Notes de leçon | Filtrer les lignes

January 2024

Introduction

Poursuivons avec le package {dplyr}, en découvrant le verbe `filter`. La dernière fois, nous avons vu comment `select` selectionne des variables (colonnes) et aujourd'hui nous verrons comment conserver ou supprimer des entrées de données, des lignes, en utilisant `filter`. Supprimer des entrées de données anormales ou conserver des sous-ensembles de vos points de données est un autre aspect essentiel de la manipulation des données.

Allons-y !

Objectifs d'apprentissage

1. Vous pouvez utiliser `dplyr::filter()` pour conserver ou supprimer des lignes d'un dataframe.
2. Vous pouvez filtrer les lignes en spécifiant des conditions sur des nombres ou des chaînes en utilisant des opérateurs relationnels comme supérieur à (`>`), inférieur à (`<`), égal à (`==`), et différent de (`!=`).
3. Vous pouvez filtrer les lignes en combinant des conditions avec des opérateurs logiques comme le esperluette (`&`) et la barre verticale (`|`).

4. Vous pouvez filtrer les lignes en négatif des conditions avec l'opérateur logique point d'exclamation (`!`).
5. Vous pouvez filtrer les lignes avec des valeurs manquantes en utilisant la fonction `is.na()`.

L'ensemble de données COVID-19 de Yaoundé

Dans cette leçon, nous utiliserons à nouveau les données de l'enquête sérologique COVID-19 réalisée à Yaoundé, au Cameroun.

```
<- read_csv(here::here('data/yaounde_data.csv'))
us-ensemble plus petit de variables
aounde %>%
(age, sex, weight_kg, highest_education, neighborhood,
 occupation, is_smoker, is_pregnant,
 igg_result, igm_result)
```

```
## # A tibble: 5 × 10
##     age sex    weight_kg highest_education neighborhood
##   <dbl> <chr>      <dbl> <chr>             <chr>
## 1    45 Female        95 Secondary         Briqueterie
## 2    55 Male          96 University        Briqueterie
## 3    23 Male          74 University        Briqueterie
## 4    20 Female        70 Secondary         Briqueterie
## 5    55 Female        67 Primary           Briqueterie
## # ℹ 5 more variables: occupation <chr>, is_smoker <chr>,
## #   is_pregnant <chr>, igg_result <chr>, igm_result <chr>
```

Notez que le jeu de donnée COVID-19 Yaoundé est en anglais !

Pour cette leçon, nous utiliserons cette version en anglais. Mais dans d'autres leçons, nous utiliserons une version partialement en français.

Introduction à `filter()`

Nous utilisons `filter()` pour conserver les lignes qui satisfont à un ensemble de conditions. Prenons un exemple simple. Si nous voulons conserver uniquement les enregistrements masculins, nous exécutons :

```
filter(sex == "Male")
```

```
## # A tibble: 5 × 10
##     age sex   weight_kg highest_education neighborhood
##   <dbl> <chr>     <dbl> <chr>             <chr>
## 1    55 Male         96 University        Briqueterie
## 2    23 Male         74 University        Briqueterie
## 3    28 Male         62 Doctorate         Briqueterie
## 4    30 Male         73 Secondary         Briqueterie
## 5    42 Male         71 Secondary         Briqueterie
## # ℹ 5 more variables: occupation <chr>, is_smoker <chr>,
## #   is_pregnant <chr>, igg_result <chr>, igm_result <chr>
```

Notez l'utilisation du double signe égal `==` plutôt que le signe égal simple `=`. Le signe `==` teste l'égalité, comme le montre l'exemple ci-dessous :

```
l'objet `sex_vector` avec trois éléments
tor <- c("Male", "Female", "Female")
r quels éléments sont égaux à "Male"
tor == "Male"
```

```
## [1]  TRUE FALSE FALSE
```

Donc le code `yao %>% filter(sex == "Male")` conservera toutes les lignes où le test d'égalité `sex == "Male"` évalue à `TRUE`.

Il est souvent utile de chaîner `filter()` avec `nrow()` pour obtenir le nombre de lignes remplissant une condition.

```
en de répondants étaient des hommes?

r(sex == "Male") %>%
```

```
## [1] 422
```

KEY POINT

Le double signe égal, `==`, teste l'égalité, tandis que le signe égal simple, `=`, est utilisé pour spécifier des valeurs aux arguments à l'intérieur des fonctions.

PRACTICE

Filtrez le dataframe `yao` pour sélectionner les répondants qui étaient enceintes lors de l'enquête.

Pratique 1_2

Combien de répondants étaient des femmes? (Utilisez `filter()` et `nrow()`)

Opérateurs relationnels

L'opérateur `==` introduit ci-dessus est un exemple d'un opérateur "relationnel", car il teste la relation entre deux valeurs. Voici une liste de certains de ces opérateurs :

Opérateur	est VRAI si
A < B	A est **inférieur à** B
A <= B	A est **inférieur ou égal à** B
A > B	A est **supérieur à** B
A >= B	A est **supérieur ou égal à** B
A == B	A est **égal à** B
A != B	A est **différent de** B
A %in% B	A **est un élément de** B

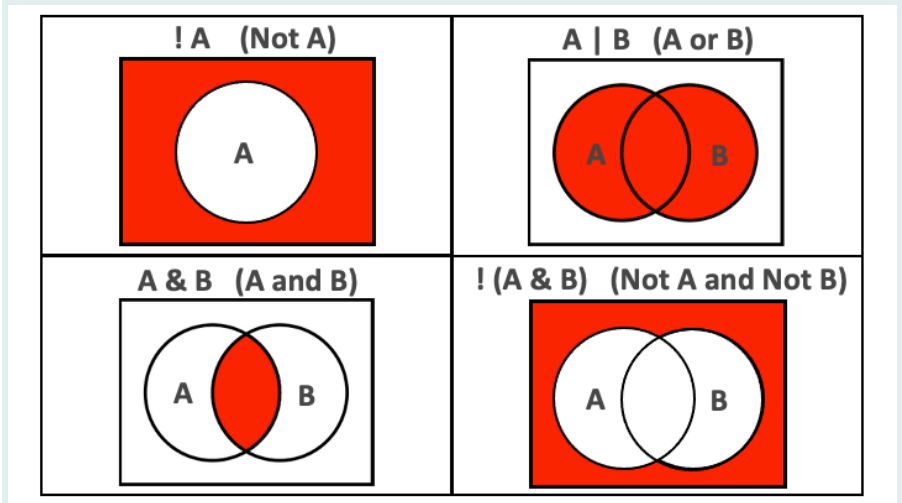

Fig: Opérateurs AND et OR visualisés.

Voyons comment les utiliser avec `filter()` :

```
filter(_____________) ## gardez les lignes où `sex` n'est pas "Male"
filter(_____________) ## gardez les répondants de moins de 6 ans
filter(_____________) ## gardez les répondants âgés d'au moins 70 ans

# les répondants dont le niveau d'éducation le plus élevé est "Primary" ou
 "Secondary"
filter(_____________________________________)
```

Pratique 2_1

De `yao`, conservez uniquement les répondants qui étaient des enfants (moins de 18 ans).

Pratique 2_2

Avec `%in%`, conservez uniquement les répondants qui vivent dans les quartiers “Tsinga” ou “Messa”.

Combiner des conditions avec `&` et `|`

Nous pouvons passer plusieurs conditions à une seule instruction `filter()` séparées par des virgules:

```
# trouver les répondants qui sont enceintes et qui sont d'anciens fumeurs
filter(is_pregnant == "Yes", is_smoker == "Ex-smoker") ## seulement une ligne
```

```
## # A tibble: 1 × 10
##     age sex    weight_kg highest_education neighborhood
##   <dbl> <chr>      <dbl> <chr>             <chr>
## 1    25 Female        90 Secondary         Carriere
## # ℹ 5 more variables: occupation <chr>, is_smoker <chr>,
## #   is_pregnant <chr>, igg_result <chr>, igm_result <chr>
```

Lorsque plusieurs conditions sont séparées par une virgule, elles sont implicitement combinées avec un **et** (`&`).

Il est préférable de remplacer la virgule par `&` pour rendre cela plus explicite.

```
# même résultat qu'auparavant, mais `&` est plus explicite
filter(is_pregnant == "Yes" & is_smoker == "Ex-smoker")
```

```
## # A tibble: 1 × 10
##     age sex    weight_kg highest_education neighborhood
##   <dbl> <chr>      <dbl> <chr>             <chr>
## 1    25 Female        90 Secondary         Carriere
## # ℹ 5 more variables: occupation <chr>, is_smoker <chr>,
## #   is_pregnant <chr>, igg_result <chr>, igm_result <chr>
```

Ne confondez pas :

- la "," pour lister plusieurs conditions dans filter `filter(A,B)` c'est-à-dire filtrer sur la condition A et (`&`) la condition B
- la "," dans les listes `c(A,B)` qui énumère différents composants de la liste (et n'a rien à voir avec l'opérateur `&`)

Si nous voulons combiner des conditions avec un **ou**, nous utilisons le symbole de la barre verticale, `|`.

```
dants qui sont enceintes OU qui sont d'anciens fumeurs
filter(is_pregnant == "Yes" | is_smoker == "Ex-smoker")
```

```
## # A tibble: 5 × 10
##     age sex    weight_kg highest_education neighborhood
##   <dbl> <chr>      <dbl> <chr>             <chr>
## 1    55 Male          96 University        Briqueterie
## 2    42 Male          71 Secondary         Briqueterie
## 3    38 Male          71 University        Briqueterie
## 4    69 Male         108 University        Briqueterie
## 5    65 Male          93 Secondary         Briqueterie
## # ℹ 5 more variables: occupation <chr>, is_smoker <chr>,
## #   is_pregnant <chr>, igg_result <chr>, igm_result <chr>
```

Pratique 3_1

Filtrez `yao` pour ne conserver que les hommes qui ont été testés positifs pour les IgG.

Pratique 3_2

PRACTICE

(in RMD)

Filtrez `yao` pour conserver à la fois les enfants (moins de 18 ans) et toute personne dont le niveau d'éducation le plus élevé est l'école primaire.

Négation des conditions avec `!`

Pour nier les conditions, nous les enveloppons dans `!()`.

Ci-dessous, nous supprimons les répondants qui sont des enfants (moins de 18 ans) ou qui pèsent moins de 30 kg :

```
imer les répondants < 18 ans OU < 30 kg
filter(!(age < 18 | weight_kg < 30))
```

```
## # A tibble: 5 × 10
##     age sex    weight_kg highest_education neighborhood
##   <dbl> <chr>      <dbl> <chr>             <chr>
## 1    45 Female        95 Secondary         Briqueterie
## 2    55 Male          96 University        Briqueterie
## 3    23 Male          74 University        Briqueterie
## 4    20 Female        70 Secondary         Briqueterie
## 5    55 Female        67 Primary           Briqueterie
## # ℹ 5 more variables: occupation <chr>, is_smoker <chr>,
## #   is_pregnant <chr>, igg_result <chr>, igm_result <chr>
```

L'opérateur `!` est également utilisé pour nier `%in%` car R n'a pas d'opérateur pour **NOT in**.

```
imer les répondants dont le niveau d'éducation le plus élevé n'est PAS
 "Primaire" ou "Secondaire"
filter(!(highest_education %in% c("Primary", "Secondary")))
```

```
## # A tibble: 5 × 10
##     age sex   weight_kg highest_education neighborhood
##   <dbl> <chr>     <dbl> <chr>             <chr>
## 1    55 Male         96 University        Briqueterie
## 2    23 Male         74 University        Briqueterie
## 3    28 Male         62 Doctorate         Briqueterie
## 4    38 Male         71 University        Briqueterie
## 5    54 Male         71 University        Briqueterie
## # ℹ 5 more variables: occupation <chr>, is_smoker <chr>,
## #   is_pregnant <chr>, igg_result <chr>, igm_result <chr>
```

KEY POINT

Il est plus facile de lire les instructions `filter()` comme des instructions **conserver**, pour éviter toute confusion sur le fait de savoir si nous filtrons **dans** ou filtrons **hors**!

Ainsi, le code ci-dessous se lirait : "**conserver** les répondants qui ont moins de 18 ans ou qui pèsent moins de 30 kg".

KEY POINT

```
filter(age < 18 | weight_kg < 30)
```

Et lorsque nous enveloppons des conditions dans `!()`, nous pouvons alors lire les instructions `filter()` comme des instructions **supprimer**.

Ainsi, le code ci-dessous se lirait : "**supprimer** les répondants qui ont moins de 18 ans ou qui pèsent moins de 30 kg".

```
filter(!(age < 18 | weight_kg < 30))
```

PRACTICE

(in RMD)

Pratique 4

Dans `yao`, supprimez les répondants qui vivent dans les quartiers Tsinga ou Messa.

Valeurs `NA`

Les opérateurs relationnels introduits jusqu'à présent ne fonctionnent pas avec `NA`.

Créons un sous-ensemble de données pour illustrer cela.

```
 <- yao %>%
(sex, is_pregnant) %>%
(1,11,50,2) # ordre de ligne personnalisé
```

```
## # A tibble: 4 × 2
##   sex    is_pregnant
##   <chr>  <chr>
## 1 Female No
## 2 Female No response
```

```
## 3 Female Yes
## 4 Male   <NA>
```

Dans `yao_mini`, le dernier répondant a une valeur `NA` pour la colonne `is_pregnant` car il est un homme.

Essayer de sélectionner cette ligne en utilisant `== NA` ne fonctionnera pas.

```
 %>% filter(is_pregnant == NA) # ne fonctionne pas
```

```
## # A tibble: 0 × 2
## # ℹ 2 variables: sex <chr>, is_pregnant <chr>
```

```
 %>% filter(is_pregnant == "NA") # ne fonctionne pas
```

```
## # A tibble: 0 × 2
## # ℹ 2 variables: sex <chr>, is_pregnant <chr>
```

C'est parce que `NA` est une valeur inexistante. Ainsi, R ne peut pas évaluer si elle est "équivalente à" ou "différente de" quoi que ce soit.

La fonction spéciale `is.na()` est donc nécessaire :

```
les lignes où `is_pregnant` est NA
 %>% filter(is.na(is_pregnant))
```

```
## # A tibble: 1 × 2
##   sex   is_pregnant
##   <chr> <chr>
## 1 Male  <NA>
```

Cette fonction peut être niée avec `!` :

```
me les lignes où `is_pregnant` est NA
 %>% filter(!is.na(is_pregnant))
```

```
## # A tibble: 3 × 2
##   sex    is_pregnant
##   <chr>  <chr>
## 1 Female No
## 2 Female No response
## 3 Female Yes
```

SIDE NOTE

SIDE NOTE

Pour les tibbles, RStudio mettra en évidence les valeurs `NA` en rouge vif pour les distinguer des autres valeurs :

```
# A tibble: 5 × 3
    age sex    is_pregnant
  <dbl> <chr>  <chr>
1    32 Male   NA
2    23 Female Yes
3    35 Male   NA
4    31 Female No
5    17 Female No response
```

Une erreur courante avec `NA`

SIDE NOTE

Les valeurs `NA` peuvent être identifiées, mais toute autre codification telle que `"NA"` ou `"NaN"`, qui sont codées comme des chaînes, sera imperceptible pour les fonctions (ce sont des chaînes, comme toutes les autres).

PRACTICE (in RMD)

Pratique 5

Dans l'ensemble de données `yao`, conservez tous les répondants qui avaient des dossiers manquants concernant le signalement de leur statut de fumeur.

PRACTICE (in RMD)

Pratique 6

Pour certains répondants, la fréquence respiratoire, en respirations par minute, a été enregistrée dans la colonne `respiration_frequency`.

Dans `yaounde`, éliminez ceux ayant une fréquence respiratoire inférieure à 20. Pensez aux NAs lors de cette opération ! Vous devriez également éviter de supprimer les valeurs NA.

En Résumé !

Maintenant, vous connaissez les deux verbes essentiels pour sélectionner (`select()`) les colonnes et filtrer (`filter()`) les lignes. De cette manière, vous conservez les variables qui vous intéressent en sélectionnant vos colonnes et vous conservez les entrées de données que vous jugez pertinentes en filtrant vos lignes.

Mais qu'en est-il de la modification, de la transformation de vos données ? Nous en apprendrons davantage à ce sujet dans la prochaine leçon. À bientôt !

Fig: Manipulation de données de base : `select()` et `filter()`.

Contributeurs

Les membres de l'équipe suivants ont contribué à cette leçon :

LAURE VANCAUWENBERGHE

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

ANDREE VALLE CAMPOS

R Developer and Instructor, the GRAPH Network
Motivated by reproducible science and education

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about world improvement

SABINA RODRIGUEZ VELÁSQUEZ
Project Manager and Scientific Collaborator, The GRAPH Network
Infectiously enthusiastic about microbes and Global Health

Références

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

- Horst, A. (2021). *Dplyr-learnr*. https://github.com/allisonhorst/dplyr-learnr (Œuvre originale publiée en 2020)
- *Sélectionner des lignes en utilisant les valeurs des colonnes—Filter*. (n.d.). Consulté le 12 janvier 2022, à partir de https://dplyr.tidyverse.org/reference/filter.html

Les œuvres d'art ont été adaptées de :

- Horst, A. (2021). *Illustrations R & stats par Allison Horst*. https://github.com/allisonhorst/stats-illustrations (Œuvre originale publiée en 2018)

Solutions des exercices pratiques

Solution exercice pratique 1_2

```
lter(sex == 'Female') %>%
ow()
```

Solution exercice pratique 2_1

```
filter(age < 18)
```

Solution exercice pratique 2_2

```
(neighborhood %in% c("Tsinga", "Messa"))
```

Solution exercice pratique 3_1

```
(sex == "Male" & igg_result == "Positive")
```

Solution exercice pratique 3_2

```
filter(age < 18 | highest_education == "Primary")
```

Solution exercice pratique 4

```
r(!(neighborhood %in% c("Tsinga", "Messa")))
```

Solution exercice pratique 5

```
filter(is.na(is_smoker))
```

Solution exercice pratique 6

```
%>% filter(respiration_frequency >= 20 | is.na(respiration_frequency))
```