
Notes de cours | Sélectionner et renommer les colonnes

February 2024

Introduction	
Objectifs d'apprentissage	
L'ensemble de données COVID-19 de Yaoundé	
Introduction à <code>select()</code>	
Sélectionner des gammes de colonnes avec :	
Exclure des colonnes avec !	
Fonctions auxiliaires pour <code>select()</code>	
<code>starts_with()</code> et <code>ends_with()</code>	
<code>contains()</code>	
<code>everything()</code>	
Changer les noms de colonnes avec <code>rename()</code>	
Renommer dans <code>select()</code>	
Conclusion !	

Introduction

Aujourd'hui, nous commencerons notre exploration du package {dplyr} ! Notre premier verbe sur la liste est `select` qui permet de conserver ou de supprimer des variables de votre dataframe. Choisir vos variables est la première étape du nettoyage de vos données.

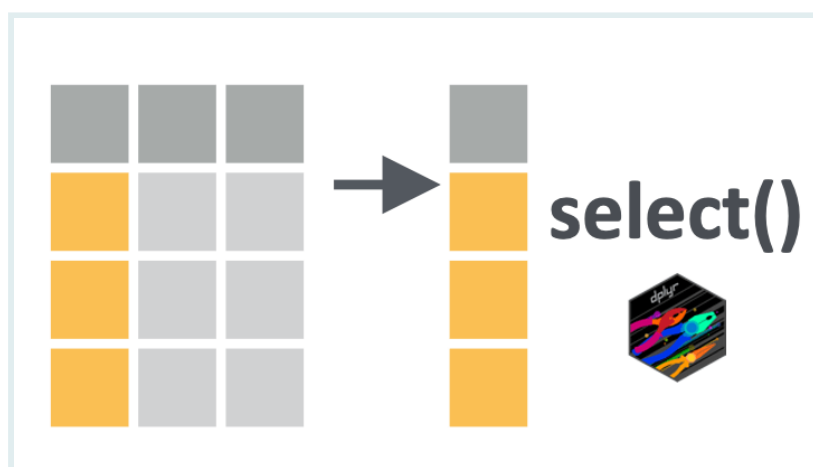


Fig: la fonction `select()`.

C'est parti !

Objectifs d'apprentissage

- Vous pouvez conserver ou supprimer des colonnes d'un dataframe en utilisant la fonction `dplyr::select()` du package {dplyr}.
- Vous pouvez sélectionner une plage ou une combinaison de colonnes à l'aide d'opérateurs comme le deux-points (:), le point d'exclamation (!) et la fonction

`c()`.

- Vous pouvez sélectionner des colonnes en fonction des motifs de leurs noms avec des fonctions d'aide comme `starts_with()`, `ends_with()`, `contains()` et `everything()`.
- Vous pouvez utiliser `rename()` et `select()` pour changer les noms des colonnes.

L'ensemble de données COVID-19 de Yaoundé

Dans cette leçon, nous analysons les résultats d'une enquête sérologique sur la COVID-19 menée à Yaoundé, au Cameroun, fin 2020. L'enquête estimait combien de personnes avaient été infectées par la COVID-19 dans la région, en testant les anticorps IgG et IgM. L'ensemble complet des données peut être obtenu sur [Zenodo](#), et l'article peut être consulté [ici](#).

Passez un peu de temps à parcourir cet ensemble de données. Chaque ligne correspond à un patient interrogé. Il y a des variables démographiques, socio-économiques et liées à la COVID. Les résultats des tests d'anticorps IgG et IgM sont dans les colonnes `igg_result` et `igm_result`.

```
<- read_csv(here::here("data/yaounde_data.csv"))
```

```
## # A tibble: 5 × 53
##   id                date_surveyed   age
##   <chr>             <date>         <dbl>
## 1 BRIQUETERIE_000_0001 2020-10-22      45
## 2 BRIQUETERIE_000_0002 2020-10-24      55
## 3 BRIQUETERIE_000_0003 2020-10-24      23
## 4 BRIQUETERIE_002_0001 2020-10-22      20
## 5 BRIQUETERIE_002_0002 2020-10-22      55
##   age_category age_category_3 sex
##   <chr>        <chr>        <chr>
## 1 45 - 64      Adult        Female
## 2 45 - 64      Adult        Male
## 3 15 - 29      Adult        Male
## 4 15 - 29      Adult        Female
## 5 45 - 64      Adult        Female
## # i 47 more variables: highest_education <chr>,
## #   occupation <chr>, weight_kg <dbl>, height_cm <dbl>, ...
```

SIDE NOTE



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

SIDE NOTE



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



Gauche : l'équipe d'enquête de Yaoundé. Droite : un test d'anticorps étant administré.

Introduction à `select()`

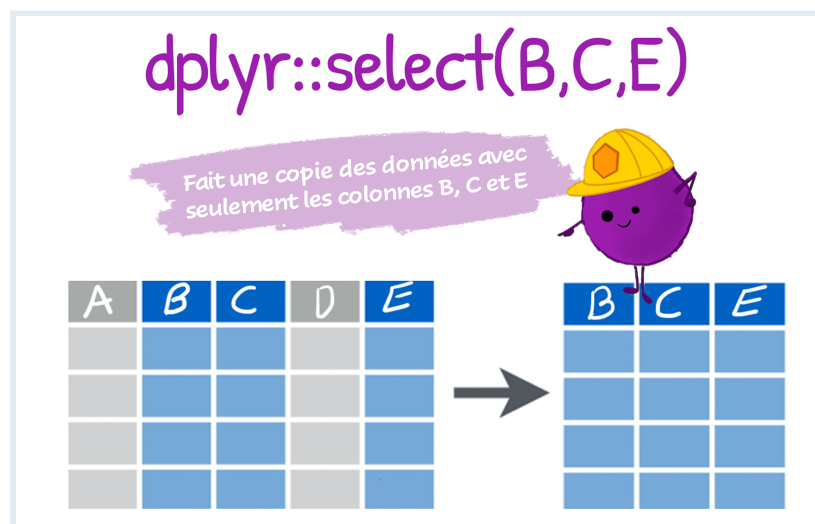


Fig : la fonction `select()`. (Dessin adapté d'Allison Horst).

`dplyr::select()` nous permet de choisir quelles colonnes (variables) conserver ou supprimer.

Nous pouvons sélectionner une colonne **par son nom** :

```
%>% select (age)
```

```
## # A tibble: 5 × 1
##   age
##   <dbl>
## 1    45
## 2    55
## 3    23
## 4    20
## 5    55
```

Ou nous pouvons sélectionner une colonne **par sa position** :

```
%>% select(3) # `age` est la 3ème colonne
```

```
## # A tibble: 5 × 1
##   age
##   <dbl>
## 1    45
## 2    55
## 3    23
## 4    20
## 5    55
```

Pour sélectionner **plusieurs variables**, nous les séparons par des virgules :

```
%>% select (age, sex, igg_result)
```

```
## # A tibble: 5 × 3
##   age sex    igg_result
##   <dbl> <chr>   <chr>
## 1    45 Female Negative
## 2    55 Male   Positive
## 3    23 Male   Negative
## 4    20 Female Positive
## 5    55 Female Positive
```

PRACTICE



(in RMD)

- Sélectionnez les variables poids et taille dans le dataframe `yaounde`.
- Sélectionnez les colonnes 16 et 22 dans le dataframe `yaounde`.

Pour la prochaine partie du tutoriel, créons un sous-ensemble plus petit des données, appelé `yao`.

```
le %>% select(age,
              sex,
              highest_education,
              occupation,
              is_smoker,
              is_pregnant,
              igg_result,
              igm_result)
```

```
## # A tibble: 5 × 8
##   age sex    highest_education
##   <dbl> <chr>  <chr>
## 1    45 Female Secondary
## 2    55 Male   University
## 3    23 Male   University
## 4    20 Female Secondary
## 5    55 Female Primary
##   occupation    is_smoker is_pregnant
##   <chr>        <chr>    <chr>
## 1 Informal worker Non-smoker No
## 2 Salaried worker Ex-smoker <NA>
## 3 Student          Smoker   <NA>
## 4 Student          Non-smoker No
## 5 Trader--Farmer   Non-smoker No
## # i 2 more variables: igg_result <chr>, igm_result <chr>
```

Sélectionner des gammes de colonnes avec :

L'opérateur `:` permet de sélectionner une **plage de variables consécutives** :

```
select(age:occupation) # Sélectionnez toutes les colonnes de `age` à
`occupation`
```

```
## # A tibble: 5 × 4
##   age sex    highest_education
##   <dbl> <chr>  <chr>
## 1    45 Female Secondary
## 2    55 Male   University
## 3    23 Male   University
## 4    20 Female Secondary
## 5    55 Female Primary
##   occupation
##   <chr>
## 1 Informal worker
## 2 Salaried worker
## 3 Student
```

```
## 4 Student
## 5 Trader--Farmer
```

Nous pouvons également spécifier une plage avec des numéros de colonne :

```
select(1:4) # Sélectionnez les colonnes 1 à 4
```

```
## # A tibble: 5 × 4
##   age sex    highest_education
##   <dbl> <chr>    <chr>
## 1    45 Female Secondary
## 2    55 Male    University
## 3    23 Male    University
## 4    20 Female Secondary
## 5    55 Female Primary
##   occupation
##   <chr>
## 1 Informal worker
## 2 Salaried worker
## 3 Student
## 4 Student
## 5 Trader--Farmer
```

PRACTICE



(in RMD)

- Avec le dataframe `yaounde`, sélectionnez les colonnes entre `symptoms` et `sequelae`, inclusivement. ("Inclusivement" signifie que vous devriez également inclure `symptoms` et `sequelae` dans la sélection.)

Exclure des colonnes avec !

Le point d'exclamation nie une sélection :

```
select(!age) # Sélectionnez toutes les colonnes sauf `age`
```

```
## # A tibble: 5 × 7
##   sex    highest_education occupation
##   <chr>    <chr>            <chr>
## 1 Female Secondary      Informal worker
## 2 Male    University      Salaried worker
## 3 Male    University      Student
## 4 Female Secondary      Student
## 5 Female Primary        Trader--Farmer
##   is_smoker is_pregnant igg_result igm_result
##   <chr>    <chr>        <chr>    <chr>
## 1 Non-smoker No          Negative Negative
## 2 Ex-smoker <NA>         Positive Negative
```



```
## 3 Smoker      <NA>      Negative Negative
## 4 Non-smoker No        Positive Negative
## 5 Non-smoker No        Positive Negative
```

Pour supprimer une plage de colonnes consécutives, nous utilisons, par exemple, `!age:occupation` :

```
select(!age:occupation) # Supprimez les colonnes de `age` à `occupation`
```

```
## # A tibble: 5 × 4
##   is_smoker is_pregnant igg_result igm_result
##   <chr>      <chr>      <chr>      <chr>
## 1 Non-smoker No        Negative Negative
## 2 Ex-smoker  <NA>      Positive Negative
## 3 Smoker     <NA>      Negative Negative
## 4 Non-smoker No        Positive Negative
## 5 Non-smoker No        Positive Negative
```

Pour supprimer plusieurs colonnes non consécutives, placez-les à l'intérieur de `!c()` :

```
select(!c(age, sex, igg_result)) # Supprimez les colonnes de `age`, `sex`, et `igg_result`
```

```
## # A tibble: 5 × 5
##   highest_education occupation is_smoker
##   <chr>             <chr>      <chr>
## 1 Secondary        Informal worker Non-smoker
## 2 University       Salaried worker Ex-smoker
## 3 University       Student      Smoker
## 4 Secondary        Student      Non-smoker
## 5 Primary          Trader--Farmer Non-smoker
##   is_pregnant igm_result
##   <chr>      <chr>
## 1 No        Negative
## 2 <NA>      Negative
## 3 <NA>      Negative
## 4 No        Negative
## 5 No        Negative
```

PRACTICE



(in RMD)

- Du dataframe `yaounde`, retirez toutes les colonnes entre `highest_education` et `consultation`, inclusivement.

Fonctions auxiliaires pour `select()`

`dplyr` possède un certain nombre de fonctions auxiliaires pour faciliter la sélection en utilisant des motifs des noms de colonnes. Jetons un œil à certaines d'entre elles.

`starts_with()` et `ends_with()`

Ces deux fonctions auxiliaires fonctionnent exactement comme leurs noms l'indiquent !

```
select(starts_with("is_")) # Colonnes qui commencent par "is"
```

```
## # A tibble: 5 × 2
##   is_smoker is_pregnant
##   <chr>      <chr>
## 1 Non-smoker No
## 2 Ex-smoker  <NA>
## 3 Smoker     <NA>
## 4 Non-smoker No
## 5 Non-smoker No
```

```
select(ends_with("_result")) # Colonnes qui finissent par "result"
```

```
## # A tibble: 5 × 2
##   igg_result igm_result
##   <chr>      <chr>
## 1 Negative  Negative
## 2 Positive  Negative
## 3 Negative  Negative
## 4 Positive  Negative
## 5 Positive  Negative
```

`contains()`

`contains()` aide à sélectionner les colonnes qui contiennent une certaine chaîne :

```
%>% select(contains("drug")) # Colonnes contenant la chaîne "drug"
```

```
## # A tibble: 5 × 12
##   drugsource is_drug_parac is_drug_antibio
##   <chr>      <dbl>      <dbl>
## 1 Self or famili...      1          0
## 2 <NA>                NA          NA
## 3 <NA>                NA          NA
## 4 Self or famili...      0          1
```

```
## 5 <NA> NA NA
## is_drug_hydrocortisone is_drug_other_anti_i...1
## <dbl> <dbl>
## 1 0 0
## 2 NA NA
## 3 NA NA
## 4 0 0
## 5 NA NA
## # i abbreviated name: 'is_drug_other_anti_inflam
## # i 7 more variables: is_drug_antiviral <dbl>, ...
```

`everything()`

Une autre fonction auxiliaire, **`everything()`**, correspond à toutes les variables qui n'ont pas encore été sélectionnées.

```
rd, `is_pregnant`, puis toutes les autres colonnes.
select(is_pregnant, everything())
```

```
## # A tibble: 5 × 8
## is_pregnant age sex highest_education
## <chr> <dbl> <chr> <chr>
## 1 No 45 Female Secondary
## 2 <NA> 55 Male University
## 3 <NA> 23 Male University
## 4 No 20 Female Secondary
## 5 No 55 Female Primary
## occupation is_smoker igg_result
## <chr> <chr> <chr>
## 1 Informal worker Non-smoker Negative
## 2 Salaried worker Ex-smoker Positive
## 3 Student Smoker Negative
## 4 Student Non-smoker Positive
## 5 Trader--Farmer Non-smoker Positive
## # i 1 more variable: igm_result <chr>
```

C'est souvent utile pour établir l'ordre des colonnes.

Disons que nous voulions amener la colonne `is_pregnant` au début du dataframe `yao`, nous pourrions écrire manuellement tous les noms des colonnes :

```
select(is_pregnant,
  age,
  sex,
  highest_education,
  occupation,
  is_smoker,
  igg_result,
  igm_result)
```

```
## # A tibble: 5 × 8
##   is_pregnant age sex highest_education
##   <chr>      <dbl> <chr> <chr>
## 1 No        45 Female Secondary
## 2 <NA>      55 Male University
## 3 <NA>      23 Male University
## 4 No        20 Female Secondary
## 5 No        55 Female Primary
##   occupation is_smoker igg_result
##   <chr>      <chr> <chr>
## 1 Informal worker Non-smoker Negative
## 2 Salaried worker Ex-smoker Positive
## 3 Student          Smoker Negative
## 4 Student          Non-smoker Positive
## 5 Trader--Farmer Non-smoker Positive
## # i 1 more variable: igm_result <chr>
```

Mais ce serait pénible pour des dataframes plus grands, comme notre dataframe original **yaounde**. Dans un tel cas, nous pouvons utiliser **everything()** :

```
# is_pregnant` à l'avant du dataframe
%>% select(is_pregnant, everything())
```

```
## # A tibble: 5 × 53
##   is_pregnant id date_surveyed
##   <chr>      <chr> <date>
## 1 No        BRIQUETERIE_000_00... 2020-10-22
## 2 <NA>      BRIQUETERIE_000_00... 2020-10-24
## 3 <NA>      BRIQUETERIE_000_00... 2020-10-24
## 4 No        BRIQUETERIE_002_00... 2020-10-22
## 5 No        BRIQUETERIE_002_00... 2020-10-22
##   age age_category age_category_3 sex
##   <dbl> <chr> <chr> <chr>
## 1 45 45 - 64 Adult Female
## 2 55 45 - 64 Adult Male
## 3 23 15 - 29 Adult Male
## 4 20 15 - 29 Adult Female
## 5 55 45 - 64 Adult Female
## # i 46 more variables: highest_education <chr>,
## # occupation <chr>, weight_kg <dbl>, height_cm <dbl>, ...
```

Cette fonction auxiliaire peut être combinée avec beaucoup d'autres.

```
# les colonnes qui se terminent par "result" à l'avant du dataframe
%>% select(ends_with("result"), everything())
```

```
## # A tibble: 5 × 53
##   igg_result igm_result id
##   <chr>      <chr> <chr>
```

```
## 1 Negative Negative BRIQUETERIE_000_0001
## 2 Positive Negative BRIQUETERIE_000_0002
## 3 Negative Negative BRIQUETERIE_000_0003
## 4 Positive Negative BRIQUETERIE_002_0001
## 5 Positive Negative BRIQUETERIE_002_0002
##   date_surveyed   age age_category
##   <date>         <dbl> <chr>
## 1 2020-10-22      45 45 - 64
## 2 2020-10-24      55 45 - 64
## 3 2020-10-24      23 15 - 29
## 4 2020-10-22      20 15 - 29
## 5 2020-10-22      55 45 - 64
## # i 47 more variables: age_category_3 <chr>, sex <chr>,
## #   highest_education <chr>, occupation <chr>, ...
```

PRACTICE



(in RMD)

- Sélectionnez toutes les colonnes dans le dataframe `yaounde` qui commencent par “is_”.
- Déplacez les colonnes qui commencent par “is_” au début du dataframe `yaounde`.

Changer les noms de colonnes avec `rename()`

`dplyr::rename()` est utilisée pour changer les noms des colonnes :

```
nez `age` et `sex` en `patient_age` et `patient_sex`
%>%
  rename(patient_age = age,
         patient_sex = sex)
```

```
## # A tibble: 5 × 53
##   id                date_surveyed patient_age
##   <chr>             <date>         <dbl>
## 1 BRIQUETERIE_000_00... 2020-10-22      45
## 2 BRIQUETERIE_000_00... 2020-10-24      55
## 3 BRIQUETERIE_000_00... 2020-10-24      23
## 4 BRIQUETERIE_002_00... 2020-10-22      20
## 5 BRIQUETERIE_002_00... 2020-10-22      55
##   age_category age_category_3 patient_sex
##   <chr>         <chr>         <chr>
## 1 45 - 64      Adult        Female
## 2 45 - 64      Adult        Male
## 3 15 - 29      Adult        Male
## 4 15 - 29      Adult        Female
```

```
## 5 45 - 64      Adult      Female
## # i 47 more variables: highest_education <chr>,
## #   occupation <chr>, weight_kg <dbl>, height_cm <dbl>, ...
```

WATCH OUT



Le fait que le nouveau nom vienne en premier dans la fonction (`rename(NOUVEAUNOM = VIEUXNOM)`) est parfois déroutant. Vous devriez vous y habituer avec le temps.

Renommer dans `select()`

Vous pouvez également renommer des colonnes tout en les sélectionnant :

```
selectionnez `age` et `sex`, et renommez-les en `patient_age` et `patient_sex`
%>%
select(patient_age = age,
patient_sex = sex)
```

```
## # A tibble: 5 × 2
##   patient_age patient_sex
##   <dbl> <chr>
## 1      45 Female
## 2      55 Male
## 3      23 Male
## 4      20 Female
## 5      55 Female
```

Conclusion !

J'espère que cette première leçon vous a permis de voir à quel point les verbes {dplyr} sont intuitifs et utiles ! Ceci est la première d'une série de verbes de base pour la manipulation de données : rendez-vous à la prochaine leçon pour en savoir plus.

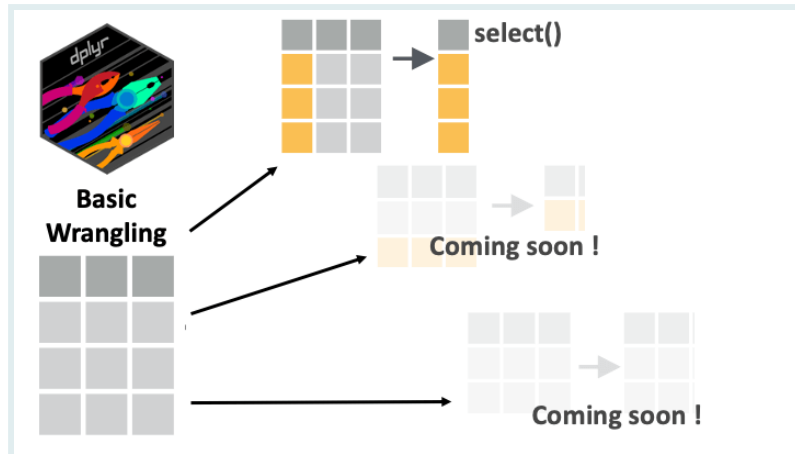


Fig: Basic Data Wrangling Dplyr Verbs.

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 colonnes en utilisant leurs noms et types---Select*. (n.d.). Récupéré le 31 décembre 2021, de <https://dplyr.tidyverse.org/reference/select.html>

L'artwork a été adapté de :

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