G. Hunault

Angers, novembre 2020

Licence L3 Informatique

Décomposition, Conception et Réalisation d'Applications

1. Un peu d'APL

On suppose que V est un vecteur non vide d'entiers strictement positifs. Ce pourrait être par exemple V ← 10 1 5 qui est un vecteur de longueur 3 dont les éléments sont 10, 1 et 5.

1.1 - Que renvoie numériquement l'expression APL +/ V ⩾ ⍳ ⍴ V pour ce vecteur ?

Vous détaillerez votre réponse.

1.2 - Que renvoie numériquement l'expression APL +/ V ⩾ ⍳ ⍴ V dans le cas général ?

A quoi cela peut-il servir ?

1.3 - Donner l'équivalent en Python/NumPy de l'expression +/ V ⩾ ⍳ ⍴ V.

1.4 - Donner l'équivalent en R de l'expression +/ V ⩾ ⍳ ⍴ V.

1.5 - Donner l'équivalent en PHP 7 de l'expression +/ V ⩾ ⍳ ⍴ V

Pour les codes Python/NumPy, R et PHP 7 on n'utilisera aucune boucle explicite.

2. Pratique de GREP et AWK

Afin de comptabiliser les cas de contamination et d'hospitalisation liés au COVID-19 on suppose qu'on dispose de fichiers nommés dept*XX*.txt où *XX* est un numéro de département français, par exemple dept49.txt pour le Maine et Loire, dept17.txt pour la Charente Maritime.

Voici un extrait d'un tel fichier, en l'occurence dep49.txt :

```
NOM     Prénom   Ville    CodeP       ResTest  Hospit
DUPONT  Jean     Angers   49100          non
TROMP   Donale   Avrillé  49240          oui   oui
TROMP   Daniel   Avrillé  49240          oui   non
[...]
```

La ligne 1 contient toujours les entêtes de colonnes, les autres lignes suivent un même format : nom, prénom, ville, code postal, résultat du test (oui/non), hospitalisation (oui/non ou vide).

Pour simplifier, on admettra que nom, prénom et ville correspondent à un seul mot (ou « champ » au sens de GREP et AWK).

2.1 Commandes GREP

Quelle commande grep permet de compter dans le fichier dept49.txt le nombre de personnes pour lesquelles la ville est Angers sachant qu'aucun nom ou prénom n'est égal à Angers ?

Quelle commande grep permet de comptabiliser, fichier par fichier, dans les seuls fichiers dept49.txt et dept17.txt, le nombre de personnes testées ?

On admettra qu'on ne trouve pas deux fois la même personne dans ces fichiers et que le mot CodeP ne figure que dans les lignes d'entêtes de colonnes.

2.2 Script AWK

On veut maintenant compter le nombre de personnes en tout qui ont été hospitalisées pour tous les départements correspondant aux fichiers dept*.txt du répertoire courant.

Quel script AWK faut-il écrire pour obtenir des comptages et un affichage comme celui-ci ?

```
Résultats pour 40 départements : 1742 hospitalisations.
```

2.3 Commandes Unix/Linux

Quelle(s) commande(s) Unix/Linux ou quel script Awk permet de comptabiliser, dans les seuls fichiers dept49.txt et dept17.txt, le total de nombre de personnes testées ? On essaiera de faire au plus simple.

3. Tirage aléatoire du loto simplifié

Afin de comprendre si on a des chances de gagner au loto national, on décide de commencer par simuler un loto simplifié dont les règles sont les suivantes : un tirage correspond à p=5 nombres distincts entre 1 et n=30 ; on gagne lorsqu'on trouve le bon tirage.

On suppose qu'on dispose d'une fonction algorithmique nommée alea(a,b) qui renvoie en principe un nombre entier "aléatoire" entre a et b inclus, tous deux entiers aussi, sachant que rien ne garantit que deux appels successifs renverront des nombres différents.

3.1 Méthode pour un tirage

Expliciter une méthode pour construire un tableau TIRAGE de p entiers **distincts** compris entre 1 et n si on se sert de la fonction alea(a,b). On pourra initialiser TIRAGE avec une des méthodes vectorielles vues en T.D. Le tableau renvoyé sera trié par ordre croissant.

3.2 Algorithme pour un tirage

Donner l'algorithme correspondant à votre méthode. A défaut du langage algorithmique GALG, on utilisera des instructions en français. On pourra supposer toute fonction "raisonnable" sur tableau déjà existante. On l'écrira de telle sorte que ce soit une fonction nommée tirage().

3.3 Méthode pour gagner au loto

Voici la méthode pour tester comment gagner au loto simplifié : on se donne un tableau BONTIRAGE de 5 entiers distincts compris entre 1 et 30 trié par ordre croissant et on compte le nombre de tirages réalisés à l'aide de la fonction tirage() avant de retrouver BONTIRAGE.

Donner l'algorithme associé.

Est-ce une bonne méthode pour tester nos chances de gagner au loto ? Pourquoi ?

3.4 Tirage aléatoire du loto simplifié en PYTHON

Sachant que l'appel de la fonction algorithmique alea(a,b) s'effectue par randint(a,b) en PYTHON via le module random, donner le code de la fonction tirage() en PYTHON.

3.5 Tirage aléatoire du loto simplifié en R

Sachant que l'appel de la fonction algorithmique alea(a,b) se réalise par sample(a :b,1) en R, donner le code de la fonction tirage() en R.

3.6 Tirage aléatoire du loto simplifié en PHP

Sachant que l'appel de la fonction algorithmique alea(a,b) se réalise par rand(a,b) en PHP, donner le code de la fonction tirage() en PHP.

3.7 Tirage du loto simplifié dans une page Web

On décide maintenant de fournir une page Web pour montrer les résultats de plusieurs essais de calculs de tirages et de bons tirages dans une page Web pour p nombres entre 1 et n.

A-t-on besoin de PHP ou peut-on se contenter de Javascript ?

Combien de tests (et lesquels) faut-il écrire pour garantir que le code écrit est correct ?

4. Discussion culturelle

Essayez de répondre à la question suivante :

> *Faut-il systématiquement parcourir la liste des fonctions, des objets et des méthodes-objets de base d'un langage de programmation ?*

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

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 SOLUTION

1. Un peu d'APL

Si V est le vecteur 10 1 5, alors ρV vaut 3 et donc $\iota\rho$V est le vecteur 1 2 3. La comparaison vectorielle V $\geqslant \iota\ \rho$ V produit le vecteur 1 0 1 puisque, en APL, 0 correspond à FAUX et 1 à VRAI. +/ calcule la somme de ce vecteur, soit 2.

Dans le cas général, l'expression +/ V $\geqslant \iota\ \rho$ V compte le nombre de fois où un élément de V est supérieur ou égal à sa position dans V. Cela peut servir, si V contient une permutation des nombres de 1 à n, à savoir si les valeurs de V ressemblent à l'ordre naturel 1, 2, 3....

Le code PYTHON correspondant ressemble certainement à :

```
import numpy as np
v = np.array([10,1,5])
sum(v>=np.arange(1,1+v.shape[0]))
```

alors que le code R correspondant ressemble certainement à :

```
v <- c(10,1,5)
sum(v>=(1:length(v)))
```

Dans la mesure où PHP 7.0 dispose de fonctions sur tableau, une implémentation équivalente en PHP peut sans doute s'écrire :

```
<?php # nombre de valeurs supérieures ou égales à leur indice
      $v   = [10,1,5] ;
      function supOuEgal($valeur,$offset) { return($valeur>=1+$offset) ; } ;
      print( count(array_filter($v,"supOuEgal",ARRAY_FILTER_USE_BOTH)) ) ;
?>
```

Remarque : $\iota\ \rho$ V peut s'écrire range(1,count($V)) en PHP mais on ne l'utilise pas ici car la fonction array_filter via son troisième paramètre passe à la fois la valeur et l'offset à la fonction voulue. Comme pour PYTHON, on a ajouté 1 à l'offset pour obtenir l'indice (la position) de l'élément dans V.

On aurait aussi pu écrire, en R, le code `sum(v>=1:length(v))` mais c'est sans doute plus dangereux. Mettre des parenthèses pour l'opérateur `:` est une bonne habitude à prendre pour garantir la lisibilité du code.

2. Pratique de GREP et AWK

Il est sans doute prudent de chercher le mot Angers entouré d'espaces. La première commande grep demandée est

```
grep -c " Angers " dept49.txt
```

Pour la seconde commande grep demandée, on retire seulement les lignes d'entêtes, qui contiennent toutes le mot "CodeP" – en admettant que personne ne se nomme ou prénomme CodeP, soit la commande :

```
grep -c -v " CodeP " dept{49,17}.txt
```

Le script AWK doit afficher le nombre de départements, c'est-à-dire le nombre de fichiers car il y a un fichier par département. On nomme nbf ce nombre, qu'on incrémente lorsque FNR vaut 1. Une hospitalisation correspond à "oui" pour le mot 6, on nomme nbh la variable correspondante. Le script AWK doit donc ressembler à :

```
# nombre de départements (nbf) et d'hospitalisations (nbh)

BEGIN { nbf = 0 ; nbh = 0 } # initialisations

(FNR==1)     { nbf++ }      # nouveau fichier
($6=="oui")  { nbh++ }      # nouvelle hospitalisation

END { print " Résultats pour " nbf " départements : " nbh " hospitalisations." }
```

Pour compter le nombre total de personnes testées, il suffit sans doute de compter toutes les lignes, sauf les lignes d'entêtes, soit le script AWK suivant :

```
# nombre personnes testées (nbpt) : tout, sauf les lignes 1 de chaque fichier

BEGIN   { nbpt = 0 }
(FNR>1) { nbpt++   }
END     { print "en tout, " nbpt " personnes testées." }
```

On aurait pu utiliser la commande précédente `grep -c -v " CodeP " dept49,17.txt` mais il aurait fallu alors additionner les résultats, à savoir ici 250 et 127 dans l'exemple ci-dessous ce qui n'est pas aussi simple :

```
\grep -c -v CodeP dept{49,17}.txt
dept49.txt:250
dept17.txt:127
```

3. Tirage aléatoire du loto simplifié

3.1 Méthode pour un tirage

On initialise TIRAGE à p=5 fois la valeur 0. On effectue un premier appel de alea(1,n=30) qu'on met dans TIRAGE[1]. Ensuite, pour les $p-1$=4 valeurs suivantes, tant que le nombre fourni par alea(1,n) est déjà présent dans le tableau, on refait un appel de alea(1,n). Lorsqu'on dispose de p nombres distincts, on les trie par ordre croissant avec une fonction nommée tri(TABLEAU) qui existe sans doute déjà.

3.2 Algorithme pour un tirage

```
# tirage de loto simplifié, auteur : gh

fonction tirage()

  AFFECTER TIRAGE     <- tableauAuHasard(p,0,0)
  AFFECTER TIRAGE[1]  <- alea(1,n)
  POUR inb DE1A  (p-1)
      AFFECTER nbAuHasard <- alea(1,n)
      TANT_QUE estDansTableau(nbAuHasard,TIRAGE)
        AFFECTER nbAuHasard <- alea(1,n)
      FIN_TANT_QUE # nbAuHasard présent dans le tableau
      AFFECTER TIRAGE[1+inb] <- nbAuHasard
  FIN_POUR # inb
  AFFECTER TIRAGE <- tri(TIRAGE)

  RENVOYER TIRAGE

fin_fonction # tirage
```

3.3 Méthode pour gagner au loto

```
# simulation pour gagner au tirage de loto simplifié, auteur : gh

AFFECTER BONTIRAGE <- tirage()

AFFECTER nbEssais  <- 0
TANT_QUE tableauxDifferents(BONTIRAGE,tirage())
   AFFECTER nbEssais  <- nbEssais + 1
FIN_TANT_QUE #
ECRIRE " on a gagné au bout de " , nbEssais " essais "
```

Cet algorithme ne donne pas une méthode pour gagner au loto mais plutôt pour tester la qualité du générateur aléatoire.

3.4 Tirage aléatoire du loto simplifié en PYTHON

```
from random import randint
import numpy as np

def tirage() :

  TIRAGE     = np.zeros(5,dtype=int)
  TIRAGE[0]  = randint(1,30)
  for inb in range(1,5) :
      nbAuHasard = randint(1,30)
      while nbAuHasard in TIRAGE :
        nbAuHasard = randint(1,30) ;
      # fin de tant que
      TIRAGE[inb] = nbAuHasard
  # fin pour inb
  TIRAGE = np.sort(TIRAGE)

  return(TIRAGE)

# fin de fonction tirage
```

3.5 Tirage aléatoire du loto simplifié en R

```
tirage <- function() {

  TIRAGE     <- rep(0,5)
  TIRAGE[1]  <- sample(1:30,1)
  for (inb in (1:4)) {
      nbAuHasard <- sample(1:30,1)
      while (nbAuHasard %in% TIRAGE) {
        nbAuHasard <- sample(1:30,1)
      } # fin de tant que
      TIRAGE[1+inb] <- nbAuHasard
  } # fin pour inb
  TIRAGE <- sort(TIRAGE)

  return(TIRAGE)

} # fin de fonction tirage
```

Remarque :

Si on écrit directement `sample(1:30,5,replace=FALSE)}` en R, on obtient directement 5 nombres distincts entre 1 et 30.
L'équivalent PYTHON est `1+numpy.random.choice(30,5,replace=False)`.

.

3.6 Tirage aléatoire du loto simplifié en PHP

```
<?php

function tirage() {

  $TIRAGE     = array_fill(1,5,0) ;
  $TIRAGE[1]  = rand(1,30) ;
  for ($inb=1;$inb<=4;$inb++) {
      $nbAuHasard = rand(1,30) ;
      $essai = 1 ;
      while (in_array($nbAuHasard,$TIRAGE)) {
        $nbAuHasard = rand(1,30) ;
      } # fin de tant que
      $TIRAGE[1+$inb] = $nbAuHasard ;
  } # fin pour inb
  sort($TIRAGE) ;

  return($TIRAGE) ;

} # fin de fonction tirage

?>
```

3.7 Tirage du loto simplifié dans une page Web

Il est sans doute maladroit d'utiliser PHP car les simulations risquent de durer longtemps et l'exécution d'un script php est limitée dans le temps (en général à 30 secondes pour un serveur LAMP).

Il est donc sans doute plus prudent d'utiliser JavaScript.

Dans la mesure où n'y a pas de saisie utilisateur, les tests peuvent se limiter à des tests unitaires (alea() renvoie les bonnes valeurs si les paramètres sont bien typés et dans la bonne plage de variation, tirage() aussi) et à un test fonctionnel de durée pour la méthode de gain au loto (durée d'exécution en fonction de n et p).