Capsule 5 – Les vecteurs

Présentation & introduction

Bonjour à toutes et à tous! Je vais d'abord me présenter. Je m'appelle Guillaume Dubé et je suis étudiant à la maîtrise en épidémiologie à l'École de santé publique de l'Université de Montréal. Je serai votre formateur pour cette capsule ainsi que les deux suivants qui aborderont les thèmes des vecteurs, des matrices et des tableaux (*data frame*). Pour ce qui est de la structure de mes capsules, j'ai divisé le tout en deux sections, soit la section théorique et la section pratique. Entrons tout de suite dans le vif du sujet.

Section théorique

Qu'est-ce qu'un vecteur ? Selon mon ami Wikipédia, en informatique ou en programmation, un vecteur « désigne un conteneur d'éléments ordonnés et accessibles par des indices, dont la taille est dynamique : elle est mise à jour automatiquement lors d'ajouts ou de suppressions d'éléments. » Nous allons revenir à cette définition durant la section pratique. En d'autres termes, les vecteurs sont l'un des objets de base du langage de programmation R. Chaque vecteur contient une liste de valeur unidimensionnelle (à une dimension). Un vecteur représente donc une ligne de valeur auquel la longueur de ce dernier correspond au nombre de valeur qu'il contient. Un vecteur qui contient quatorze valeurs sera d'une longueur de quatorze. Il existe trois types de vecteurs, ils peuvent autant être des vecteurs numériques, ce qui signifie que les valeurs qu'il inclut sont des nombres réels ou des nombres entiers, des vecteurs caractères, ce qui signifie que les valeurs qu'il inclut sont des chaînes de caractères (du texte) ou des vecteurs logiques, ce qui signifie que les valeurs qu'il inclut sont des « vrai » ou des « faux ».

En résumé, dans le langage de programmation R, une ligne de valeur de longueur non nulle se nomme vecteur. Passons maintenant à la section théorique pour se familiariser avec le concept ainsi qu'avec le langage de programmation.

Section pratique

Tout d'abord, nous allons créer un vecteur avec la formule suivante :

```
x <- c(1,3,5,7,9)
```

Avec cette nouvelle formule, nous allons revisiter la définition précédemment citée de Wikipédia. Premièrement, un vecteur désigne un conteneur d'éléments ordonnés. Ce conteneur d'éléments est donc les différentes valeurs entre les parenthèses, ici 1, 3, 5, 7 et 9. Deuxièmement, Wikipédia nous dit que ce conteneur d'éléments ordonnés est accessible par un indice. Dans notre exemple, « x » est l'indice qui appelle notre vecteur qui contient les valeurs 1, 3, 5, 7, et 9. Nous pouvons donc appeler ce vecteur « le vecteur x ». Troisièmement, la longueur de ce vecteur est de cinq, puisqu'il inclut cinq valeurs. Finalement, puisque les valeurs sont des nombres entiers, le vecteur est de type numérique. Certaines commandes peuvent vérifier cette information en cas de doute. Toutefois, avant d'explorer ces commandes, nous allons retourner à notre formule initiale. Deux symboles n'ont pas encore été abordés. La flèche (composé des signes « < » et « - ») est une formule d'assignation de base. Elle signifie simplement que nous voulons enregistrer la formule suivante dans cet indice. Le symbole = fait la même chose. Pour ce qui est du « c », il s'agit d'une fonction pour combiner des valeurs dans un vecteur ou dans une liste (*Combine Values into a Vector or List*). Nous allons voir plusieurs fonctions tout au long des différentes capsules.

Tel que mentionné précédemment, il est possible d'utiliser une fonction pour connaître la classe d'un vecteur. Il s'agit de la fonction *class*. La fonction *str* permet aussi de vérifier cette information. Il suffit donc d'entrer une des formules suivantes : class(x) ou str(x)

Il est a noté que lorsque nous créons un vecteur avec plusieurs types de valeur, tel que numérique et caractère, R les traitera tous comme des caractères. Par exemple : b <- c("Canada", 3.1, 194), le vecteur b sera de type caractères (utiliser la fonction class(b) pour le confirmer).

Maintenant que nous avons plusieurs vecteurs de créés, il est possible de les fusionné avec la même fonction, soit :

a <- c(x,b)

Cette fonction ordonne selon le rang spécifié les vecteurs un à la suite de l'autre. Le vecteur a sera donc composé des valeurs du vecteur x et ceux du vecteur b.

Il existe maintenant d'autres fonctions utiles à la création de vecteurs. En voici quelques-unes d'entre-elles :

x <- rep(x, y) où x représente une valeur qu'on souhaite avoir y fois. Si nous souhaitons avoir un vecteur x qui contient 100 fois la valeur 8, il suffit d'utiliser la formule : x <- rep(8,100)

x <- seq(x, y, by = z) où x représente le début de la séquence, y la fin de la séquence et z la longueur des bonds pour se rendre à la fin de la séquence. Par exemple, si nous souhaitons avoir un vecteur x qui débute à 10, qui se rend jusqu'à 105 avec des bons de 1,5, il suffit d'utiliser la formule : x <- seq(10,105, by = 1.5). Dans cet exemple, puisqu'il est impossible d'attendre 105 en utilisant des bonds de 1.5, le vecteur se termine à 104,5.

Il est possible de connaître la longueur du vecteur avec la fonction length().

Afin de nous simplifier la tâche, les développeurs du langage R ont insérer quelques fonctions raccourcies, telles que :

letters : ce vecteur contient toutes les lettres de l'alphabet en minuscule. Il suffit d'écrire la fonction en majuscule pour obtenir l'alphabet en majuscule.

month.name : ce vecteur contient les douze mois de l'année. La fonction month.abb nous donne les abréviations des mois.

Base de données

Maintenant que nous avons vu ces différentes fonctions, ouvrons nos jeux de données pour manipuler certains vecteurs. Analysons d'abord les données de la variable sodium de la base de données apacheApsVar. Afin d'obtenir uniquement cette variable, il suffit d'utiliser la fonction suivante : summary(apacheApsVar$sodium)

Avec cette formule, nous avons demandé à R de sortir un sommaire de la variable (ou vecteur) sodium qui se trouve dans la base de données apacheApsVar. Le signe « $ » permet de préciser le lieu de la variable. Lorsque nous observons les valeurs de cette variable, nous pouvons observer que le minimum est de « -1 ». Puisque cette donnée n'a pas de sens, nous allons vouloir la transformer pour indiquer qu'il s'agit à vrai dire d'une donnée manquante (NA).

apacheApsVar$sodium[apacheApsVar$sodium == -1] <- NA

Nous avons ici demandé à R de remplacer toutes les valeurs qui sont égales à -1 dans la variable sodium (se trouvant dans la base de données apacheApsVar) par le symbole « NA » qui signifie « valeur manquante ».

Il est à noter qu'il existe souvent plusieurs méthodes pour arriver à nos fins. Certaines d'entre elles sont plus simples, mais demandent l'ouverture d'un *package*. Vous allez certainement devoir vous en servir tout au long de votre séjour avec R.

Nous pouvons effectuer la fonction summary(apacheApsVar$sodium) pour obtenir la vraie moyenne de notre échantillon.

Maintenant que cette modification a été effectuée, il est aussi possible de connaître le nombre d'observations dans chaque catégorie.

```
table(apacheApsVar$sodium)
```

Avec cette même fonction, nous pouvons déterminer combien de patients ont un taux de sodium inférieur à 135 avec la fonction :

```
table(apacheApsVar$sodium < 135)
```

Nous pouvons donc voir dans la console qu'il y a 96 individus avec un taux de sodium à 135, 160 individus avec un taux de sodium à 139, 4 individus avec un taux de sodium à 154, etc. Nous apercevons dans ce tableau qu'il existe deux catégories avec une décimale. Puisque cette information est trop précise, nous allons vouloir la modifier. Pour ce faire, nous allons utiliser une formule similaire à celle pour les données manquantes.

```
apacheApsVar$sodium[apacheApsVar$sodium == 131.6] <- 132

apacheApsVar$sodium[apacheApsVar$sodium == 135.8] <- 136
```

Lorsque nous effectuons la commande table(apacheApsVar$sodium), nous pouvons voir que ces catégories ont maintenant disparu.

Nous allons maintenant créer un nouveau vecteur à partir de la variable sodium. Nous allons créer un vecteur binaire qui nous indique si le taux de sodium est inférieur à 135. Nous allons appeler cette variable hyponatrémie.

```
apacheApsVar$hyponatremia <- ifelse(apacheApsVar$sodium < 135,1,0)
```

La fonction ifelse() indique une condition, sa nouvelle valeur si elle est respectée ainsi que sa nouvelle valeur si elle n'est pas respectée.

Avec la fonction table(), nous pouvons vérifier si nous avons le même résultat que nous avons eu plus tôt.

```
table(apacheApsVar$hyponatremia)
```

Nous allons maintenant modifier le nom des catégories avec la fonction levels

```
levels(apacheApsVar$hyponatremia) <-c("No", "Yes")
```

Cependant, si nous voulons créer un vecteur avec trois catégories, nous pouvons utiliser deux techniques :

```
apacheApsVar$sodiumcat <- ifelse(apacheApsVar$sodium < 135, 0,
ifelse(apacheApsVar$sodium > 145,2, 1))
```

```
levels(apacheApsVar$sodiumcat) <- c("Hyponatremia", "Normal", "Hypernatremia")
```

ou la technique

```
apacheApsVar$sodiumcat[apacheApsVar$sodium < 135] <- "Hyponatremia"
```

```
apacheApsVar$sodiumcat[apacheApsVar$sodium > 145] <- "Hypernatremia"
```

```
apacheApsVar$sodiumcat[apacheApsVar$sodium >= 135 & apacheApsVar$sodium <= 145] <-
"Normal"
```

Les deux techniques se valent, c’est une question de préférence.

Voilà tout pour cette capsule. On se retrouve dans la prochaine où nous allons aborder les matrices!