

R : Introduction à R — Analyse R | Cours 5

Ményssa Cherifa-Luron

2023-09-15

Contents

Introduction	2
Organisation	2
Ressources	2
Importation de données externes	2
Importation de Données txt avec <code>readLines</code>	2
Importation de Données CSV avec <code>read.csv</code>	2
Importation de Données Excel avec <code>readxl</code>	4
Manipuler des données avec Dplyr	4
Aperçu des Données : Starwars	4
L'opérateur : “%>%”	6
Les fonctions avancées	6
Sélection et Filtrage Avancés	6
Manipulation de colonnes	10
Groupement et Résumé	12
Tri et Organisation	14
Jointures	14
Aggrégation Conditionnelle et Opérations Plus Complexes	15
Lien entre Dplyr et ggplot2	16
Philosophie Commune : Tidyverse	16
Lien Direct dans le Workflow d'Analyse	16
Exemple de Workflow	17

Introduction

Ce cours vous guide à travers les fondamentaux de l'importation de données en R, en couvrant divers formats de fichiers tels que les fichiers texte, CSV et Excel. Vous apprendrez à utiliser des fonctions comme `readLines`, `read.csv`, et `readxl` pour charger des données dans votre environnement R depuis diverses sources externes.

Par la suite, le cours se concentre sur la gestion de données en utilisant le package `dplyr`, un outil puissant et flexible pour la manipulation de données. Vous explorerez des concepts clés tels que l'opérateur `%>%` pour chaîner les commandes, ainsi que diverses fonctions avancées pour la sélection, le filtrage, la manipulation de colonnes, le groupement, le tri, et les jointures. Des techniques d'aggrégation conditionnelle et d'opérations complexes seront également abordées.

À travers des exemples pratiques, notamment avec le jeu de données `starwars`, ce cours vise à renforcer vos compétences en matière de préparation et d'analyse de données, vous rendant ainsi plus apte à tirer des insights pertinents de vos ensembles de données.

Organisation

Contenu du Cours : - Démonstrations - Exercices

Ressources

- Rbloggers : Blog Populaire sur le langage R
- Datacamp : Plateforme d'apprentissage de la data science interactive
- Big Book of R : Edition qui rassemble des livres open-source sur le langage R
- Introduction accélérée au langage R pour la data science : L'essentiel et plus de tout ce que nous verrons

Importation de données externes

Voici une explication détaillée des scripts R que vous avez fournis, qui traitent de l'importation de données externes et de l'aperçu des données dans le contexte du jeu de données `starwars`.

Importation de Données txt avec `readLines`

```
texte <- readLines("../data/orgueil_et_prejuges.txt", encoding = "UTF-8")
```

Importation de Données CSV avec `read.csv`

```
# Importation d'un fichier CSV
bike <- read.csv("../data/Animation.csv", header = TRUE, sep = ";")
```

- `read.csv` est une fonction de base en R pour lire les fichiers CSV.
- `header = TRUE` indique que la première ligne du fichier contient les noms des colonnes.
- `sep = ";"` spécifie que le séparateur de colonnes dans le fichier CSV est un point-virgule.



Figure 1: Illustration du langage R par DALL-E

Importation de Données Excel avec readxl

```
# Importation d'un fichier Excel
library(readxl)

bike2 <- read_excel("../data/BikeSalesAnalysis.xlsx", sheet = "BikeSales")
```

- `read_excel` est une fonction du package `readxl`, utilisée pour lire des fichiers Excel.
- `sheet = "BikeSales"` spécifie le nom de la feuille de calcul à importer.

```
# Récupération des noms des feuilles dans un fichier Excel
liste_noms <- excel_sheets("../data/BikeSalesAnalysis.xlsx")
```

- `excel_sheets` est utilisée pour lister toutes les feuilles disponibles dans un fichier Excel.

Manipuler des données avec Dplyr

La manipulation de données avec `dplyr` à l'aide de **verbes**, qui correspondent chacun à une action différente appliquée à un tableau de données.

Aperçu des Données : Starwars

```
library(tidyverse)

# Ouvre un aperçu interactif de la dataframe
View(starwars)
```

- `View` ouvre une fenêtre interactive pour visualiser la dataframe `starwars`. C'est utile pour explorer les données de manière graphique.

Exploration des Données avec glimpse

- `glimpse` est un verbe du package `dplyr` qui fournit un aperçu rapide de la structure d'une dataframe, y compris les noms des colonnes, les types de données et les premières valeurs de chaque colonne.

```
# Aperçu structuré des données
glimpse(starwars)

## Rows: 87
## Columns: 14
## $ name      <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Leia Or~
## $ height    <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 180, 2~
## $ mass      <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 84.0, 77.~
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "brown", N~
## $ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "light", "~
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue", "blue",~
```

```
## $ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0, 57.0, ~
## $ sex <chr> "male", "none", "none", "male", "female", "male", "female", ~
## $ gender <chr> "masculine", "masculine", "masculine", "masculine", "femini~
## $ homeworld <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan", "T~
## $ species <chr> "Human", "Droid", "Droid", "Human", "Human", "Human", "Huma~
## $ films <list> <"The Empire Strikes Back", "Revenge of the Sith", "Return~
## $ vehicles <list> <"Snowspeeder", "Imperial Speeder Bike">, <>, <>, <>, "Imp~
## $ starships <list> <"X-wing", "Imperial shuttle">, <>, <>, "TIE Advanced x1",~
```

- slice sélectionne des lignes du tableau selon leur position. On lui passe un chiffre ou un vecteur de chiffres.

```
# sélection des dernières lignes de starwars
slice_tail(starwars, n = 5)
```

```
## # A tibble: 5 x 14
##   name      height  mass hair_color skin_color eye_color birth_year sex  gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 Rey            NA    NA brown      light      hazel            NA fema~ femin~
## 2 Poe Dame~      NA    NA brown      light      brown            NA male  mascu~
## 3 BB8            NA    NA none       none       black            NA none  mascu~
## 4 Captain ~      NA    NA unknown  unknown  unknown          NA <NA> <NA>
## 5 Padmé Am~    165    45 brown      light      brown            46 fema~ femin~
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```
# sélection des premières lignes de starwars
slice_head(starwars, n = 5)
```

```
## # A tibble: 5 x 14
##   name      height  mass hair_color skin_color eye_color birth_year sex  gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 Luke Sky~    172    77 blond      fair      blue            19  male  mascu~
## 2 C-3P0        167    75 <NA>      gold      yellow           112  none  mascu~
## 3 R2-D2         96    32 <NA>      white, bl~ red            33  none  mascu~
## 4 Darth Va~    202   136 none       white     yellow           41.9  male  mascu~
## 5 Leia Org~    150    49 brown      light     brown            19  fema~ femin~
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```
# sélection aléatoire de 5 lignes de starwars
slice_sample(starwars, n = 5)
```

```
## # A tibble: 5 x 14
##   name      height  mass hair_color skin_color eye_color birth_year sex  gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 R2-D2         96    32 <NA>      white, bl~ red            33  none  mascu~
## 2 Adi Gall~    184    50 none       dark      blue            NA  fema~ femin~
## 3 Tion Med~    206    80 none       grey      black            NA  male  mascu~
## 4 Palpatine    170    75 grey      pale      yellow           82  male  mascu~
## 5 R5-D4         97    32 <NA>      white, red red            NA  none  mascu~
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

L'opérateur : “%>%”

La programmation avec le “pipe” en R, symbolisée par l'opérateur %>%, est une technique populaire pour chaîner des opérations de manière lisible.

Cet opérateur, popularisé par le package `magrittr` et largement utilisé avec `dplyr`, permet de transférer le résultat d'une expression vers une autre, rendant le code plus lisible et évitant la création de multiples variables temporaires.

Les fonctions avancées

Les fonctions avancées utilisées avec l'opérateur %>% dans R, en particulier avec le package `dplyr`, offrent des capacités étendues pour manipuler et analyser des ensembles de données de manière efficace et intuitive.

Voici un aperçu de quelques-unes de ces fonctions :

Sélection et Filtrage Avancés

- **`select()`** : Permet de choisir des colonnes spécifiques dans un data frame. Peut être utilisée avec des fonctions comme `contains()`, `starts_with()`, `ends_with()`, et `matches()` pour sélectionner des colonnes basées sur des motifs dans leurs noms.

```
# Sélectionner toutes les colonnes SAUF 'name' : select()
starwars %>%
  select(-name)
```

```
## # A tibble: 87 x 13
##   height mass hair_color skin_color eye_color birth_year sex gender
##   <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1    172    77 blond      fair        blue        19   male  masculine
## 2    167    75 <NA>      gold        yellow     112   none  masculine
## 3     96    32 <NA>      white, blue red         33   none  masculine
## 4    202   136 none      white      yellow     41.9  male  masculine
## 5    150    49 brown      light      brown       19  female feminine
## 6    178   120 brown, grey light      blue       52   male  masculine
## 7    165    75 brown      light      blue       47  female feminine
## 8     97    32 <NA>      white, red  red        NA   none  masculine
## 9    183    84 black      light      brown       24   male  masculine
## 10   182    77 auburn, white fair        blue-gray   57   male  masculine
## # i 77 more rows
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```
# Sélectionner uniquement les colonnes 'name' et 'skin_color'
starwars %>%
  select(name, skin_color)
```

```
## # A tibble: 87 x 2
##   name          skin_color
##   <chr>         <chr>
## 1 Luke Skywalker fair
## 2 C-3PO         gold
```

```
## 3 R2-D2          white, blue
## 4 Darth Vader    white
## 5 Leia Organa    light
## 6 Owen Lars      light
## 7 Beru Whitesun lars light
## 8 R5-D4          white, red
## 9 Biggs Darklighter light
## 10 Obi-Wan Kenobi fair
## # i 77 more rows
```

```
# Sélectionner uniquement les colonnes contenant un "_" : contains("_")
starwars %>%
  select(contains("_"))
```

```
## # A tibble: 87 x 4
##   hair_color  skin_color eye_color birth_year
##   <chr>      <chr>      <chr>      <dbl>
## 1 blond      fair        blue        19
## 2 <NA>      gold        yellow     112
## 3 <NA>      white, blue red         33
## 4 none      white        yellow    41.9
## 5 brown     light        brown       19
## 6 brown, grey light        blue        52
## 7 brown     light        blue        47
## 8 <NA>      white, red  red         NA
## 9 black     light        brown       24
## 10 auburn, white fair        blue-gray   57
## # i 77 more rows
```

```
# Sélectionner uniquement les colonnes commençant par "s" : starts_with("s")
starwars %>%
  select(starts_with("s"))
```

```
## # A tibble: 87 x 4
##   skin_color sex   species starships
##   <chr>      <chr> <chr>    <list>
## 1 fair      male  Human   <chr [2]>
## 2 gold      none  Droid   <chr [0]>
## 3 white, blue none  Droid   <chr [0]>
## 4 white     male  Human   <chr [1]>
## 5 light     female Human   <chr [0]>
## 6 light     male  Human   <chr [0]>
## 7 light     female Human   <chr [0]>
## 8 white, red none  Droid   <chr [0]>
## 9 light     male  Human   <chr [1]>
## 10 fair     male  Human   <chr [5]>
## # i 77 more rows
```

```
# Sélectionner la colonne 'name' et toutes les colonnes se terminant par "color" :
# ends_with("color")
starwars %>%
  select(ends_with("color"))
```

```
## # A tibble: 87 x 3
##   hair_color    skin_color eye_color
##   <chr>         <chr>      <chr>
## 1 blond        fair        blue
## 2 <NA>         gold        yellow
## 3 <NA>         white, blue red
## 4 none         white        yellow
## 5 brown        light        brown
## 6 brown, grey  light        blue
## 7 brown        light        blue
## 8 <NA>         white, red  red
## 9 black        light        brown
## 10 auburn, white fair        blue-gray
## # i 77 more rows
```

- **distinct()** : filtre les lignes du tableau pour ne conserver que les lignes distinctes, en supprimant toutes les lignes en double.

```
# Sélectionner uniquement les colonnes 'name' et 'skin color'
starwars %>%
  select(skin_color) %>%
  distinct(skin_color)
```

```
## # A tibble: 31 x 1
##   skin_color
##   <chr>
## 1 fair
## 2 gold
## 3 white, blue
## 4 white
## 5 light
## 6 white, red
## 7 unknown
## 8 green
## 9 green-tan, brown
## 10 pale
## # i 21 more rows
```

- **filter()** : Utilisée pour extraire des lignes qui satisfont certaines conditions. Supporte les opérateurs logiques (&, |) et peut être utilisée pour des comparaisons complexes.

```
# Filtrer les personnages dont l'espèce est "Human" : filter()
starwars %>%
  filter(species == "Human")
```

```
## # A tibble: 35 x 14
##   name      height mass hair_color skin_color eye_color birth_year sex  gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 Luke Sk~    172    77 blond      fair        blue          19 male masculin
## 2 Darth V~    202   136 none       white       yellow        41.9 male masculin
## 3 Leia Or~    150    49 brown      light       brown          19 fema~ féminin
## 4 Owen La~    178   120 brown, gr~ light       blue          52 male masculin
```



```
## 5 Beru Wh~ 165 75 brown light blue 47 fema~ femin~
## 6 Biggs D~ 183 84 black light brown 24 male mascu~
## 7 Obi-Wan~ 182 77 auburn, w~ fair blue-gray 57 male mascu~
## 8 Anakin ~ 188 84 blond fair blue 41.9 male mascu~
## 9 Wilhuff~ 180 NA auburn, g~ fair blue 64 male mascu~
## 10 Han Solo 180 80 brown fair brown 29 male mascu~
## # i 25 more rows
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## # vehicles <list>, starships <list>
```

```
# Filtrer et supprimer le personnage "Jabba Desilijic Tiure"
starwars %>%
  filter(name != "Jabba Desilijic Tiure")
```

```
## # A tibble: 86 x 14
##   name      height mass hair_color skin_color eye_color birth_year sex gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 Luke Sk~ 172 77 blond fair blue 19 male mascu~
## 2 C-3PO 167 75 <NA> gold yellow 112 none mascu~
## 3 R2-D2 96 32 <NA> white, bl~ red 33 none mascu~
## 4 Darth V~ 202 136 none white yellow 41.9 male mascu~
## 5 Leia Or~ 150 49 brown light brown 19 fema~ femin~
## 6 Owen La~ 178 120 brown, gr~ light blue 52 male mascu~
## 7 Beru Wh~ 165 75 brown light blue 47 fema~ femin~
## 8 R5-D4 97 32 <NA> white, red red NA none mascu~
## 9 Biggs D~ 183 84 black light brown 24 male mascu~
## 10 Obi-Wan~ 182 77 auburn, w~ fair blue-gray 57 male mascu~
## # i 76 more rows
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## # vehicles <list>, starships <list>
```

```
# Filtrer les personnages dont l'espèce est soit "Human", soit "Droid"
starwars %>%
  filter(species == "Human" | species == "Droid")
```

```
## # A tibble: 41 x 14
##   name      height mass hair_color skin_color eye_color birth_year sex gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 Luke Sk~ 172 77 blond fair blue 19 male mascu~
## 2 C-3PO 167 75 <NA> gold yellow 112 none mascu~
## 3 R2-D2 96 32 <NA> white, bl~ red 33 none mascu~
## 4 Darth V~ 202 136 none white yellow 41.9 male mascu~
## 5 Leia Or~ 150 49 brown light brown 19 fema~ femin~
## 6 Owen La~ 178 120 brown, gr~ light blue 52 male mascu~
## 7 Beru Wh~ 165 75 brown light blue 47 fema~ femin~
## 8 R5-D4 97 32 <NA> white, red red NA none mascu~
## 9 Biggs D~ 183 84 black light brown 24 male mascu~
## 10 Obi-Wan~ 182 77 auburn, w~ fair blue-gray 57 male mascu~
## # i 31 more rows
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## # vehicles <list>, starships <list>
```

```
# Filtrer les personnages de l'espèce "Human" et sélectionner uniquement
# les colonnes 'name', 'height' et 'birth_year'
starwars %>%
  filter(species == "Human") %>%
  select(name, height, birth_year)
```

```
## # A tibble: 35 x 3
##   name          height birth_year
##   <chr>         <int>     <dbl>
## 1 Luke Skywalker    172         19
## 2 Darth Vader      202        41.9
## 3 Leia Organa      150         19
## 4 Owen Lars        178         52
## 5 Beru Whitesun lars 165         47
## 6 Biggs Darklighter 183         24
## 7 Obi-Wan Kenobi    182         57
## 8 Anakin Skywalker  188        41.9
## 9 Wilhuff Tarkin    180         64
## 10 Han Solo         180         29
## # i 25 more rows
```

Manipulation de colonnes

- **separate()** : Divise une colonne en plusieurs colonnes en fonction d'un séparateur spécifié, ce qui est utile pour séparer les noms complets en prénoms et noms de famille, par exemple.

```
# Séparer la colonne 'name' en plusieurs colonnes
# (prénom, deuxième prénom, troisième prénom) en utilisant l'espace comme séparateur ;
starwars %>%
  separate(name, c("prenom", "nom"), sep = " ")
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 7 rows [7, 16, 18, 29, 34,
## 60, 65].
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 24 rows [2, 3, 8, 13, 15,
## 19, 20, 22, 23, 25, 26, 38, 39, 46, 49, 58, 63, 64, 73, 77, ...].
```

```
## # A tibble: 87 x 15
##   prenom nom height mass hair_color skin_color eye_color birth_year sex
##   <chr> <chr> <int> <dbl> <chr> <chr> <chr> <dbl> <chr>
## 1 Luke Skywal~ 172 77 blond fair blue 19 male
## 2 C-3PO <NA> 167 75 <NA> gold yellow 112 none
## 3 R2-D2 <NA> 96 32 <NA> white, bl~ red 33 none
## 4 Darth Vader 202 136 none white yellow 41.9 male
## 5 Leia Organa 150 49 brown light brown 19 fema~
## 6 Owen Lars 178 120 brown, gr~ light blue 52 male
## 7 Beru Whites~ 165 75 brown light blue 47 fema~
## 8 R5-D4 <NA> 97 32 <NA> white, red red NA none
## 9 Biggs Darkli~ 183 84 black light brown 24 male
## 10 Obi-Wan Kenobi 182 77 auburn, w~ fair blue-gray 57 male
## # i 77 more rows
## # i 6 more variables: gender <chr>, homeworld <chr>, species <chr>,
## # films <list>, vehicles <list>, starships <list>
```

- **mutate()** : Ajoute de nouvelles colonnes ou modifie des colonnes existantes. Par exemple, vous pouvez l'utiliser pour calculer de nouvelles valeurs à partir des données existantes.
- **transmute()** : Semblable à **mutate()**, mais ne garde que les nouvelles colonnes créées.

Calcul de l'Indice de Masse Corporelle (BMI)

Le BMI (Indice de Masse Corporelle) est calculé en divisant la masse (en kilogrammes) par le carré de la hauteur (en mètres).

```
starwars %>%
  select(mass, height) %>%
  mutate(bmi = mass / (height / 100)^2)
```

```
## # A tibble: 87 x 3
##   mass height  bmi
##   <dbl> <int> <dbl>
## 1    77   172  26.0
## 2    75   167  26.9
## 3    32    96  34.7
## 4   136   202  33.3
## 5    49   150  21.8
## 6   120   178  37.9
## 7    75   165  27.5
## 8    32    97  34.0
## 9    84   183  25.1
## 10   77   182  23.2
## # i 77 more rows
```

```
starwars %>%
  select(name, mass, height) %>%
  transmute(bmi = mass / (height / 100)^2)
```

```
## # A tibble: 87 x 1
##   bmi
##   <dbl>
## 1  26.0
## 2  26.9
## 3  34.7
## 4  33.3
## 5  21.8
## 6  37.9
## 7  27.5
## 8  34.0
## 9  25.1
## 10 23.2
## # i 77 more rows
```

```
# Replace() remplace toutes les occurrences de 41.9 dans la colonne birth_year par 41.
starwars %>%
  mutate(birth_year = replace(birth_year, birth_year == 41.9, 41))
```

```
## # A tibble: 87 x 14
##   name      height  mass hair_color skin_color eye_color birth_year sex  gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 Luke Sk~    172    77 blond      fair        blue         19 male  mascu~
## 2 C-3PO      167    75 <NA>      gold        yellow        112 none  mascu~
## 3 R2-D2       96    32 <NA>      white, bl~  red          33 none  mascu~
## 4 Darth V~   202   136 none      white       yellow        41 male  mascu~
## 5 Leia Or~   150    49 brown     light       brown        19 fema~ femin~
## 6 Owen La~   178   120 brown, gr~ light       blue         52 male  mascu~
## 7 Beru Wh~   165    75 brown     light       blue         47 fema~ femin~
## 8 R5-D4       97    32 <NA>      white, red  red          NA none  mascu~
## 9 Biggs D~   183    84 black     light       brown        24 male  mascu~
## 10 Obi-Wan~  182    77 auburn, w~ fair        blue-gray    57 male  mascu~
## # i 77 more rows
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

- **rename()** : Renommer les colonnes d'un data frame.

```
# Renommer les colonnes 'height' et 'mass' en 'height_cm' et 'mass_kg' : rename()
starwars %>%
  rename("height_cm" = "height",
        "mass_kg" = "mass") %>%
  select(height_cm, mass_kg)
```

```
## # A tibble: 87 x 2
##   height_cm mass_kg
##   <int>    <dbl>
## 1     172     77
## 2     167     75
## 3      96     32
## 4     202    136
## 5     150     49
## 6     178    120
## 7     165     75
## 8      97     32
## 9     183     84
## 10     182     77
## # i 77 more rows
```

Groupement et Résumé

- **group_by()** : Permet de regrouper les données par une ou plusieurs colonnes. Très utile pour les analyses ultérieures qui nécessitent des calculs par groupe.
- **summarise()** : Utilisée pour créer des résumés statistiques des groupes de données. Elle peut être utilisée pour calculer des moyennes, des sommes, des médianes, etc., sur des groupes spécifiés par **group_by()**.

```
starwars %>%
  group_by(species) %>%
  summarise(
```

```

average_height = mean(height, na.rm = TRUE),
average_mass = mean(mass, na.rm = TRUE),
count = n()
)

```

```

## # A tibble: 38 x 4
##   species average_height average_mass count
##   <chr>         <dbl>         <dbl> <int>
## 1 Aleena           79             15      1
## 2 Besalisk        198            102      1
## 3 Cerean          198             82      1
## 4 Chagrian        196             NA      1
## 5 Clawdite        168             55      1
## 6 Droid           131.            69.8      6
## 7 Dug             112             40      1
## 8 Ewok             88             20      1
## 9 Geonosian       183             80      1
## 10 Gungan         209.            74      3
## # i 28 more rows

```

Dans cet exemple : - `group_by(species)` regroupe les données par l'espèce. - `summarise()` est utilisé pour calculer les moyennes de la taille et de la masse pour chaque groupe d'espèces. `na.rm = TRUE` est utilisé pour exclure les valeurs manquantes (NA) des calculs. - `n()` est utilisé pour compter le nombre de personnages dans chaque groupe d'espèces.

```

# Regrouper les données par 'species' et 'hair_color'
starwars %>%
  group_by(species, hair_color) %>%
  summarise(count = n(), .groups = 'drop')

```

```

## # A tibble: 52 x 3
##   species hair_color count
##   <chr>    <chr>     <int>
## 1 Aleena  none         1
## 2 Besalisk none         1
## 3 Cerean  white         1
## 4 Chagrian none         1
## 5 Clawdite blonde         1
## 6 Droid   none         3
## 7 Droid   <NA>         3
## 8 Dug     none         1
## 9 Ewok    brown         1
## 10 Geonosian none         1
## # i 42 more rows

```

Dans cet exemple : - `group_by(species, hair_color)` regroupe les données à la fois par l'espèce et la couleur des cheveux. - `summarise(count = n(), .groups = 'drop')` calcule le nombre de personnages (`n()`) pour chaque combinaison d'espèce et de couleur de cheveux. L'option `.groups = 'drop'` est utilisée pour éviter que le résultat ne soit regroupé, ce qui facilite sa manipulation ultérieure. - `head(starwars_hair_color)` affiche les premières lignes du résultat pour avoir un aperçu.

Tri et Organisation

- **arrange()** : Trie les données en fonction d'une ou plusieurs colonnes. Peut être utilisée avec **desc()** pour un tri décroissant.

```
# Tri ascendant par la colonne 'height'
starwars_sorted_by_height <- starwars %>%
  arrange(height)
```

```
# Tri décroissant par la colonne 'mass'
starwars_sorted_by_mass <- starwars %>%
  arrange(desc(mass))
```

Jointures

Les fonctions de jointure dans **dplyr** permettent de combiner des data frames en fonction de colonnes clés, de manière similaire aux jointures dans SQL. Voici des exemples pour chaque type de jointure : **inner_join()**, **left_join()**, **right_join()**, et **full_join()**. Pour ces exemples, imaginons deux data frames simplifiés : **df1** et **df2**.

```
# Data frames exemples
df1 <- data.frame(id = c(1, 2, 3), nom = c("Alice", "Bob", "Carol"))
df2 <- data.frame(id = c(2, 3, 4), ville = c("Paris", "Lyon", "Marseille"))
```

inner_join()

inner_join() retourne toutes les lignes de **df1** et **df2** ayant des valeurs correspondantes dans les colonnes clés.

```
result_inner <- inner_join(df1, df2, by = "id")
```

Dans cet exemple, **result_inner** contient les lignes de **df1** et **df2** où les **id** correspondent.

left_join()

left_join() retourne toutes les lignes de **df1** et toutes les colonnes de **df1** et **df2**. Les lignes de **df2** sont incluses seulement si leurs clés correspondent à celles de **df1**.

```
result_left <- left_join(df1, df2, by = "id")
```

Ici, **result_left** contient toutes les lignes de **df1** avec les informations correspondantes de **df2**.

right_join()

right_join() fonctionne comme **left_join()**, mais il retourne toutes les lignes de **df2** et toutes les colonnes de **df1** et **df2**.

```
result_right <- right_join(df1, df2, by = "id")
```

result_right contient toutes les lignes de **df2** avec les informations correspondantes de **df1**.

full_join()

full_join() combine **df1** et **df2** en incluant toutes les lignes de **df1** et **df2**, même si les clés ne correspondent pas.

```
result_full <- full_join(df1, df2, by = "id")
```

result_full contient toutes les lignes de df1 et df2, avec des valeurs NA pour les données manquantes.

Aggrégation Conditionnelle et Opérations Plus Complexes

- `summarise_at()`, `mutate_at()`, `if_else()` : Permettent de réaliser des opérations sur un ensemble de colonnes spécifiées ou d'appliquer des conditions complexes.

`summarise_at()` est utilisée pour appliquer une ou plusieurs fonctions de résumé à un ensemble de colonnes spécifiées.

```
# Calculer la moyenne et l'écart-type pour certaines colonnes numériques
starwars %>%
  summarise_at(vars(height, mass), list(mean = ~mean(., na.rm = TRUE),
                                         sd = ~sd(., na.rm = TRUE)))
```

```
## # A tibble: 1 x 4
##   height_mean mass_mean height_sd mass_sd
##   <dbl>      <dbl>    <dbl>    <dbl>
## 1      174.      97.3      34.8     169.
```

Dans cet exemple, `summarise_at()` calcule la moyenne (`mean`) et l'écart-type (`sd`) pour les colonnes `height` et `mass`, en excluant les valeurs NA.

`mutate_at()` est utilisée pour appliquer une fonction à un ensemble de colonnes spécifiées.

```
# Normaliser les colonnes 'height' et 'mass'
starwars %>%
  mutate_at(vars(height, mass), ~(. - mean(., na.rm = TRUE)) / sd(., na.rm = TRUE))
```

```
## # A tibble: 87 x 14
##   name    height    mass hair_color skin_color eye_color birth_year sex  gender
##   <chr>    <dbl>    <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 Luke~ -0.0678 -0.120 blond     fair       blue       19   male masculin
## 2 C-3PO -0.212 -0.132 <NA>      gold       yellow     112  none masculin
## 3 R2-D2 -2.25  -0.385 <NA>      white, bl~ red        33   none masculin
## 4 Dart~  0.795  0.228 none      white      yellow     41.9 male masculin
## 5 Leia~ -0.701 -0.285 brown     light      brown      19   fema~ féminin
## 6 Owen~  0.105  0.134 brown, gr~ light      blue       52   male masculin
## 7 Beru~ -0.269 -0.132 brown     light      blue       47   fema~ féminin
## 8 R5-D4 -2.22  -0.385 <NA>      white, red red        NA   none masculin
## 9 Bigg~  0.249 -0.0786 black     light      brown      24   male masculin
## 10 Obi~- 0.220 -0.120 auburn, w~ fair       blue-gray  57   male masculin
## # i 77 more rows
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

Ici, `mutate_at()` normalise les valeurs des colonnes `height` et `mass` (soustraction de la moyenne et division par l'écart-type).

`if_else()` est une version vectorisée et type-safe de la fonction `ifelse()`.

```
# Créer une nouvelle colonne 'height_category'
starwars %>%
  mutate(height_category = if_else(height > 170, "tall", "short", missing = "unknown"))
```

```
## # A tibble: 87 x 15
##   name      height  mass hair_color skin_color eye_color birth_year sex  gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr> <chr>
## 1 Luke Sk~    172    77 blond      fair        blue        19   male mascu~
## 2 C-3PO      167    75 <NA>      gold        yellow       112  none mascu~
## 3 R2-D2       96    32 <NA>      white, bl~ red         33   none mascu~
## 4 Darth V~   202   136 none       white       yellow      41.9 male mascu~
## 5 Leia Or~   150    49 brown      light       brown       19   fema~ femin~
## 6 Owen La~   178   120 brown, gr~ light       blue       52   male mascu~
## 7 Beru Wh~   165    75 brown      light       blue       47   fema~ femin~
## 8 R5-D4       97    32 <NA>      white, red red         NA   none mascu~
## 9 Biggs D~   183    84 black      light       brown       24   male mascu~
## 10 Obi-Wan~  182    77 auburn, w~ fair        blue-gray   57   male mascu~
## # i 77 more rows
## # i 6 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>, height_category <chr>
```

Dans cet exemple, `if_else()` est utilisée pour créer une nouvelle colonne `height_category` basée sur la condition si la `height` est supérieure à 170. Si oui, la catégorie est “tall”, sinon “short”. En cas de valeurs NA, la catégorie est définie comme “unknown”.

Lien entre Dplyr et ggplot2

`dplyr` et `ggplot2` sont deux des packages les plus populaires en R pour la manipulation de données et la visualisation, respectivement. Ils font tous deux partie de l'écosystème de packages développé sous l'égide du projet `tidyverse`, créé par Hadley Wickham et d'autres. Voici les principaux aspects de leur relation et interaction :

Philosophie Commune : Tidyverse

1. **Approche Tidyverse** : `dplyr` et `ggplot2` adhèrent à la philosophie du “tidyverse”, qui encourage une syntaxe cohérente et lisible, et un format de données (“tidy data”) où chaque colonne est une variable et chaque ligne est une observation. Cette approche facilite grandement la manipulation et la visualisation des données.
2. **Pipelines et Lisibilité** : Les deux packages utilisent l'opérateur pipe `%>%` pour enchaîner des commandes de manière fluide, ce qui rend le code plus lisible et logique. Par exemple, avec `dplyr`, on peut transformer des données puis les passer directement à `ggplot2` pour la visualisation.

Lien Direct dans le Workflow d'Analyse

Dans un workflow typique d'analyse de données en R : 1. **Manipulation** : Les données sont d'abord manipulées et explorées à l'aide de `dplyr`. Par exemple, on peut grouper des données, calculer des résumés statistiques ou filtrer des sous-ensembles de données.

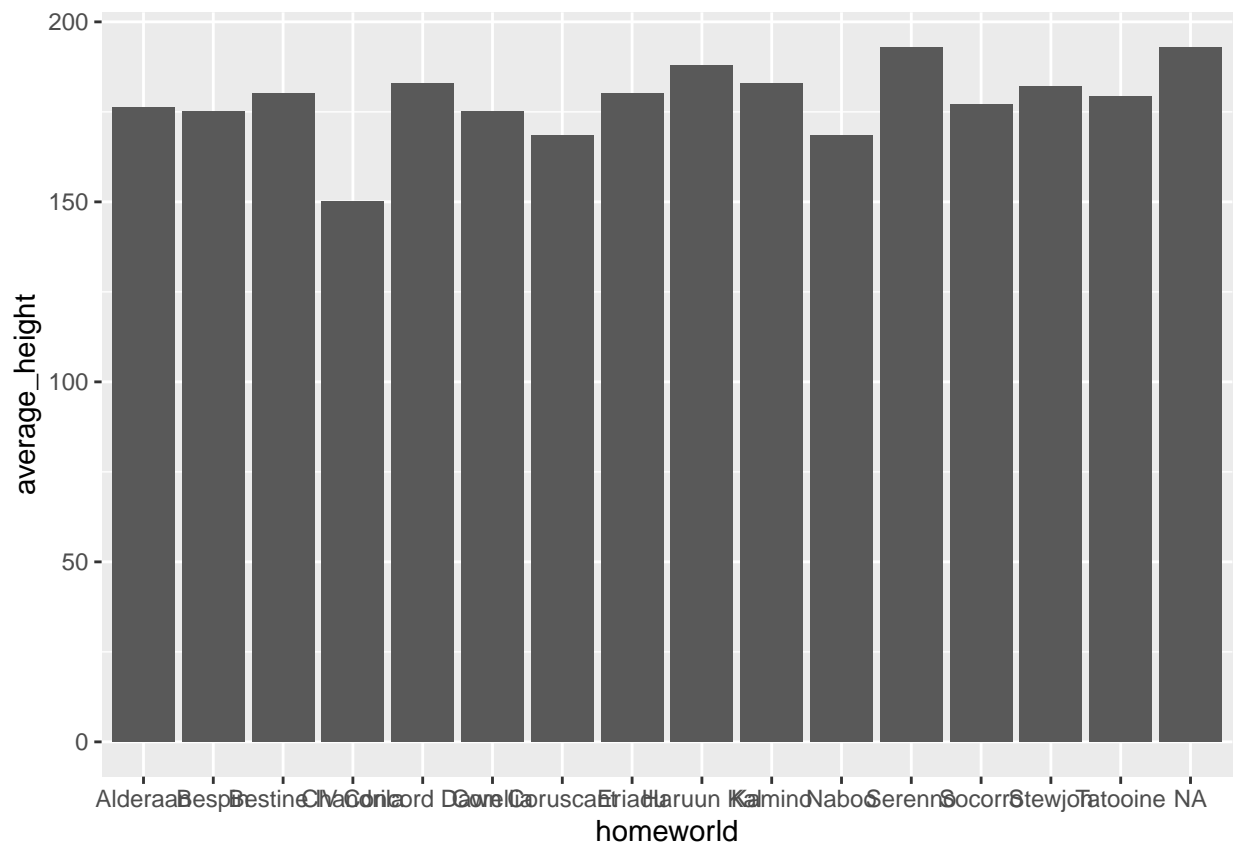
2. **Visualisation** : Ensuite, les données transformées sont visualisées avec `ggplot2`. La nature intégrée de ces deux packages permet un transfert fluide des données manipulées directement vers les fonctions de graphique.

Exemple de Workflow

```
library(dplyr)
library(ggplot2)

# Manipulation des données avec dplyr
data_processed <- starwars %>%
  filter(species == "Human") %>%
  group_by(homeworld) %>%
  summarize(average_height = mean(height, na.rm = TRUE))

# Visualisation avec ggplot2
ggplot(data_processed, aes(x = homeworld, y = average_height)) +
  geom_col()
```



Dans cet exemple, `dplyr` est utilisé pour préparer les données, qui sont ensuite visualisées avec `ggplot2`. Ce flux de travail intégré est un des principaux avantages de l'utilisation conjointe de `dplyr` et `ggplot2`.

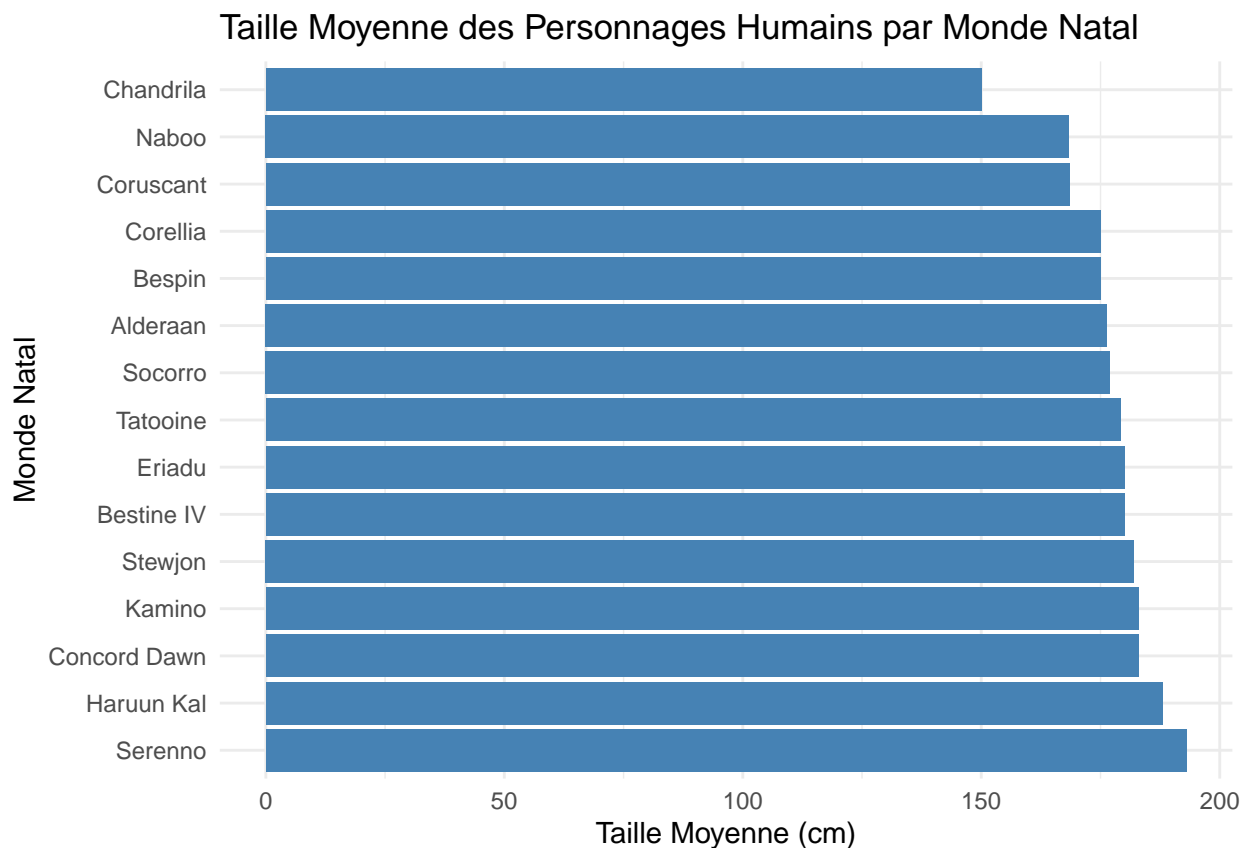
Voici un autre exemple avec les données `starwars`

```
starwars %>%
  filter(species == "Human") %>% # Filtrer pour garder seulement les humains
  select(name, homeworld, height) %>% # Sélectionner les colonnes pertinentes
  group_by(homeworld) %>% # Grouper par monde natal
  summarize(average_height = mean(height, na.rm = TRUE)) %>% # Calculer la taille moyenne
```

```

arrange(desc(average_height)) %>% # Trier par taille moyenne en ordre décroissant
# Ajout de ggplot pour visualiser les résultats
drop_na() %>% # Retire les personnages sans monde natal
ggplot(aes(x = reorder(homeworld, -average_height), y = average_height)) +
geom_bar(stat = "identity", fill = "steelblue") +
theme_minimal() +
labs(title = "Taille Moyenne des Personnages Humains par Monde Natal",
      x = "Monde Natal",
      y = "Taille Moyenne (cm)") +
coord_flip() # Retourner les axes pour une meilleure lisibilité

```



Exercices

Pour commencer à pratiquer, suivez ces étapes :

1. **Accédez au Dépôt GitHub** : Visitez l'URL fournie : <https://github.com/universdesdonnees/Introduction-a-R> pour accéder au dépôt GitHub contenant les matériaux du cours.
2. **Trouvez le Fichier des Exercices** : Dans le dépôt, localisez le fichier nommé **exercices5.txt**. Ce fichier contient les premiers exercices que vous devez pratiquer.
3. **Lisez et Essayez de Résoudre les Exercices** : Ouvrez le fichier **exercices5.txt** et lisez attentivement les exercices. Essayez de les résoudre par vous-même dans votre environnement R (comme RStudio). Il est important de pratiquer par vous-même avant de regarder les solutions pour mieux apprendre.
4. **Consultez la Correction** : Une fois que vous avez tenté de résoudre les exercices, ou si vous rencontrez

des difficultés, consultez le fichier **correction_exercices5.R** pour voir les solutions. Analysez les solutions pour comprendre les méthodes et logiques utilisées.