chats.sommeil < 15)

# Il y a 3 TRUE sur un total de 10 éléments
(3/10) * 100
```

Autres fonctions importantes pour l'indexation

- is.na()
- duplicated()
- which()
- is.finite()
- max()
- min()
- which.min()
- which.max()

```
vec <- c(5,NA ,3, NA, 20, NA, 2, 3, 5, 7, 8, NA)
is.na(vec)

sum(is.na(vec))

duplicated(vec)

which(is.na(vec))

which(chats.race == 'Persan')

which(is.finite(vec))

vec <- c (1,20, 4 ,5, 6 , 8 , 9, 0.5)

which.max(vec)
vec[which.max(vec)]

which.min(vec)
vec[which.min(vec)]

min(vec)
max(vec)

which(vec == min(vec))
which(vec == max(vec))
```

Indexation par comparaison avec matrices et data.frames

```
mat.chats <-cbind(chats.age, chats.poids, chats.sommeil)
mat.chats
colnames(mat.chats)<- c("age","poids","sommeil")
mat.chats[mat.chats[,1]<5,]

mat.chats[mat.chats[,1] < 10 & mat.chats[,3] == 18,]

mat.chats[mat.chats[,"age"] < 10 & mat.chats[,"sommeil"] == 18,]


df.chats <- data.frame(nom = chats.nom, sexe = chats.sexe, race =chats.race, mat.chats, stringsAsFactors
df.voiture

df.chats[df.chats$race == "Persan",]
```

```r
df.chats[df.chats$race %in% c("Persan","Goutière"),]
subset(df.chats, subset = race %in% c("Persan","Goutière"))
# le resultat est un vecter data.frame

df2 <- subset(df.chats, subset = race %in% c("Persan","Goutière"), select = 'sommeil')

mean(df2)
#Pourquoi ca ne marche pas??
class(df2)
#Parce que la fonction subset retourne un data.frame (même si c'est juste un vecteur)
# Pour acceder au vecteur dans le data frame, il faut utiliser le "$"
mean(df2$sommeil)
```

Autres fonctions utiles

- *unique()*
- *order()*
- *table()*

```r
# Extraire les valeurs uniques du vecteur des "race"
races <- unique(df.chats$race)
# Calculer la longueur du vecteur
length(races)

# en une ligne
length(unique(df.chats$race))

# Donnees-moi l'indice d'ordre croissante des poids des chats
idx_poids <- order(df.chats$poid)
idx_poids

# Organiser le tableau données avec l'ordre des couts de voiture
df.chats[idx_poids,]

# Par défault order mets un ordre croissante. Si vous voulez l'ordre décroissante
df.chats[order(df.chats$poid, decreasing =TRUE),]

# Si vous voulez organiser le tableau par plusieurs colonnes. Par exemple, organiser
# l'ordre des cout et ensuite reordonner par l'ordre des prix
df.chats[order(df.chats$sommeil, df.chats$poid, decreasing =TRUE),]
# Il va donner toujours priorité à la première colonne

table(df.chats$race)

table(df.chats$race,df.chats$sexe)
```

Gestion du repertoire de travail et import/export des données

Reportoire

getwd()

list.files()

setwd()

paste('chien', 'et','chat')

Vous pouvez definir un séparateur entre les différents éléments

```
paste('chien', 'et','chat', sep ='.')
paste('chien', 'et','chat', sep ='+')
paste('chien', 'et','chat', sep ='/')
```

Imports de données

```
df <- read.table(file ='donnees_exemple.txt')

df <- read.table(file ='donnees_exemple.txt', sep=';')


df <- read.table(file ='https://raw.githubusercontent.com/RenatoHS/parallel_null_modelling/master/data/


args(read.table)

head(df)

str(df)


df <- read.table(file ='donnees_exemple.txt', sep=';', header= TRUE, stringsAsFactors = FALSE)

str(df)
```

Exports de données

```
 write.table(df.chats, file ='chats.txt', sep ='\t')

 # Vérifiez votre nouveau fichier.
```

Autres fonctions de gestion du répertoire de travail

ls() rm() rm(list = ls())

EXERCICES II

MODULE III

Les conditions If/Else

```
if(condition){

  CODE

}
```

IF

```r
x <- -3
if(x < 0){
  print('x est un nombre négatif')

}
# si on change x
x <- 4
if(x < 0){

  print('x est un nombre négatif')

}
# rien ne se passe
```

ELSE

```r
x <- 4

# si x est négatif
if(x < 0){
  # fais ceci

  print('x est un nombre négatif')

  # sinon
}else{
  # fais cela

  print('x est un chiffre positif')
}
```

ELSE IF

```r
x <- 0

if(x < 0){

  print('x est un nombre négatif')

}else if(x == 0 ){

  print('x est égale à 0')

}else{

  print('x est un chiffre positif')

}
```

CONDITIONS IF-ELSE EMBOITÉES

```r
# fonction sample pour choisir un chat aléatoirement
x <- sample(nrow(df.chats),1)

if(df.chats$race[x] == "Persan"){
```

```
  if(df.chats$sexe[x] == "M"){
    print("C'est un chat Persan mâle")

  }else{
    print("C'est un chat  Persan femelle")
  }
}else if (df.chats$race[x] == 'Main Coon'){

    if(df.chats$sexe[x] == "M"){
    print("C'est un chat Main Coon mâle")

  }else{
    print("C'est un chat Main Coon femelle")
  }
}else if (df.chats$race[x] == 'Siamois'){

  if(df.chats$sexe[x] == "M"){
    print("C'est un chat Siamois mâle")

  }else{
    print("C'est un chat Siamois femelle")
  }

}else if (df.chats$race[x] == 'Angora'){

  if(df.chats$sexe[x] == "M"){
    print("C'est un chat Angora mâle")

  }else{
    print("C'est un chat Angora femelle")
  }

}else{
  if(df.chats$sexe[x] == "M"){
    print("C'est un chat sans race mâle")

  }else{
    print("C'est un chat sans race femelle")
  }
}
```

Boucles

structure générale

```
for(iterateur in objet){

 CODE

}
```

```
chats.nom

for(chats in chats.nom){
```

```
  print(chats)

}
# si on veux avoir accès a l'index du loop pour indexer un autre objet, il faut le faire explicitement

for(i in 1:length(chats.nom)){

  print(paste(chats.nom[i],"a besoin de", chats.sommeil[i], 'heures de sommeil pour survivre'))

}

#R fait ceci

for(i in 1:10){

  print(chats.nom[i])
}
```

Break et Next

```
# Par exemple, si on voudrais que le for loop s'arrête dès qui rencontre un nom de chat avec 4 caractèr
# On utilise la fonction nchar() pour conter le nombre de caractère dans un string
# Il devrait s'arreter dès qui rencontre le nom "MAYA" qui est avant "PATOUNE"
for(chats in chats.nom){

  if(nchar(chats)== 4){
    break
  }

  print(chats)

}
# Ici il ignore le reste du code dans la boucle quand il exectute "next" et il passe à la prochaine ité
# Si notre code marche, il ne doit pas imprimer les noms "MAYA" et "LUNA"
for(chats in chats.nom){

  if(nchar(chats)== 4){
    next
  }

  print(chats)

}
```

For loop sur une liste

```
# version direct sans obtenir l'index
list.chats <- list(chats.nom, chats.poids, chats.age, chats.race, chats.sexe, chats.sommeil)

for (chat in list.chats){

  print(chat)
```

```
}

# version avec obtention de l'index du loop
for(ch in 1:length(list.chats)){

  print(list.chats[[ch]])


}
```

for loop sur une matrice et data.frame

```
# Supposez que les chats dorment le même nombre d'heures par jour tout les jours.
# On va créer un loop pour calculer combiens d'heures ils ont dormi toute leur vie.

for(i in 1:nrow(mat.chats)){

  # calculez le nombre de jours qui a vécu un chat
  # Assumez que toutes les années on 365 jours
  n_jours <- mat.chats[i,"age"]*365

  n_heures_sommeil <- n_jours * mat.chats[i,"sommeil"]


}
n_heures_sommeil
```

Est-ce qu'il a un problème dans ce loop?

```
# Créez un objet vide
n_heures_sommeil<-NULL
for(i in 1:nrow(mat.chats)){

  # calculez le nombre de jours qui a vécu un chat
  # Assumez que toutes les années on 365 jours
  n_jours <- mat.chats[i,"age"]*365

  n_heures_sommeil <- c(n_heures_sommeil, n_jours * mat.chats[i,"sommeil"])
}
n_heures_sommeil
```

PAS OPTIMALE –> allocation du vecteur

```
# Cette fois-si il faut utiliser NA et pas NULL parce que NULL n'occupe pas d'espace
n_heures_sommeil<-vector(NA, length = nrow(mat.chats))
for(i in 1:nrow(mat.chats)){

  # calculez le nombre de jours qui a vécu un chat
  # Assumez que toutes les années on 365 jours
  n_jours <- mat.chats[i,"age"]*365

  n_heures_sommeil[i] <-  n_jours * mat.chats[i,"sommeil"]
}
```

```r
n_heures_sommeil
```

MIX LOOP ET CONDITIONS IF

```r
n_heures_sommeil<-vector(NA, length = nrow(mat.chats))
for(i in 1:nrow(mat.chats)){

  if(chats.race[i] %in% c('Siamois','Persan')){
  # calculez le nombre de jours qui a vécu un chat
  # Assumez que toutes les années on 365 jours
  n_jours <- mat.chats[i,"age"]*365

  n_heures_sommeil[i,1] <-  n_jours * mat.chats[i,"sommeil"]
  }

}
n_heures_sommeil
```

Fonctions

INPUT -> BLACK BOX -> OUTPUT

Parler de la fonction SD

```r
# Quel est l'écart-type de "1, 4, 7 et 9"
sd(c (1, 4, 7, 9))

# Vous pouvez affecter ce vecteur à "chiffres"
chiffres <- c (1, 4, 7, 9)
# et ensuite faire
sd(chiffres)

# vous pouvez affecter le output de sd() aussi
sd_x <- sd(chiffres)
sd_x
```

```r
help(sd)

?sd
```

PARLER DES DEUX ARGUMENTS, UN OPTIONALE ET L'AUTRE OBLIGATOIRE

```r
sd()
```

MONTRER QUE LA POSITION EST IMPORTANTE

```r
sd(TRUE,chiffres)
```

```r
# Incluez des valeurs mancantes dans "chiffres"
chiffres <- c( chiffres, NA, NA)
chiffres

# Maintenant, si on ne spécifie pas na.rm, il ne vas pas éliminer les valeurs mancantes parce que le dé
sd(chiffres)
# Et le résultats est une valeur mancante.

# Changez na.rm pour TRUE
```

```r
sd(chiffres,TRUE)
# Bingo, vous obtenez l'écart-type
```

MONTRER QU'EN NOMMANT LES ARGUMENT LA POSITION N'EST PLUS IMPORTANTE

```r
sd(na.rm=TRUE,x = chiffres)
```

À retenir

- Les fonction de R: input –> blackbox –> output
- Les arguments des fonctions peuvent être spécifié par **position** ou par **nom**
- Quelques arguments des fonctions R ont déjá une option/valeur défault qui est reconnaisable par le "=".
- D'autres arguments n'ont pas d'option de défault, est la fonction retourne une erreur si ceux-ci ne sont pas spécifié.