Langage R
Mme. HDIOUD Ferdaous
A.U 2018- 2019
2ème Semestre

TP DIRIGE: LES VECTEURS MANIPULATION ET COMPARAISON

Objectifs:
- Analyser un *jeu de données* correspondant à une petite expérience qui sera détaillée ci-dessous.
- Apprendre à harmoniser les données entre les différents vecteurs et à les comparer statistiquement en découvrant les méthodes et fonctions adéquates.

Description de jeu de données :

Nous allons travailler sur quatre vecteurs issus d'une expérience visant à déterminer si la taille d'un individu est liée à sa performance établie lors d'une épreuve de saut en hauteur et si il existe des différences entre filles et garçons tant du point de vue performance. On mesure pour cela la taille de chacun de ces individus et on relève leur meilleure performance lors d'une série de 3 sauts en hauteur.

Durant le TP on va travailler avec le fichier de sauvegarde *Chapitre2.RData* contenant les quatre vecteurs suivants:

- tailleG: vecteur contenant la taille en centimètres de tous les individus de sexe masculin.
- performanceG: vecteur contenant la meilleure de 3 performances (en cm) en saut en hauteur pour les individus de sexe masculin.
- tailleF: vecteur contenant la taille en centimètres de tous les individus de sexe féminin.
- performanceF: vecteur contenant la meilleure de 3 performances (en cm) en saut en hauteur pour les individus de sexe féminin.

1- Sauvegardez donc le fichier dans le dossier *donnees* créé lors du premier TP et importez-le grâce à la fonction *load()*.

```
> # On définit le répertoire de travail
> setwd("/Users/votrelogin/R/")
>
> # On importe l'objet contenant les données
> objets <- load("donnees/Chapitre2.RData")
>
> # On affiche le nom des objets qui ont été importés
> objets
```

NB: A l'aide du chapitre 2, vous pouvez d'ores et déjà jeter un premier coup d'œil aux vecteurs importés et par exemple en déterminer la longueur, en afficher le contenu ou encore calculer les mesures statistiques (médiane, moyenne, variance,...) définissant les distributions numériques contenues dans ces vecteurs.

Opérations Ensemblistes:

2- Afficher le nombre d'observation de chacune des vecteurs.
3- Afficher les noms attribués à chacune des vecteurs (i.e. les indexes par noms).

Nous ne pourrons travailler qu'avec les individus pour lesquels nous disposons à la fois de la taille et de la performance en saut en hauteur. Donc on doit générer de nouveaux vecteurs ne contenant que les données que nous pouvons analyser.

Quel est l'ensemble des noms présents dans l'étude ?

4- La fonction *union()* nous permet de lister les prénoms masculins et féminins pour lesquels nous disposons d'au moins une mesure (soit la taille ou la performance).

```
> union(names(tailleG), names(performanceG))
 [1] "Vincent"   "Pierre"    "Karim"     "Michel"    "Eric"      "Kevin"     "Paul"
"Youssef"
 [9] "Alain"     "Ruy"       "Adrien"    "Bastien"   "Jacques"   "Christian" "Justin"
"Julien"
[17] "Philippe"  "Benoit"    "Thimothee" "Francois"  "Diego"     "Jean"      "Fabien"
"Claude"
[25] "Cedric"    "Matteo"    "David"
> union(names(tailleF), names(performanceF))
 [1] "Katia"       "Mireille"    "Angelique"   "Emilie"      "Christine"   "Estelle"
"Jacqueline"
 [8] "Magali"      "Ana"         "Cheryl"      "Lucie"       "Emmannuelle" "Marion"
"Chloe"
[15] "Zoe"         "Fatima"      "Irene"       "Aurelia"     "Karen"       "Julie"
"Mathilde"
[22] "Juliette"    "Chen"        "Sandrine"    "Delphine"    "Elsa"        "Liza"
"Josette
>
> # On calcule la longueur de ces vecteurs
> length(union(names(tailleG), names(performanceG)))
[1] 27
> length(union(names(tailleF), names(performanceF)))
[1] 28
```

Quels noms sont communs aux deux ensembles ?

5- Nous allons maintenant déterminer l'ensemble de noms pour lesquels nous avons des mesures à la fois de taille et de performance.

```
>communG <- intersect(names(tailleG), names(performanceG))
>communF <- intersect(names(tailleF), names(performanceF))
>
> length(communG)
[1] 20
> length(communF)
[1] 21
```

Quels noms ne sont présents que dans un seul ensemble ?

6- Définir les noms pour lesquels nous n'avons qu'un seul type de mesure.

```
> # Noms présents dans les index de tailleG mais pas performanceG
> setdiff(names(tailleG), names(performanceG))
[1] "Kevin"  "Alain"  "Justin"
> # Et l'inverse!
> setdiff(names(performanceG), names(tailleG))
[1] "Claude" "Cedric" "Matteo" "David"
>
> setdiff(names(tailleF), names(performanceF))
[1] "Jacqueline" "Zoe"
> setdiff(names(performanceF), names(tailleF))
[1] "Sandrine" "Delphine" "Elsa"     "Liza"     "Josette"
```

7- Indiquer quels sont les éléments du vecteur *tailleG* qui sont présents dans le vecteur *performanceG*.

```
> # Noms présents dans les index de tailleG mais pas performanceG
> setdiff(names(tailleG), names(performanceG))
[1] "Kevin"  "Alain"  "Justin"
> # Et l'inverse!
> setdiff(names(performanceG), names(tailleG))
[1] "Claude" "Cedric" "Matteo" "David"
>
> setdiff(names(tailleF), names(performanceF))
[1] "Jacqueline" "Zoe"
> setdiff(names(performanceF), names(tailleF))
[1] "Sandrine" "Delphine" "Elsa"     "Liza"     "Josette"
```

Cette information brute est peu commode à lire, mais peut être très utile car les vecteurs peuvent être indexés grâce à des booléens, en ne retenant que les éléments d'index de valeur TRUE.

8- Créer un vecteur contenant les éléments de *tailleG* associés à un nom aussi présent dans le vecteur *performanceG*.

```
> tailleG[names(tailleG) %in% names(performanceG)]
  Vincent    Pierre     Karim    Michel      Eric      Paul   Youssef       Ruy    Adrien   Bastien
      169       175       182       166       179       165       168       179       167       172
  Jacques Christian    Julien  Philippe    Benoit Thimothee  Francois     Diego      Jean    Fabien
      156       191       183       178       175       190       172       173       185       171
```

Ainsi, le vecteur *tailleG* sera indexé selon le vecteur d'éléments logiques généré grace à l'opérateur *%in%*.

Ordonner les vecteurs:

➢ la réciprocité des données

Pour pouvoir continuer les analyses, il faut créer de nouveaux vecteurs contenant seulement les sujets pour lesquels nous avons toutes les données nécessaires. Grace à la section précédente vous avez déjà des éléments de réponse quant aux fonctions à utiliser. Cependant, il y a un aspect important à prendre en compte avant de commencer: la réciprocité des données **(càd le 1er élément du 1er vecteur est associé au 1er élément du 2ème vecteur et ainsi de suite).** C'est exactement ce que fait le code suivant:

```
> # On créé deux nouveaux vecteurs en utilisant le même vecteur de noms afin d'avoir une réprocité
des données entre les deux vecteurs
> tailleG2 <- tailleG[communG]
> performanceG2 <- performanceG[communG]
>
> # Même chose pour les sujets féminins
> tailleF2 <- tailleF[communF]
> performanceF2 <- performanceF[communF]
>
> tailleG2
  Vincent    Pierre     Karim    Michel      Eric      Paul   Youssef       Ruy    Adrien   Bastien
      169       175       182       166       179       165       168       179       167       172
  Jacques Christian    Julien  Philippe    Benoit Thimothee  Francois     Diego      Jean    Fabien
      156       191       183       178       175       190       172       173       185       171
> performanceG2
  Vincent    Pierre     Karim    Michel      Eric      Paul   Youssef       Ruy    Adrien   Bastien
      139       155       130       125       135       149      160s       142       155       155
  Jacques Christian    Julien  Philippe    Benoit Thimothee  Francois     Diego      Jean    Fabien
      133       168       138       154       144       145       139       138       142       145
>
```

➢ le tri des données

Le même problème revient lorsque l'on doit trier un des deux vecteurs. En en triant un, son ordre sera modifié et la réprocité avec le second vecteur sera perdue.

Comme vu dans le cours, il faut alors préférer la fonction *order()* à la fonction *sort*(). La fonction *order()* renverra effectivement, non pas les valeurs triées des éléments du vecteur, mais leurs index. Cette fonction génèrera alors un vecteur contenant des index qui pourra être utilisé pour trier les deux vecteurs.

Alors pour notre analyse on souhaite maintenant trier les vecteurs tailleG2 et tailleF2 par ordre de taille croissant tout en conservant la réciprocité avec leurs vecteurs associés performanceG2 et performanceF2. Donc, pour créer des vecteurs ordonnés de la même façon il suffit de les indexer avec un même vecteur d'index:

```
tailleG2_tri <- tailleG2[order(tailleG2)]
performanceG2_tri <- performanceG2[order(tailleG2)]

tailleF2_tri <- tailleF2[order(tailleF2)]
performanceF2_tri <- performanceF2[order(tailleF2)]
```