
Notes de leçon | Structures de données

GRAPH Network & OMS, soutenu par le Fonds Mondial

January 2024

Ce cours a été créé par le Réseau GRAPH, une organisation à but non lucratif basée à l'Institut de santé globale de l'Université de Genève, en collaboration avec l'Organisation mondiale de la Santé, dans le cadre d'une subvention du Fonds mondial pour créer des cours afin de renforcer les capacités nationales en matière d'analyse épidémiologique.

Intro	
Objectifs d'apprentissage	
Packages	
Introduction aux vecteurs	
Manipulation de vecteurs	
Des vecteurs aux data frames	
Tibbles	
read_csv() crée des tibbles	
Conclusion	
Solutions	

Intro

Dans cette leçon, nous allons examiner les structures de données dans R. Nous commencerons par étudier les vecteurs, qui sont la structure de données fondamentale dans R. Puis, nous apprendrons à combiner des vecteurs dans des data frames, qui sont la structure la plus utilisée pour organiser et analyser les données. Alors, commençons !

Objectifs d'apprentissage

1. Vous pouvez créer des vecteurs avec la fonction `c()`.
2. Vous pouvez combiner des vecteurs dans des data frames.
3. Vous comprenez la différence entre un tibble et un data frame.

Packages

Veuillez charger les packages nécessaires pour cette leçon avec le code ci-dessous :

```
if(!require(pacman)) install.packages("pacman")
pacman::p_load(tidyverse)
```

Introduction aux vecteurs

Les vecteurs sont les structures de données les plus fondamentales dans R. Ils sont une collection de valeurs qui partagent toutes la même classe (par exemple, toutes numériques ou toutes caractères).

Les vecteurs peuvent être créés en utilisant la fonction `c()` :

On écrit :

```
c(1, 2, 3)
```

```
## [1] 1 2 3
```

Nous pouvons assigner ce vecteur à un objet appelé `mon_vec` de cette manière :

```
mon_vec <- c(1, 2, 3)
```

Définissons maintenant un ensemble de quatre vecteurs différents pour trois personnes :

```
age <- c(18, 25, 46) # age est égal à dix-huit, vingt-cinq et quarante-six  
sexe <- c('H', 'F', 'F') # sexe est égal à c, "H" pour homme, "F" pour femme,  
"F" pour femme  
test_positif <- c(T, T, F) # test_positif est égal à c T, qui représente TRUE,  
T, F, qui représente FALSE  
id <- 1:3 # et enfin, id
```

On peut également vérifier les classes de ces vecteurs :

```
class(age)
```

```
## [1] "numeric"
```

```
class(sexe)
```

```
## [1] "character"
```

```
class(test_positif)
```

```
## [1] "logical"
```

```
class(id)
```

```
## [1] "integer"
```

Chaque ligne de code ci-dessous tente de définir un vecteur avec trois éléments mais contient une erreur. Corrigez les erreurs et effectuez les commandes.

```
mon_vec_1 <- (1, 2, 3)
mon_vec_2 <- c("Obi", "Chika" "Nonso")
```

Manipulation de vecteurs

La plupart des fonctions et opérations que vous connaissez déjà peuvent être appliquées à des vecteurs.

Par exemple, considérons le vecteur `age`, nous pouvons multiplier `age` par 2 :

```
age
```

```
## [1] 18 25 46
```

```
age * 2
```

```
## [1] 36 50 92
```

Remarquez que chaque élément du vecteur a été multiplié par 2.

Ou, ci-dessous nous prenons la racine carrée de `age` :

```
age
```

```
## [1] 18 25 46
```

```
sqrt(age)
```

```
## [1] 4.242641 5.000000 6.782330
```

Vous pouvez également additionner des vecteurs :

Par exemple, si nous voulons ajouter le vecteur `age` au vecteur `id`.

Regardons d'abord le vecteur `age` :

```
age
```

```
## [1] 18 25 46
```

et id :

```
id
```

```
## [1] 1 2 3
```

Nous pouvons simplement les additionner de cette manière :

```
age + id
```

```
## [1] 19 27 49
```

Notez que le premier élément de `age` est ajouté au premier élément de `id` et le deuxième élément de `age` est ajouté au deuxième élément de `id`, et ainsi de suite.

Des vecteurs aux data frames

Maintenant que nous maîtrisons la création de vecteurs, passons à l'objet le plus utilisé dans l'analyse de données : les data frames

Un data frame est une collection de vecteurs de même longueur. Nous pouvons créer un data frame en utilisant la fonction `data.frame()`.

Rappelez-vous que nous avons précédemment créé trois objets vecteurs, (`id`, `age`, `sexe` et `test_positif`) pour trois individus :

Nous pouvons maintenant utiliser la fonction `data.frame()` pour les combiner dans une seule data frame :

```
donnees_epi <- data.frame(id, age, sexe, test_positif)
donnees_epi
```

```
##   id age sexe test_positif
## 1  1  18   H         TRUE
## 2  2  25   F         TRUE
## 3  3  46   F        FALSE
```

Au lieu de créer chaque vecteur séparément, vous pouvez créer votre data frame en définissant chacun des vecteurs à l'intérieur de la fonction `data.frame()`.

```
donnees_epi_2 <- data.frame(age = c(18, 25, 46),
                             sexe = c('M', 'F', 'F'))
```

```
donnees_epi_2
```

```
##   age sexe
## 1  18    M
## 2  25    F
## 3  46    F
```

SIDE NOTE

La plupart du temps lorsque vous travaillez avec des données dans R, vous les importerez à partir de contextes externes. Mais il est parfois utile de créer des jeux de données *dans* R lui-même. C'est dans de tels cas que la fonction `data.frame()` sera utile.

Pour extraire les vecteurs du data frame, utilisez la syntaxe `$`, appelée le signe dollar. Exécutez les lignes de code suivantes dans votre console pour observer cela.

```
donnees_epi$age
is.vector(donnees_epi$age) # vérifie que cette colonne est bien un vecteur
class(donnees_epi$age) # vérifie la classe du vecteur
```

Combinez les vecteurs ci-dessous dans un data frame, avec les noms de colonnes suivants : "nom" pour le vecteur de caractères, "nb_enfants" pour le vecteur numérique et "est_marie" pour le vecteur logique.

```
vecteur_caracteres <- c("Bob", "Jane", "Joe")
vecteur_numerique <- c(1, 2, 3)
vecteur_logique <- c(T, F, F)
```

Utilisez la fonction `data.frame()` pour définir dans R un data frame qui ressemble au tableau suivant :

salle	nb_fenetres
cuisine	2
chambre	5

Tibbles

La version par défaut des données tabulaires dans R s'appelle un data frame, mais il existe une autre représentation des données tabulaires fournie par le package *tidyverse*. Elle s'appelle un *tibble*, et c'est une version améliorée du data frame.

Vous pouvez convertir un data frame en tibble avec la fonction `as_tibble()` :

```
donnees_epi
```

```
##   id age sexe test_positif
## 1  1  18   H      TRUE
## 2  2  25   F      TRUE
## 3  3  46   F     FALSE
```

```
tibble_epi <- as_tibble(donnees_epi)
tibble_epi
```

```
## # A tibble: 3 × 4
##   id   age sexe test_positif
##   <int> <dbl> <chr> <lgl>
## 1     1    18   H      TRUE
## 2     2    25   F      TRUE
## 3     3    46   F     FALSE
```

Remarquez que le tibble affiche les dimensions des données dans la première ligne :

```
# A tibble: 3 × 4
#   id   age sexe test_positif
#   <int> <dbl> <chr> <lgl>
# 1     1    18   M     VRAI
# 2     2    25   F     VRAI
# 3     3    46   F     FAUX
```

Et indique également les types de données en haut de chaque colonne.

Ici, "int" signifie "entier" ("integer"), "dbl" signifie "double" (qui est un type numérique), "chr" signifie "caractère" et "lgl" signifie "logique".

L'autre avantage des tibbles est qu'ils évitent d'inonder votre console lorsque vous imprimez un grand tableau.

Considérez, par exemple, ce que vous voyez dans votre console lorsque vous exécutez les lignes ci-dessous :


```
# imprime le data frame infert (un jeu de données R intégré)
infert # Impression très longue
as_tibble(infert) # impression plus gérable
```

Pour la plupart de vos besoins d'analyse de données, vous devriez préférer les tibbles aux data frames classiques.

`read_csv()` crée des tibbles

Lorsque vous importez des données avec la fonction `read_csv()` de {readr}, vous obtenez un tibble :

```
ebola_tib <- read_csv("https://tinyurl.com/ebola-data-sample") # Nécessite
internet pour fonctionner
class(ebola_tib)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"        "data.frame"
```

Mais lorsque vous importez des données avec la fonction `read.csv()` de base, vous obtenez un `data.frame` :

```
ebola_df <- read.csv("https://tinyurl.com/ebola-data-sample") # Nécessite
internet pour fonctionner
class(ebola_df)
```

```
## [1] "data.frame"
```

Essayez d'imprimer `ebola_tib` et `ebola_df` dans votre console pour observer le comportement d'impression différent des tibbles et des data frames.

C'est une des raisons pour lesquelles nous recommandons d'utiliser `read_csv()` plutôt que `read.csv()`.

Conclusion

Avec votre compréhension des classes et des structures de données, vous êtes désormais bien équipé pour effectuer des tâches de manipulation de données dans R. Dans les prochaines leçons, nous explorerons les puissantes capacités de transformation de données du package dplyr, qui amélioreront encore vos compétences en analyse de données.

Félicitations d'être arrivé jusque-là ! Vous avez beaucoup appris et vous pouvez être fiers de vous.

Solutions

Solution au premier bloc r-practice :

```
mon_vec_1 <- c(1,2,3) # Utilisez la fonction 'c' pour créer un vecteur
mon_vec_2 <- c("Obi", "Chika", "Nonso") # Séparez chaque chaîne par une virgule
```

Solution au deuxième bloc r-practice :

```
df <- data.frame(nom = vecteur_caracteres,
                 nb_enfants = vecteur_numerique,
                 est_marie = vecteur_logique)
```

Solution au troisième bloc r-practice :

```
# Solution au troisième bloc r-practice
pieces <- data.frame(piece = c("salle à manger", "cuisine", "chambre"),
                    nb_fenetres = c(3, 2, 5))
```

Contributeurs

L'équipe suivante a contribué à cette leçon :



DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom



EDUARDO ARAUJO

Student at Universidade Tecnológica Federal do Parana
Passionate about reproducible science and education



LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education



KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about world improvement

Références

Certains éléments de cette leçon ont été adaptés des sources suivantes :

- Wickham, H., & Grolemund, G. (s.d.). *R for data science*. 15 Factors | R for Data Science. Consulté le 26 octobre 2022. <https://r4ds.had.co.nz/factors.html>.

This work is licensed under the [Creative Commons Attribution Share Alike](#) license.

