Programmer en R

ING1 EC552 Introduction à R

Galharret Jean-Michel
département MSC
https://galharret.github.io/WEBSITE/cours_ONIRIS.html

Instructions conditionnelles :

if(condition){instruction} permet de calculer des instructions uniquement lorsque la condition est vraie.

Regarder l'aide des fonctions runif et paste pour comprendre ce qu'elles retournent.

```
x<-runif(1,-10,10)
if(x>0){print(paste("La valeur",round(x,1),"est positive",sep=" "))}
```

if(condition){instruction1}else{instruction2} permet de calculer l'instruction1 lorsque la condition est vraie et l'instruction2 lorsque la condition est fausse.

```
x<-runif(1,-10,10)
if(x>0){
  print(paste("La valeur",round(x,1),"est positive",sep=" "))}else{
  print(paste("La valeur",round(x,1),"est négative",sep=" "))}
```

On peut imbriquer plusieurs conditions if.

Rappel :

1. l'opérateur | correspond à OU : A | B est faux lorsque les deux événements A et B sont faux simultanément.
2. l'opérateur & correspond à ET : A & B est vrai uniquement lorsque les deux événements A et B sont vrais.

Exercice :

1. Choisir deux nombres x, y au hasard dans l'intervalle $[0, 10]$
2. La valeur stockée dans la variable z sera égale à $x + 1$ si $x < y$ sinon égale à $x + 2$ si $x > 5$ et $y > 5$ sinon égale à x.
3. Retourner les valeurs de x,y,z. On pourra utiliser la fonction paste().

```
x<-runif(1,0,10)
y<-runif(1,0,10)
if(x<y){z<-x+1}else if(x>5 & y>5){z<-x+2}else{z<-x}
print(paste("x=",x,"y=",y,"z=",z))
```

Boucle FOR, WHILE :

FOR

for (var in seq) {commandes} permet de définir un nombre d'itérations dans une séquence.

Par exemple on veut Stocker dans la variable x la somme des entiers de 1 à n où n est fixé par l'utilisateur :

```
n<-100
x<-0
for(i in 1:n){x<-x+i}
print(x)
```

Mais on peut aussi faire la somme de tous les entiers impairs compris entre 1 et n

```
n<-100
x<-0
impairs<-seq(1,n,2)
for(i in impairs){x<-x+i}
print(x)
```

Exercice : On considère deux variables x, y initialisées à 0. On propose le jeu suivant : "à chaque itération on tire un nombre au hasard entre 0 et 1, lorsque ce nombre est supérieur à 0.5 on incrémente la valeur précédente de y de 1, sinon c'est la valeur précédente de x qui est incrémentée de 1."

Le joueur gagne si au bout de n itérations $y > x$. Ecrire le programme pour $n = 10$.

```
n<-10
y=x=0
for(i in 1:n){
a<-runif(1,0,1)
if(a>0.5){y<-y+1}else{x<-x+1}
}
if(y>x){print("Gagnant")}else{print("Perdant")}
```

WHILE

while(condition){instruction} répète une instruction tant que la condition considérée est vraie. Attention, la condition est évaluée avant toute exécution dans while.

```
i<-1
while (i<10){
   print(i)
   i=i+1}
```

Exercice

1. Créer un vecteur nommé vecAlea de 100 valeurs entières entre 1 et 100. On utilisera la fonction ***sample*** avec remise.

a. déterminer le vecteur IND qui contient les indices des valeurs strictement supérieures à 50, (avec une boucle et sans une boucle)

b. déterminer le vecteur VEC contenant ces valeurs.

2. On calcule le maximum du vecteur vecAlea. Déterminer à l'aide d'une boucle le nombre de fois où ce maximum apparaît dans vecAlea. Retrouver ce nombre sans utiliser de boucle.

3. Créer le vecteur nommé vecPM5 contenant tous les nombres de 1 à 100 qui ne sont pas des multiples de 5 (avec une boucle puis sans). Penser à la fonction ***floor*** qui calcule la partie entière.

```
vecAlea<-sample(1:100,100,T)
## Q1
## AVEC la boucle
```

```r
IND<-c()
for(i in 1:100){
  if(vecAlea[i]>50){IND<-c(IND,i)}
  }

# SANS LA BOUCLE
IND<-which(vecAlea>50)

VEC<-vecAlea[IND]

## Q2
M<-max(vecAlea)

IND_M<-c()
for(i in 1:100){
  if(vecAlea[i]==M){IND_M<-c(IND_M,i)}
  }
length(IND_M)

length(which(vecAlea==M))

## Q3
x<-c()
i<-1
while(i<=100){
  if(i/5-floor(i/5)>0){x<-c(x,i)}else{x<-x}
  i<-i+1
}
```

Ecrire une fonction en R :

On veut écrire une fonction qui étant donnée le rayon r d'un cercle permet de calculer son périmètre $P = 2\pi r$, on note cette fonction perim()

```r
perim<-function(r){
  return(2*pi*r)
}
perim(1)
```

On remarque que dans le langage R il n'est pas nécessaire de donner le type des arguments (entier, carcatère,...) la fonction s'appliquera sauf si le type n'est pas correct

```
perim("rayon")
```

On peut améliorer cette fonction en indiquant à l'utilisateur que la fonction ne sera calculée que lorsque $r > 0$:

```
perim<-function(r){
  if(r>0){return(2*pi*r)}else{return("On ne calcule le périmètre que lorsque r>0")}
}
perim(1)
perim(-1)
```

Par contre l'erreur reste identique si on applique la fonction périmètre à "rayon". On peut également définir des fonctions qui ont plusieurs arguments en entrée et qui peuvent retourner plusieurs valeurs en sortie.

Autre Exemple : On va écrire une fonction restangle() ayant pour arguments L et l qui renvoie le périmètre $P = 2 \times (L + l)$ et l'aire $A = L \times l$ du rectangle.

```
rectangle<-function(L,l){
  P=2*(L+l)
  A=L*l
  return(list(Perim=P,Aire=A))
}
rectangle(11,10)
## Que le périmètre :
rectangle(11,10)$Perim
```

Exercice :

Créér une fonction SomEnt() ayant pour argument un nombre entier n et qui retourne la somme des entiers inférieurs à n.

```
SomEnt<-function(n){
  x<-0
  for(i in 1:n) x<-x+i
  return(x)
}
```

Exercices :

Exercice 1 (Droite des moindres carrés)

On va écrire une fonction R qui permet de déterminer la droite des moindres carrées ordinaires. Vous verrez (ultérieurment en stat) que pour un nuage de points $(x_i, y_i)_{i=1,\dots,n}$ la droite la plus proche (au sens des moindres carées) de ce nuage a pour pente $b = \frac{\frac{1}{n}\sum x_i y_i - \bar{x}\bar{y}}{\frac{1}{n}\sum x_i^2 - \bar{x}}$ et pour ordonnée à l'origine $a = \bar{y} - b\bar{x}$.

Créer une fonction nommée droite() ayant pour arguments deux vecteurs x,y et qui calcule le coefficient directeur de la droite et son ordonnée à l'origine.

```
droite<-function(x,y){
  b<-(mean(x*y)-mean(x)*mean(y))/(mean((x-mean(x))^2))
  a<-mean(y)-b*mean(x)
  return(list(a=a,b=b))
}
```

Exercice 2 : Boucles et graphiques

Reprendre la base de données iris :

```
data("iris")
```

1. Utiliser une boucle de manière à faire une boîte à moustaches par variable quantitative Sepal.Length, Sepal.Width, Petal.Length, Petal.Width selon l'espèce Species.

```
par(mfrow=c(1,1))
for(i in 1:4){
  boxplot(iris[,i]~iris$Species,
          ylab=colnames(iris)[i],
          xlab="Espèces")
  abline(h=mean(iris[,i]),col=2,lty=2,lwd=2)
}
```

2. Utiliser une boucle de reproduire le graphique suivant (on pensera à utiliser la fonction ***abline*** pour ajouter la droite des moindres carrés):

```
par(mfrow=c(1,3))
for(i in 1:3){
    esp<-levels(iris$Species)[i]
    D<-droite(iris$Sepal.Length[iris$Species==esp],iris$Sepal.Width[iris$Species==esp])
```

```
    plot(iris$Sepal.Length[iris$Species==esp],iris$Sepal.Width[iris$Species==esp],
         main=paste("Pour l'espèce",esp),xlab="L Sépale",ylab="l Sépale",pch=20)
    abline(a=D$a,b=D$b,col=2,lty=2,lwd=2)
    text(min(iris$Sepal.Length[iris$Species==esp])+1,
         max(iris$Sepal.Width[iris$Species==esp]),
         paste("y=",round(D$a,1),"+",round(D$b,1),"*x",sep=""),col=2,cex=0.75)

}
```

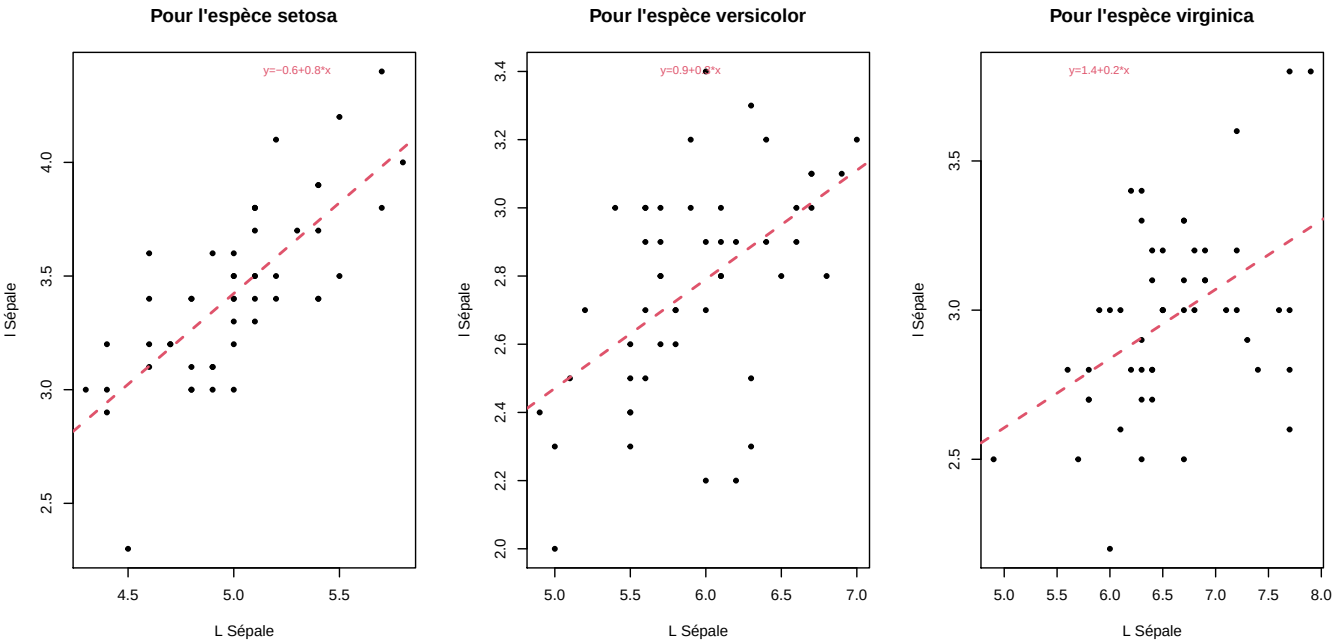

Figure 1: Trois graphes différents

3. Maintenant on fait les trois graphes sur le même environnement graphique, on pensera à faire un plot() sans points type="n".

```
par(mfrow=c(1,1))

plot(iris$Sepal.Length,iris$Sepal.Width,
     xlab="L (cm)",
     ylab="l (cm)",
     type="n"
     )
couleur<-c("purple","pink","blue")
Points<-c(20,17,18)
for(i in 1:3){
```

```
  I<-which(iris$Species==levels(iris$Species)[i])
  points(iris$Sepal.Length[I],iris$Sepal.Width[I],
       pch=Points[i],cex=.5,col=couleur[i])
  par<-droite(iris$Sepal.Length[I],
              iris$Sepal.Width[I])
  abline(a=par$a,b=par$b,
         col=couleur[i],lty=2,lwd=1.5)
}

legend("topright",levels(iris$Species),
       col=couleur,
       pch=Points,
       cex=.5)
```

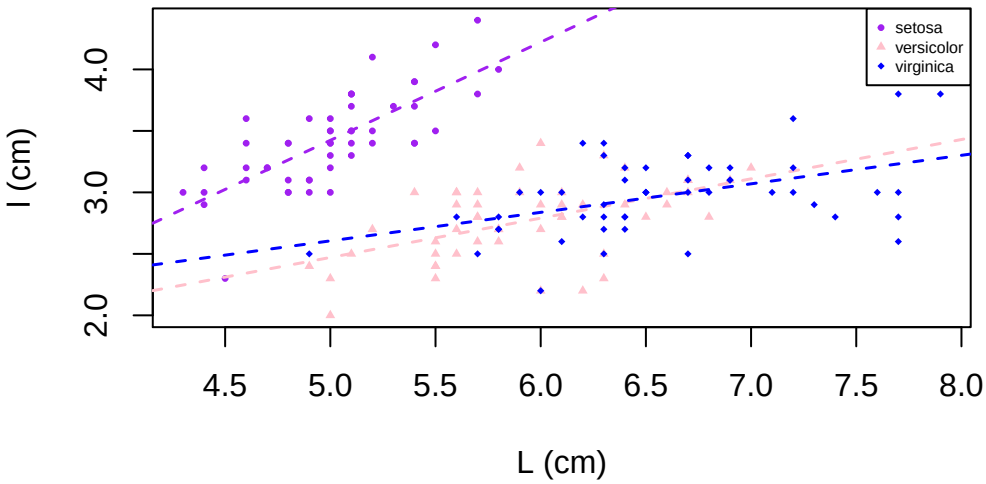

Figure 2: Trois graphes dans le même environnement