G. Hunault
L3 Info

Angers, octobre 2019

Décomposition, Conception et Réalisation d'Applications

1. Question de cours

Détailler ce que produit l'expression APL suivante si N vaut 5, expression vue en cours et présentée dans le tuteur APL :

$$(N, N)\ \rho\ 1\ ,\ N\ \rho\ 0$$

2. Séries de mesures

Pour ce qui suit, on suppose qu'on dispose de deux fichiers de données nommés respectivement dataJ.txt et dataM.txt structurés et formatés de la même façon, à savoir, après la ligne d'en-ête, soit des lignes vides, soit des lignes avec trois informations par ligne, un identifiant, un numéro de mesure et une valeur. Voici un exemple de contenu (six lignes dont une ligne d'en-tête, la cinquième est vide) :

```
Id          Mesure   Valeur  # fichier dataJ.txt 15/02/2007
Angers_182    1       8.7
Nantes_183    1       7.71
Nantes_183    2       7.72

Angers_186    2      10.8
```

Question 2.1

Donner la commande UNIX qui permet d'afficher toutes les lignes, y compris les lignes vides, du seul fichier dataJ.txt sauf la première ligne.

Question 2.2

En déduire un enchainement de commandes UNIX avec tail, grep et wc qui permet de compter le nombre de lignes de mesures non vides en tout sur l'ensemble des deux seuls fichiers dataJ.txt et dataM.txt sachant qu'il existe des fichiers comme dataX.txt ou dataABC.txt dans le répertoire qu'on ne veut pas comptabiliser, bien sûr.

Question 2.3

Indiquer et expliquer par quoi il faut remplacer le mot CONDITION du script AWK suivant pour qu'il effectue le même comptage si on l'exécute par `awk -f dataJM.awk data[JM].txt`

```
## Fichier dataJM.awk  ; (gH) 11/02/2004

## comptage des lignes de mesures non vides dans dataJ.txt et dans dataM.txt
#  (on ne prend pas en compte la ligne 1 qui correspond au nom des colonnes)

BEGIN      { nbm = 0 }  # initialisation
CONDITION  { nbm++   }  # comptage
END        { print " en tout, il y a ",nbm," lignes de mesures non vides\n" }
```

Question 2.4

On voudrait réaliser le même comptage à l'aide d'un script PHP qui utilise soit la fonction file() soit la fonction file_get_contents() pour lire chacun des fichiers de données.

Expliquer quelle méthode vous utiliseriez dans ce script pour compter le nombre de lignes de mesures non vides en tout dans ces deux fichiers, en supposant qu'on a déjà testé qu'ils existent et qu'ils sont non vides.

Question 2.5

Donner le code PHP complet correspondant à votre méthode.

On nommera `$nbm` le nombre de lignes de mesures non vides.

3. Nombre de patients et séries de mesures

Question 3.1

Comment pourrait-on faire en AWK pour compter le nombre de patients distincts dans un ensemble de fichiers formatés comme auparavant ?

Ecrire le script AWK correspondant. On nommera `nbp` le nombre de patients distincts en tout.

Question 3.2

Comment pourrait-on faire en PYTHON pour compter le nombre de patients distincts dans un ensemble de fichiers formatés comme auparavant ?

Faudrait-il passer les fichiers à traiter en paramètres ou serait-ce mieux d'utiliser un fichier texte qui contient la liste des fichiers à traiter ?

Ecrire le script PYTHON correspondant.

Question 3.3

On suppose que toutes les données des fichiers dataJ.txt et dataM.txt ont été transférées dans la table Mesures2019 d'une base de données nommée MESURES, les noms des champs de la table étant identiques aux noms de l'en-tête commune des fichiers.

Quelle expression SQL permet de connaitre le nombre de patients distincts ?

Quelle expression SQL permet de connaitre l'id des cinq premiers patients ayant le plus de mesures (affichage par nombre de mesures décroissant puis par ordre alphabétique si égalité) ?

Question 3.4

On suppose maintenant que les données ont été transférées dans un *data frame* au sens de R, nommé DFM et dont les noms de colonnes sont identiques aux noms de l'en-tête commune des fichiers.

Quelle expression R basée sur les fonctions `length()` et `unique()` permet de connaitre le nombre de patients distincts ?

Quelle expression R basée sur les fonctions `length()` et `table()` permet de connaitre le nombre de patients distincts ? Est-ce plus efficace ? Comment le tester ?

4. Discussion culturelle

Essayez de répondre à la question suivante :

> *Est-ce qu'un environnement d'exécution avec accès aux variables et aux fonctions est vraiment un outil indispensable dans le cadre du développement de grands scripts ou de gros programmes ?*

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 ».

ANNEXE

Extrait de tail --help
==========

```
NAME
       tail - output the last part of files

SYNOPSIS
       tail [OPTION]... [FILE]...

DESCRIPTION
       Print  the  last  10  lines of each FILE to standard output.  With more
       than one FILE, precede each with a header giving the file name.

       With no FILE, or when FILE is -, read standard input.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -c, --bytes=[+]NUM
              output  the  last  NUM  bytes; or use -c +NUM to output starting
              with byte NUM of each file

       -f, --follow[={name|descriptor}]
              output appended data as the file grows;

              an absent option argument means 'descriptor'

       -F     same as --follow=name --retry

       -n, --lines=[+]NUM
              output the last NUM lines, instead of the last  10;  or  use  -n
              +NUM to output starting with line NUM
[...]
       -q, --quiet, --silent
              never output headers giving file names

       --retry
              keep trying to open a file if it is inaccessible

       -v, --verbose
              always output headers giving file names
```

ESQUISSE DE SOLUTION

1. Question de cours

L'expression $(N, N)\ \rho$ formate ce qui est à sa droite en renvoyant une matrice carrée de taille N, quitte à recycler (réutiliser) les valeurs jusqu'à en obtenir $N \times N$.

L'expression $1\ ,\ N\ \rho\ 0$ correspond au vecteur composé d'un 1 suivi de N zéros. Comme il y a $N + 1$ valeurs dans cette expression, lorsque APL recycle ce vecteur en une matrice $N \times N$, le 1 se trouve décalé d'une position à chaque ligne.

L'ensemble de l'expression renvoie donc la matrice identité de taille N, soit ici, pour $N \leftarrow 5$:

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

Question 2.1

Il faut commencer l'affichage à la ligne 2. L'aide de la commande `tail` fournie en annexe permet de trouver qu'il faut écrire `tail --lines=+2 dataJ.txt`.

Question 2.2

L'aide de la commande `tail` fournie en annexe indique que pour plus d'un fichier, `tail` affiche par défaut le nom du fichier (*header* ou en-tête).

Il faut donc commencer par écrire `tail -q --lines=+2 data[MJ].txt`.

Pour ne pas comptabiliser les lignes vides, if faut passer par grep, soit `grep -v "^$"`.

Enfin, il ne reste plus qu'à compter avec `wc -l`.

L'enchaînement de commandes demandé est donc

```
tail -q --lines=+2 data[MJ].txt | grep -v "^$"  | wc -l
```

Question 2.3

Il faut ignorer la ligne 1 de chaque fichier, ce qui se fait avec la condition FNR>1 en AWK. Il faut aussi ignorer les lignes vides, soit la condition NF>0. Au final, la condition doit donc s'écrire (FNR>1) && (NF>0).

Question 2.4

Il faut certainement passer par un tableau qui contient la liste des fichiers à traiter.

Pour chaque fichier à traiter, on lit chaque ligne de ce fichier. On numérote les lignes au fur et à mesure, ce qui permet d'igorer la ligne 1 de chaque fichier. Un test sur la longueur de la ligne permet de savoir s'il faut la comptabiliser ou non.

Si on lit avec `file()`, il faut retirer les caractères de fin de ligne avant de tester si la ligne est vide ou non. Si on lit avec `file_get_contents()`, il faut découper les lignes de la la chaine lue avec `preg_split()` avant de pouvoir compter les lignes de mesures.

Question 2.5

Voici un exemple de code solution qui utilise `file_get_contents()`.

```
<?php

## comptage des lignes de mesures non vides dans dataJ.txt et dans dataM.txt
#  (on ne prend pas en compte la ligne 1 qui correspond au nom des colonnes)

$nbm    = 0 ;

$lstFic = array("dataJ.txt","dataM.txt") ;
foreach ($lstFic as $fichier) {
  $strFic = file_get_contents($fichier) ;
  $tabf   = preg_split("/\n/",$strFic) ;
  $nbl    = 0 ;
  foreach ($tabf as $ligne) {
    $nbl++ ;
    if ($nbl>1) { # omission de la ligne 1
       $ligne = strtr($ligne,array("\n"=>"","\r"=>"")) ;
       if (strlen($ligne)>0) { $nbm++ ; } ; # que des lignes non vides
    } # fin si
  } # fin pour chaque ligne du fichier
} # fin pour chaque fichier

echo " en tout, il y a $nbm lignes de mesure non vides.\n"  ;
?>
```

La solution la plus concise est sans doute celle qui utilise des paramètres supplémentaires de `file()` et qui retire 1 à chaque taille de tableau puisqu'il y a toujours une ligne d'en-tête, soit l'extrait de code modifié :

```
   foreach ($lstFic as $fichier) {
     $tabf = file($fichier,FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES ) ;
     $nbm += count($tabf) - 1 ;
   } # fin pour chaque fichier
```

Question 3.1

On peut certainement utiliser les tableaux associatifs de AWK pour résoudre le problème. Lorsque le comptage d'un patient vaut 1, il s'agit d'un nouveau patient, soit le code :

```
## nombre de patients distincts ; s'utilise par awk -f nbpat.awk data*.txt

BEGIN              {  nbp = 0 }
(FNR>1) &&(NF>0)   {  datap[$1]++ ; if (datap[$1]==1) { nbp++ }  }
END                {  print nbp " patients distincts en tout\n"  }
```

Question 3.2

Les dictionnaires de PYTHON sont l'équivalent des tableaux associatifs de AWK, le comptage du nombre de patients distincts est donc très similaire au point de vue code.

Il est sans doute « raisonnable » de passer par un `glob` en PYTHON pour traiter les fichiers car c'est ce qui ressemble le plus à AWK et qui est le plus simple à écrire. On peut faire un peu plus court qu'en AWK car le mot clé `in` permet de tester si une clé est présente dans le dictionnaire. Lorsque ce n'est pas le cas, il s'agit d'un nouveau patient.

```
## nombre de patients distincts ; s'utilise par python nbpat.py "data*.txt"

import sys
import glob

if (len(sys.argv)==1) :
   print " syntaxe : python nbpat.py \"FICHIERS\"\n"
   sys.exit()

lstFic   = glob.glob(sys.argv[1])
nbp      = 0
patients = {}  # dictionnaire des noms de patients

for nomFichier in lstFic :
    with open(nomFichier) as fichier :
         nbl = 0
         for ligne in fichier :
             nbl += 1
             if (nbl>1) : # on saute la ligne 1
                mots = ligne.split()
                if (len(mots)>0) :  # on ignore les lignes vides
                   patient = mots[0]
                   if  not (patient in patients) : # nouveau patient
                       nbp += 1
                       patients[ patient ] = 1
# fin pour chaque fichier

print("%s" % nbp + " patients en tout\n")
```

Il est possible d'utiliser la notion d'ensemble mathématique (`set` en anglais) et de profiter de la fonction `readlines()` en PYTHON qui renvoie un tableau pour démarrer directement à la ligne 2 (notation `[1:]`). On peut alors utiliser les listes en définition `[ ... if ... for x  in ...]` afin de garder tous les noms de patients sans s'occuper des lignes vides car il suffit d'enlever 1 au nombre d'éléments de l'ensemble en fin de script.

Voici le code (merci BENOIT DA MOTA) :

```
## nombre de patients distincts ; s'utilise par python nbpatv2.py "data*.txt"

import sys
import glob

if (len(sys.argv)==1) :
   print " syntaxe : python nbpatv2.py \"FICHIERS\" # les \" sont obligatoires \n"
   sys.exit()

lstFic   = glob.glob(sys.argv[1])
patients = set()  # ensemble des noms de patients

for nomFichier in lstFic :
    with open(nomFichier) as fichier :
        patients = patients.union( set([
           ligne.split()[0] if len(ligne.split()) > 0 else ''
           for ligne in fichier.readlines()[1:]
        ]))
# fin pour chaque fichier
nbp = len(patients) - 1

print("%s" % nbp + " patients en tout\n")
```

Question 3.3

Le nombre de patients distincts s'obtient simplement en MYSQL par

```
USE MESURES ;
SELECT COUNT(DISTINCT Id) as nbPatientsDistincts FROM Mesures2019 ;
```

Pour le nom des cinq premiers patients ayant le plus de mesures (affichage par nombre de mesures décroissant puis par ordre alphabétique si égalité), il faut compter par patient (`COUNT(...)` ... `GROUP BY`), trier sur le résultat (`ORDER BY`) et se limiter aux cinq premiers résultats (`LIMIT`), soit l'expression

```
USE MESURES ;
 SELECT Id,   COUNT(Mesure) AS nbm
        FROM  Mesures2019
        GROUP BY Id
        ORDER BY nbm DESC, Id ASC
        LIMIT 5 ;
```

Question 3.4

L'expression `length(unique(DFM$Id))` fournit le nombre de patients distincts, de même que l'expression `length(table(DFM$Id))`.

Dans la mesure où `table()` comptabilise le nombre de fois où chaque identifiant est vu, la deuxième expression est certainement beaucoup plus lente que la première. Comme il ne s'agit que d'expressions et pas de code complet, on peut utiliser `microbenchmark` pour le vérifier.

Avec un de nos fichiers d'exemple (3988 lignes, 434 patients distincts) le résultat est sans appel : la première expression est dix fois plus rapide que la seconde.

```
> microbenchmark( length(unique(dataM$Id_patient)) , length(table(dataM$Id_patient)))
# Unit: microseconds
                            expr     min      lq      mean  median       uq     max
  length(unique(dataM$Id_patient))  42.327  42.865  43.57648  43.623  44.1745  46.092
  length(table(dataM$Id_patient))  425.796 427.508 430.29487 428.743 430.3265 523.723

> microbenchmark( length(unique(dataM$Id_patient)) , length(table(dataM$Id_patient)))
                            expr     min       lq      mean   median      uq     max
  length(unique(dataM$Id_patient))  42.617  43.5035  47.27833  44.2420  45.019  92.982
  length(table(dataM$Id_patient))  425.557 428.6160 443.02367 431.8555 438.873 668.410

> microbenchmark( length(unique(dataM$Id_patient)) , length(table(dataM$Id_patient)))
                            expr     min      lq      mean   median       uq     max
  length(unique(dataM$Id_patient))  42.215  43.124  44.08283  43.8585  44.3535  67.359
  length(table(dataM$Id_patient))  425.253 427.585 430.89622 428.5635 430.5850 500.224

> microbenchmark( length(table(dataM$Id_patient)) , length(unique(dataM$Id_patient)))
                            expr     min       lq     mean   median       uq     max
  length(table(dataM$Id_patient)) 427.320 437.0295 442.6394 439.9925 442.7055 532.801
  length(unique(dataM$Id_patient)) 42.593  43.9510  45.4746  44.5140  45.4130  85.226

> microbenchmark( length(table(dataM$Id_patient)) , length(unique(dataM$Id_patient)))
Unit: microseconds
                            expr     min       lq      mean   median       uq     max
  length(table(dataM$Id_patient)) 427.690 429.8975 436.86607 431.2150 433.6925 532.525
  length(unique(dataM$Id_patient)) 42.413  43.1330  44.30165  44.0785  44.6060  75.639
```