G. Hunault

Angers, décembre 2019

L3 Info

Décomposition, Conception et Réalisation d'Applications

1. Question de cours

On suppose qu'un site *Web* a été conçu selon une architecture REST et qu'on peut y trouver des cours en ligne à partir de son adresse (fictive) `https://kursera.org`.

Indiquer quelle URI permet d'obtenir :

1.1 - la liste des cours ;

1.2 - les informations sur le cours 5 ;

1.3 - le panneau d'édition du cours 50 ;

1.4 - le troisième chapitre du cours 123 ;

Vous utiliserez uniquement les mots `cours`, `chapitre`, `edite` (sans accent).

2. Question de présence en cours

2.1 - quel animal a été présenté dans le dernier cours en amphithéâtre ?

2.2 - dans quel contexte (codage, développement, test, interfaçage, application Web...) ?

2.3 - cet animal est en couverture d'un livre très fortement recommandé ; lequel ?

Pour mémoire, le dernier cours dont il est fait référence a eu lieu le vendredi 18 octobre de 8 h à 10 h 50. Le livre et l'animal ont également été vus en T.D.

3. Vive la ligne de commandes !

Pour ce qui suit, on suppose qu'on est dans un répertoire Linux qui contient de nombreux fichiers et de nombreux sous-répertoires, ce qui exclut d'utiliser la commande `ls`.

Indiquer quel enchaînement de commandes permet d'obtenir avec `find` :

3.1 - le nombre de fichiers en tout, si on inclut tous les sous-répertoires

3.2 - le nombre de fichiers en tout dont le nom se termine par `.php`

3.3 - le nombre de fichiers en tout dont le nom se termine par `.htm` ou par `.html`

3.4 - le nombre de fichiers en tout dont le nom se termine par `.php`, `.htm` ou `.html`

Vous utiliserez uniquement les commandes `find`, `grep` et `wc`. On ne se préoccupera pas de savoir si les fichiers . et .. sont comptabilisés.

4. Questions de T.P.

4.1 - *Expressions régulières*

L'expression régulière `^[0-9]+$` n'accepte qu'un seul nombre entier, positif ou nul.

Quelle expression régulière permet de n'accepter qu'un seul nombre entier **strictement positif** ?

4.2 - *Nommage de colonnes comme Excel*

Sous Excel, la colonne 1 se nomme A, la colonne 26 se nomme Z et la colonne 27 se nomme AA. Comment se nomme la colonne 2020 ?

Vous n'oublierez pas de justifier et de détailler votre calcul.

5. Un peu de PYTHON

On appelle *consensus* d'une liste de caractères le caractère le plus présent. Par exemple le consensus de `ldCar=["A", "C", "A"]` est "A".

5.1 - *Ambiguité de la définition*

Expliquez pourquoi cette définition de *consensus* est incorrecte, incomplète ou ambigue. Vous donnerez des exemples (courts) pour de telles listes `ldCar` afin de justifier votre analyse.

5.2 - *Nouvelle définition*

On décide de modifier la définition précédente. Désormais, on appelle *consensus-p* d'une liste de caractères le caractère le plus présent de la liste si son pourcentage de présence dans la liste est supérieur ou égal à p % où p est un entier positif.

Est-ce que cette définition de *consensus* est incorrecte, incomplète ou ambigue ? Là encore, vous donnerez des exemples (courts) pour justifier votre analyse.

5.3 - *Méthode pour programmer un consensus-51* de caractères

On suppose qu'on dispose en PYTHON d'une variable de type liste de caractères, comme par exemple `ldCar=["A","C","A","T","A"]`. Si on voulait calculer le *consensus-51* d'une telle liste en PYTHON, quelle méthode utiliseriez-vous ?

5.4 - *Programmation d'un consensus-51* de caractères

Donner le code d'une fonction PYTHON nommée `consensus51` qui prend comme paramètre une liste de caractères nommé `ldCar` et qui renvoie, s'il existe, le *consensus-51* de la liste selon votre méthode ainsi que toute autre information que vous jugez pertinente.

Que doit renvoyer, selon vous, la fonction s'il n'y a pas de *consensus-51* dans la liste ?

5.5 - *Méthode pour programmer un consensus-51 de séquences*

On généralise la notion de *consensus-p* d'une liste de caractères à celle de *consensus-p* d'une liste de chaines de caractères de même longueur. Ainsi le *consensus-51* de la liste `ldSeq=["AC","CC","AC","TC","AA"]` s'obtient en prenant le *consensus-p* de la liste de tous les premiers caractères, puis le *consensus-p* de la liste de tous les seconds caractères etc. On obtiendrait ici `["AC"]`.

Si on voulait calculer le *consensus-51* d'une telle liste en python, quelle méthode utiliseriez-vous ?

5.6 - *Programmation d'un consensus-51 de séquences*

Donner le code d'une fonction PYTHON nommée `consensus51seq` qui prend comme paramètre une liste nommée `ldSeq` de chaines de caractères de même longueur et qui en renvoie le *consensus-51* sous forme d'une chaine unique de même longueur que les chaines initiales.

On utilisera obligatoirement la fonction `consensus51` dans le code de la fonction `consensus51seq`.

6. Un peu de R

6.1 - *Méthode pour programmer un consensus-51 de caractères*

Si on voulait calculer en R le *consensus-51* d'un vecteur de caractères comme `vdCar <- c("A","C","A","T","A")`, quelle méthode utiliseriez-vous ?

Est-ce que le fait que R dispose d'instructions vectorielles change quelque chose ici ?

6.2 - *Programmation d'un consensus-51 de caractères*

Donner le code d'une fonction R nommée `consensus51` qui prend comme paramètre un vecteur `vdCar` de caractères et qui renvoie, s'il existe, le *consensus-51* de la liste selon votre méthode ainsi que toute autre information que vous jugez pertinente. Que doit renvoyer, selon vous, la fonction s'il n'y a pas de *consensus-51* dans le vecteur ?

6.3 - *Méthode pour programmer un consensus-51 de séquences*

Si on voulait calculer le *consensus-51* d'un vecteur `vdSeq` de chaines en R, quelle méthode utiliseriez-vous ?

6.4 - *Programmation d'un consensus-51 de séquences*

Donner le code d'une fonction R nommée `consensus51seq` qui prend comme paramètre un vecteur `vdSeq` de chaines de caractères de même longueur et qui en renvoie le *consensus-51* sous forme d'une chaine unique de même longueur que les chaines initiales.

On utilisera obligatoirement la fonction `consensus51` dans le code de la fonction `consensus51seq`.

7. Discussion culturelle

Essayez de répondre à la question suivante :

> *Est-ce difficile d'écrire des tests quand on ne sait pas quelles erreurs les utilisateurs et utilisatrices risquent de faire ?*

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 SOLUTION

1. Question de cours

1.1 - la liste des cours
`https://kursera.org/cours`
1.2 - les informations sur le cours 5
`https://kursera.org/cours/5`
1.3 - le panneau d'édition du cours 50
`https://kursera.org/cours/50/edit`
1.4 - le troisième chapitre du cours 123
`https://kursera.org/cours/123/chapitre/3`

2. Question de présence en cours

2.1 - l'animal était un bouc (ou une chèvre), en anglais *goat*.

2.2 - c'était dans le cadre de tests, notamment de pages Web.

2.3 - l'ouvrage, disponible aux éditions O'Reilly, se nomme Test-Driven Development with Python, obey the testing goat using Django, Selenium and Javascript ; son auteur est H. J.W. Percival ; dans le T.D. 4 on voit même la couverture du livre.

3. Vive la ligne de commandes !

3.1 - le nombre de fichiers en tout, si on inclut tous les sous-répertoires :
`find . | wc -l`
3.2 - le nombre de fichiers en tout dont le nom se termine par `.php` :
`find . | grep "\.php$" | wc -l`
3.3 - le nombre de fichiers en tout dont le nom se termine par `.htm` ou par `.html` :
`find . | grep "\.html\?$" | wc -l`
3.4 - le nombre de fichiers en tout dont le nom se termine par `.php`, `.htm` ou `.html` :
`find . | grep "\.php$\|\.html\?$" | wc -l`

4. Questions de T.P.

4.1 - *Expression régulière pour un nombre strictement positif*

Il suffit de rajouter les nombres de 1 à 9 au début de l'expression proposée puis de remplacer le répétiteur `+` par `*`, soit l'expression `^[1-9][0-9]*$`.

4.2 - *Nommage de colonnes comme Excel*

La division euclidienne de 2020 par 26 est $77 \times 26 + 18$. La division euclidienne de 77 par 26 est $2 \times 26 + 25$. On en déduit que l'écriture en base 26 de 2020 est $(2 \times 26 + 25) \times 26 + 18 = 2 \times 26^2 + 25 \times 26 + 18$. Comme les lettres correspondant à 2, 25 et 18 sont respectivement B, Y et R on peut affirmer que sous Excel la colonnes 2020 se nomme BYR.

5. Un peu de PYTHON.

5.1 - *Ambiguité de la définition*

Cette définition est incomplète car elle ne dit pas ce qu'on doit renvoyer pour une liste vide, donc pour l'exemple `ldCar=[]`.

Cette définition est ambigue car elle ne dit pas ce qu'on doit renvoyer si deux caractères sont tous les deux présents un nombre maximum de fois, par exemple pour la liste `ldCar=["A","C"]`.

5.2 - *Nouvelle définition*

La nouvelle définition est toujours incomplète pour le cas de la liste vide.

Elle reste ambigue pour $p \leqslant 50\%$. L'exemple précédent `ldCar=["A","C"]` le montre. L'exemple `ldCar=["A","B","C"]` aussi.

A partir de $p = 51\%$ il n'y a plus d'ambiguité car on ne peut pas avoir deux caractères dont les pourcentages dépasssent $p = 51$ % vu que le total fait 100 % :

$$\Sigma\ x_i = 100 \text{ et } \forall i, x_i \geqslant \text{ et } x_1 \geqslant 51 \implies x_j < 51 \text{ pour } j > 1$$

5.3 - *Méthode pour programmer un consensus-51* de caractères

Le plus simple en PYTHON « pur » est d'utiliser un dictionnaire pour comptabiliser chacun des caractères.

Ensuite on calcule les pourcentages, on cherche le maximum et le caractère correspondant. Il faut prévoir les mauvais cas : on renvoie `"?"` si la liste est vide, `"<"` si le plus grand pourcentage est inférieur à 51. Dans la mesure où on aura sans doute besoin de donner le plus grand pourcentage, le mieux est de renvoyer le caractère consensus et son pourcentage (ce qui n'était pas explicitement demandé), soit le code de la page suivante.

Une autre solution serait de passer par pandas et de traiter les caractères comme une variable qualitative comme vu en T.D.

5.4 - *Programmation d'un consensus-51* de caractères

```
# consensus d'une liste de caractères

def consensus(ldCar,seuilPct=51) :

   # on sort de suite si c'est la liste vide

   if (len(ldCar)==0) :
      return(["?",0])

   # calcul des effectifs puis des pourcentages

   pcts = {}
   for c in ldCar :
       pcts[c] = pcts.get(c,0) + 1
   # fin pour

   ############### autre solution comme en T.D. :

   ## for c in ldCar :
   ##    if not c in pcts :
   ##       pcts[c] = 1
   ##   else :
   ##       pcts[c] = pcts[c] + 1
   ## fin pour

   for c in pcts :
       pcts[c] = 100*pcts[c]/len(ldCar)
   # fin pour

   # recherche du pourcentage maximal

   pctList   = list(pcts.values())
   indMax    = pctList.index(max(pctList))
   pctMax    = pctList[ indMax ]
   carList   = list(pcts.keys() )
   carPctMax = carList[ indMax ]

   # si le pourcentage max est inférieur au seuil maximal
   # on renvoie "<" sinon, on renvoie le caractère

   if (pctMax < seuilPct) :
       return(["<",carPctMax+str(pctMax)])
   else :
       return( [carPctMax,pctMax] )

# fin de fonction consensus
```

```
## quelques exemples :

l1 = []
print(consensus51(l1))
*** ['?', 0]

l2 = ["A","C"]
print(consensus51(l2))
*** ['<', 'A50.0']

l3 = ["A","C","A"]
print(consensus51(l3))
*** 'A', 66.66666666666667]
```

On pourrait croire que notre solution n'est pas totalement générale puisqu'elle renvoie les caractères `"?"` et `"<"` dans les mauvais cas. Toutefois, le fait de renvoyer aussi le pourcentage permet de savoir si éventuellement ces caractères étaient les plus présents, comme avec l'exemple `ldCar=["?","<","?"]`.

La fonction fournie en solution, nommée *consensus* est paramétrée, la fonction *consensus-51* demandée peut être définie par

```
def consensus51(ldCar) :

    # calcule un consensus à un seul caractère
    # le consensus est valide au-delà d'un certain pourcentage (ici 51 %)

    return( consensus(ldCar,51) )

# fin de fonction consensus51(ldCar)
```

mais on peut aussi l'utiliser comme un équivalent fonctionnel de *consensus-51* puisque la valeur par défaut du seuil de pourcentage à dépasser est 51.

5.5 - *Méthode pour programmer un consensus-51 de séquences*

Il suffit sans doute d'une boucle sur chaque k-ième caractère de chaque séquence pour calculer le k-ième caractère de la séquence consensus.

En d'autres termes : si `lngC` est la longueur commune des séquences, on fait une boucle `pour k de 1 a lngC`. Pour chaque `k` on construit la liste des k-ièmes caractères de chaque séquence et on en calcule le *consensus-51*.

On concatène alors tous ces *consensus-51* de caractères pour former la séquence consensus.

5.6 - *Programmation d'un consensus-51 de séquences*

```
# calcul du consensus d'un ensemble de séquences
# via le calcul du consensus caractère par caractère
# à la même position dans les séquences

from consensus51 import *

def consensus51seq( listeSeq ) :

  if (len(listeSeq)==0) :
      return(["?",0])

  # longueur commune

  lngC = len(listeSeq[0])

  # initialisation de la séquence consensus

  seqConsensus = ""

  # calcul des consensus caractère par caractère

  for k in range(lngC) :

      listeCar = []

      for seq in listeSeq :
          listeCar.append( seq[k] )
      # fin pour chaque séquence

      carConsensus  = consensus51(listeCar)[0]

      # ajout à la séquence consensus

      seqConsensus += carConsensus

  # fin pour chaque caractère

  return( seqConsensus )

# fin de fonction consensus51seq
```

6. Un peu de R

6.1 - 6.2 *Méthode pour programmer un consensus-51 de caractères et script R*

Puisque R dispose d'une fonction vectorielle *table()* qui calcule les effectifs de chaque valeur, et puisque la fonction *prop.table()* renvoie les effectifs relatifs, il suffit de multiplier par 100, de chercher le maximum et de renvoyer, comme en PYTHON, le vecteur formé de "?" et 0 pour un vecteur vide, "<" et le caractère le plus présent concaténé à son pourcentage s'il est inférieur à 51 ou le vecteur composé du caractère le plus présent et de son pourcentage dans les bons cas. Le code associé est :

```
consensus <- function(vdCar,seuilPct=51) {

   # on sort de suite si c'est la liste vide

   if (length(vdCar)==0) { return( c("?",0) ) }

   # calcul des effectifs puis des pourcentages

   pcts   = 100*prop.table( table(vdCar) )

   # recherche du pourcentage maximal

   indMax    <- which.max(pcts)
   pctMax    <- pcts[indMax]
   carPctMax <- names(pcts)[indMax]
   names(pctMax) <- NULL

   # si le pourcentage max est inférieur au seuil maximal
   # on renvoie "<" sinon, on renvoie le caractère

   if (pctMax < seuilPct) {
       return( c("<",paste(carPctMax,pctMax,sep="")) )
   } else {
       return( c(carPctMax,pctMax) )
   } # fin de si

} # fin de fonction consensus

### résultats

# consensus( c()            )  renvoie  "?"   "0"
# consensus( c("A","C")     )  renvoie  "<"   "A50"
# consensus( c("A","C","A") )  renvoie  "A"   "66.6666666666667"
```

6.3 - 6.4 *Méthode pour programmer un consensus-51 de séquences* et script R

Là encore on peut suivre la méthode indiquée pour PYTHON : on calcule la séquence consensus de proche en proche pour chaque position. Toutefois, ici, au lieu de concaténer les consensus caractères on commencera par créer un vecteur de caractères qu'on remplira au fur et à mesure, et qu'on convertira en chaine en fin de fonction. Remarque : comme la fonction `substr()` de R est vectorielle, on a une boucle POUR explicite en moins à écrire.

```
# calcul du consensus d'un ensemble de séquences
# via le calcul du consensus caractère par caractère
# à la même position dans les séquences

consensus51seq <- function ( vectSeq ) {

  if (length(vectSeq)==0) { return( c() ) }

  # longueur commune

  lngC = nchar(vectSeq[1])

  # initialisation de la séquence consensus

  seqConsensus = rep("",lngC)

  # calcul des consensus caractère par caractère

  for (k in 1:lngC) {

      vdCar = substr(vectSeq,k,k)
      carConsensus  = consensus(vdCar)[1]

      # ajout à la séquence consensus

      seqConsensus[k] = carConsensus

  } # fin pour chaque caractère

  # conversion du vecteur de caractères en une chaine

  seqConsensus = paste(seqConsensus,collapse="")

  return( seqConsensus )

} # fin de fonction consensus51seq
```