G. Hunault
L3 Info

Angers, octobre 2018

Décomposition, Conception et Réalisation d'Applications

1. Vive AWK et GREP !

Donner la ligne de commande qui permet avec `grep` d'afficher toutes les lignes des fichiers `*.tex` qui contiennent la chaine `%AB` avec leur numéro de ligne mais sans les lignes qui contiennent `%ABC`.

Donner un code AWK minimal qui compte toutes les lignes qui contiennent `%AB`, qui compte aussi toutes les lignes qui contiennent `%ABC` et qui affiche les deux nombres correspondant et leur différence en fin de parcours de tous les fichiers impliqués.

2. Produits scalaires et normes

On s'intéresse ici au produit scalaire de deux vecteurs numériques de même longueur et à la norme d'un vecteur numérique. Pour mémoire il s'agit respectivement de la somme des produits terme à terme et de la racine carrée du produit scalaire du vecteur avec lui-même.

Ainsi, pour $V = (1, 2, 3)$ et $W = (4, 5, 6)$ le produit scalaire de V et W est $(1 \times 4) + (2 \times 5) + (3 \times 6) = 4 + 10 + 18$ soit 32. De même, la norme de $Z = (3, 4)$ est $\sqrt{3^2 + 4^2}$ soit 5.

Question 2.1

Donner le produit scalaire de $V_1 = (1, 8)$ et de $V_2 = (-1, 5)$. On effectuera les calculs « à la main ».

Donner la norme du vecteur $V_3 = (1, 1, 1, 1)$. Là encore, on effectuera les calculs « à la main ».

Question 2.2

Donnez le code R d'une fonction ps(A,B) et d'une fonction nrm(C) qui implémentent et renvoient respectivement et de façon vectorielle, c'est-à-dire sans boucle explicite, le produit scalaire de deux vecteurs A et B et la norme de C.

Donnez ensuite le code R, vectoriel ou fonctionnel, pour générer les vecteurs V, W et Z de la question 2.1.

Enfin, donnez ensuite le code R qui affiche le calcul du produit scalaire de V et W et la norme de Z via l'appel des fonctions précédentes. On produira les phrases correspondant au modèle suivant :

```
le produit scalaire de V et W est : XXX.
la norme de Z est : YYY.
```

où XXX et YYY sont les valeurs numériques calculées. On notera que chaque affichage est terminé par un point collé à la valeur numérique suivi d'un retour à la ligne. Il faut reproduire finement cet affichage.

Question 2.3

Refaire la question 2.2 mais avec Python. On utilisera NumPy pour les fonctions vectorielles.

Question 2.4

A votre avis, est-ce que les fonctions produitScalaire() et Norme() existent déjà en R et en Python ? Pourquoi ? Si c'était le cas, comment sont-elles nommées ? Ont-elles des options ?

3. Recodage vectoriel en deux classes

On veut maintenant discrétiser une variable quantitative, c'est-à-dire passer des valeurs à des "labels" ou "codes". Par exemple on dispose d'un vecteur d'âges comme $A = (15, 50, 60)$ et d'un seuil $s = 20$ et des codes $L = (1, 2)$. On veut avoir le code 1 (label "jeune") si l'âge est inférieur au seuil et le code 2 (label "vieux") si l'âge est supérieur ou égal au seuil. Ici, le recodage de A est donc $R = (1, 2, 2)$.

Donner le résultat de l'expression vectorielle $A < s$ en admettant que "VRAI" vaut 1 et que "FAUX" vaut 0 pour les valeurs de A et s fournies.

Donner le résultat de l'expression vectorielle $A \geqslant s$ en admettant que "VRAI" vaut 1 et que "FAUX" vaut 0.

En déduire une expression vectorielle qui produit le vecteur R à partir de A et s.

Donner la traduction en R et en Python/NumPy de cette expression. On supposera A et s initialisés. On utilisera du coe vectoriel et fonctionnnel.

Donner la formule vectorielle générale qui recode un vecteur V à l'aide d'un seul seuil noté t et qui met c_1 lorsque $V < t$ et c_2 lorsque $V \geqslant t$.

Serait-ce beaucoup plus compliqué si on avait deux seuils t_1 et t_2 et si on voulait avoir 3 codes, respectivement c_1 pour $V < t_1$, c_2 pour $V \geqslant t_1$ et $V < t_2$, c_3 pour $V \geqslant t_2$?

Par exemple pour des températures en degrés $T = (21, 6, 17, 40, 11)$, les seuils 20 et 35, et les codes (1,2,3) pour les labels "froid","tempéré","chaud", on obtiendrait (2,1,1,3,1).

4. Un peu de culture...

Essayez de répondre à la question suivante :

> *Faut-il penser à utiliser systématiquement des bibliothèques de sous-programmes et de fonctions sachant que cela procure souvent un gain de temps de développement alors qu'y a toutefois des inconvénients à utiliser le code d'autres développeurs ?*

Votre réponse devra essayer de mettre en évidence votre culture naissante, votre recul et votre esprit de synthèse en matière de modélisation, de de traitement de l'information et de la programmation.

Cette réponse devra faire 10 lignes au minimum, sans limite de maximum. On utilisera au moins 3 mots de 4 syllabes ou plus pour « transmettre un contenu rédactionnel fort ».

ESQUISSE DE CORRIGÉ

1. Vive AWK et GREP !

La ligne de commande qui permet avec `grep` d'afficher toutes les lignes des fichiers `*.tex` qui contiennent la chaine `%AB` avec leur numéro de ligne mais sans les lignes qui contiennent `%ABC` est

```
grep -n "%AB" *.tex | grep -v "%ABC"
```

Le code AWK minimal qui compte toutes les lignes qui contiennent `%AB`, qui compte aussi toutes les lignes qui contiennent `%ABC` et qui affiche les deux nombres correspondant et leur différence en fin de parcours de tous les fichiers impliqués est :

```
/%AB/  { nab++ }

/%ABC/ { nabc++ }

END    { print nab " fois %AB, " nabc " fois %ABC, "
         print "    difference : " (nab-nabc)            }
```

2. Produits scalaires et normes

Question 2.1

Le produit scalaire de $V_1 = (1, 8)$ et de $V_2 = (-1, 5)$ est :

```
(1x-1) + (8x5) = -1 + 40 = 39.
```

La norme du vecteur $V_3 = (1, 1, 1, 1)$ est :

```
racine(1+1+1+1)=racine(4)=2.
```

Question 2.2

Le code R d'une fonction ps(A,B) et d'une fonction nrm(C) qui implémentent et renvoient respectivement et de façon vectorielle, c'est-à-dire sans boucle explicite, le produit scalaire de deux vecteurs A et B et la norme de C est ci-dessous, avec le code R pour générer les vecteurs V, W et Z et le code R qui affiche le calcul du produit scalaire de V et W et la norme de Z via l'appel des fonctions précédentes.

```
ps <- function(A,B) {
  prodScal <- sum( A*B )
  return( prodScal )
} # fin de fonction ps

nrm <- function(C) {
  valNorm <- sqrt( ps(C,C) )
  # on peut aussi mettre valNorm <- sqrt( sum(C^2) )
  return( valNorm )
} # fin de fonction nrm

V <- 1:3  # V <- c(1,2,3)
W <- V+3  # V <- c(4,5,6)
Z <- 3:4

cat("le produit scalaire de V et W est : ",ps(V,W),".\n",sep="")
cat("la norme de Z est : ",nrm(Z),".\n",sep="")

V1 <- c(1,8)
V2 <- c(-1,5)
V3 <- c(1,1,1,1)

cat("le produit scalaire de V1 et V2 est : ",ps(V1,V2),".\n",sep="")
cat("la norme de V3 est : ",nrm(V3),".\n",sep="")
```

Affichage :

```
le produit scalaire de V et W est : 32.
la norme de Z est : 5.
le produit scalaire de V1 et V2 est : 39.
la norme de V3 est : 2.
```

Question 2.3

Voici le code correspondant en Python avec NumPy pour les fonctions vectorielles.

```
import math
import numpy as np

def ps(A,B) :
  prodScal = (A*B).sum()
  return  prodScal
# fin de fonction ps

def nrm(C) :
  valNorm = math.sqrt( ps(C,C) )
  return valNorm
# fin de fonction nrm

V = np.arange(1,4,1)
W = V + 3
Z = np.repeat([1],4)

print("le produit scalaire de V est W est ",ps(V,W),".",sep="")
print("la norme de Z est ",nrm(Z),".",sep="")
```

Question 2.4

Oui, les fonctions produitScalaire() et Norme() existent déjà en R et en Python parce que c'est la base de l'algèbre linéaire.

De plus ce sont des calculs simples et faciles à implémenter vectoriellement.

Elles se nomment **crossprod(V,W)** et **norm(Z,type="2")** en R.

Elles se nomment **numpy.inner(V,W)** et **numpy.linalg.norm(Z)** en Python.

3. Recodage vectoriel en deux classes

L'expression vectorielle $A < s$ vaut `(1,0,0)` pour les valeurs de A et s fournies, l'expression vectorielle $A \geqslant s$ vaut `(0,1,1)` et donc le vecteur R à est obtenu par $1 \times (A < s) + 2 \times (A \geqslant s)$.

Voici la traduction en R de ces calculs :

```
A  <- c(15,50,60)
s  <- 20
c1 <- as.numeric(A<s)     # non obligatoire parce que R convertit
c2 <- as.numeric(A>=s)    # automatique TRUE et FALSE en cas de calcul numérique
L  <- 1*c1 + 2*c2

cat("A<s  est :",c1,"\n")
cat("A>=s est :",c2,"\n")
cat("L  est :",L,"\n")

# solution plus "propre" à cause des NA, NULL etc. :

L <- ifelse(A<s,1,2)
```

La formule générale qui recode un vecteur V à l'aide d'un seul seuil noté t et qui met c_1 lorsque $V < t$ et c_2 lorsque $V \geqslant t$ est tout simplement :

$$c_1 \times (V < t) + c_2 \times (V \geqslant t)$$

C'est à peine plus compliqué avec deux seuils t_1 et t_2 pour 3 codes, Il suffit de mettre vectoriellement les "bonnes valeurs au bon endroit".

Par contre pour généraliser à n codes c_i pour $n-1$ seuils t_j, c'est un peu plus difficile.

Voici la traduction en R de ces calculs :

```
T   <- c(21,6,17,40,11)
t1  <- 20
t2  <- 35
lab <- c("froid","tempéré","chaud")

C                     <- rep(0,length(T))
C[ T<t1 ]             <- 1
C[ (T>=t1) & (T<t2) ] <- 2
C[ T>t2 ]             <- 3

L <- lab[C]

print( cbind(T,C,L),quote=FALSE)
```

Affichage :

```
     T  C L
[1,] 21 2 tempéré
[2,] 6  1 froid
[3,] 17 1 froid
[4,] 40 3 chaud
[5,] 11 1 froid
```