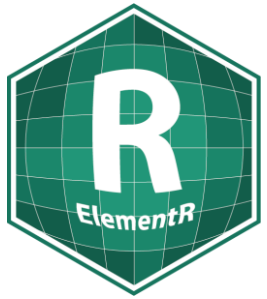


Initiation au langage R

Séance 2 d'introduction

Groupe ElementR

Joséphin Béraud, Léa Christophe, Aurélie Douet

16 décembre 2022

RIATE

PRODIG

Programme

- Rappel séance 1
- Introduction tidyverse
- Import de données
- Manipuler des données
- Export de données

Programme

- **Rappel séance 1**
- Introduction tidyverse
- Import de données
- Manipuler des données
- Export de données

Les bonnes pratiques

- Créer un projet pour une meilleure organisation, reproductibilité et portabilité.
- Créer un script puisque la console ne sauvegarde pas les éléments.
- Lorsqu'on nomme un script ou un fichier, il ne faut pas utiliser d'accent ou d'espace.
- Attention au nommage de vos objets dans le script lorsque vous utilisez l'assignation <- il est possible d'écraser des fonctions (Ex : on ne fait pas mean <- « Hello » sinon on écrase la fonction permettant de calculer la moyenne).
- En plus de l'installation d'un package, il est nécessaire de le charger en début du script en utilisant library(nom_du_package)
- Afin de rendre vos codes compréhensibles facilement : pensez à le commenter avec # et à faire des sections avec ---- ou ####

L’assignation

- Raccourci clavier (Windows/Linux) pour insérer l’opérateur d’assignation : alt+-

- Permet de stocker un résultat pour le réutiliser plus tard dans le script .

Exemple :

```
monobjet <- 1+1
monobjet2 <- monobjet + 2
```

- Exécution d’une ligne de code dans le volet Editeur : Ctrl + entrée

- Autocomplétion : Tab

Les types d'objets et leurs différentes natures

Les types d'objet vus lors de la 1ère séance d'initiation

- Les vecteurs

Ex : exVec <- c(2007, 95, 200)

- Les facteurs

Ex : exFac <- factor(c(« Femme », « Homme », « Femme », « Femme », « Homme »))

- Les dataframes

Ex: exDF <- data.frame(nom = c(«Aurélie», « Joséphin », « Léa »), pointure = c(37, 41, 39))

Grand type	Type	Description	Exemple
numeric	integer	nombres entiers	10
	double	nombres réels	10,56
character	character	Chaîne de charactère	"Hello Word"
Logical/boolean	logical ou bolean	Vrai/ faux/manquant	TRUE/FALSE/NA

L’indexation

L'indexation permet d'intervenir dans un vecteur pour transformer, extraire, supprimer ou ajouter des éléments

Par position

```
> vNum[2]
[1] 38
```

```
> df[1,3]
[1] 41.5
```

Par condition

```
> vNum[vNum==38]
[1] 38
```

Les fonctions de base

Une fonction R est un code qui produit un certain résultat lorsqu'il est exécuté.

Lors de la séance précédente nous avons vu des fonctions de base :

- pour explorer la structure et le contenu de ses données

```
> str(df)
'data.frame':   3 obs. of  3 variables:
 $ NOM     : chr  "Joséphin" "Léa" "Aurélie"
 $ FEMME   : logi  FALSE TRUE TRUE
 $ POINTURE: num  41.5 38 37
```

```
> nrow(df)
[1] 3
```

- pour explorer et décrire des données quantitatives

```
> summary(vNum)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   25.75   50.50   50.50   75.25  100.00
```

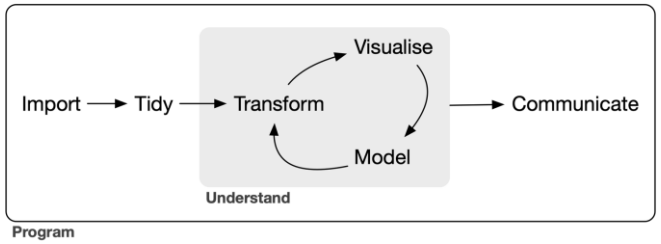
```
> # par défaut, quartiles
> quantile(vNum)
    0%    25%    50%    75%   100%
  1.00  25.75  50.50  75.25 100.00
```

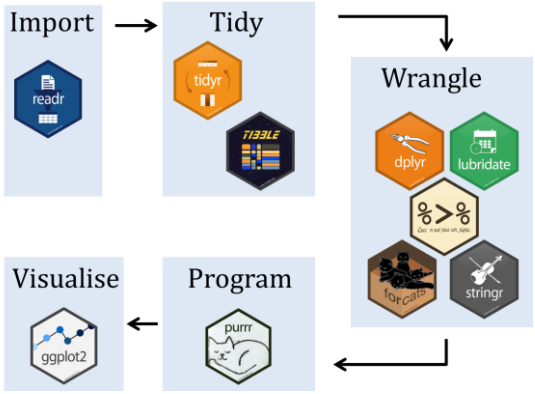
Programme

- Rappel séance 1
- **Introduction tidyverse**
- Import de données
- Manipuler des données
- Export de données

Le tidyverse

Tidy verse, « *univers rangé* » : Ensemble de packages R fait pour fonctionner ensemble (même logique, même grammaire) pour l'ensemble de la chaîne de traitement de la donnée





Tidy data (« *donnée propre* »), concept d'Hadley Wickham : modèle d'organisation des données pour faciliter le nettoyage, structurer la donnée pour faciliter son exploitation

variables
chaque variable est une colonne

observations
chaque observation est une ligne

values
chaque valeur est dans une cellule

Livre de ressource par Wickham et Grolemund

Le tibble

Tibble est un dataframe avec plus de fonctionnalités :

- affichage plus complet (type des colonnes, dimension)
- type de variables ne sont pas modifiées automatiquement au chargement
- pas de nom de lignes (ne conserve que des numéro de ligne)
- colonnes ne sont pas renommées automatiquement (sauf si nom identique)

```
> penguins
# A tibble: 333 x 8
   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>  <int>
 1 Adelie  Torgersen           39.1          18.7               181        3750 male    2007
 2 Adelie  Torgersen           39.5          17.4               186        3800 female  2007
 3 Adelie  Torgersen           40.3          18                 195        3250 female  2007
 4 Adelie  Torgersen           36.7          19.3               193        3450 female  2007
 5 Adelie  Torgersen           39.3          20.6               190        3650 male    2007
 6 Adelie  Torgersen           38.9          17.8               181        3625 female  2007
 7 Adelie  Torgersen           39.2          19.6               195        4675 male    2007
 8 Adelie  Torgersen           41.1          17.6               182        3200 female  2007
 9 Adelie  Torgersen           38.6          21.2               191        3800 male    2007
10 Adelie  Torgersen           34.6          21.1               198        4400 male    2007
# ... with 323 more rows
```

```
> as.data.frame(penguins)
    species    island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g    sex year
1    Adelie Torgersen           39.1          18.7               181        3750   male 2007
2    Adelie Torgersen           39.5          17.4               186        3800 female 2007
3    Adelie Torgersen           40.3          18.0               195        3250 female 2007
4    Adelie Torgersen           36.7          19.3               193        3450 female 2007
5    Adelie Torgersen           39.3          20.6               190        3650   male 2007
6    Adelie Torgersen           38.9          17.8               181        3625 female 2007
7    Adelie Torgersen           39.2          19.6               195        4675   male 2007
8    Adelie Torgersen           41.1          17.6               182        3200 female 2007
9    Adelie Torgersen           38.6          21.2               191        3800   male 2007
10   Adelie Torgersen           34.6          21.1               198        4400   male 2007
11   Adelie Torgersen           36.6          17.8               185        3700 female 2007
                                                [...]
123  Adelie Torgersen           39.0          17.1               191        3050 female 2009
124  Adelie Torgersen           44.1          18.0               210        4000   male 2009
125  Adelie Torgersen           38.5          17.9               190        3325 female 2009
 [ reached 'max' / getOption("max.print") -- omitted 208 rows ]
```

Les fonctions du Tidyverse peuvent se faire avec des data frames mais elles retournent des tibbles.

Pour convertir en tibble : as_tibble()
Pour convertir en data frame : as.data.frame()

Le pipe

Pipe (%>%) permet de :

- structurer des séquences d'opérations,
- minimiser la création d'objets intermédiaire
- faciliter la lecture en évitant le *nesting* (encastrement des fonctions)

Il se lit de gauche à droite, le résultat de la première opération est donnée via le pipe à la suivante

```
> x <- c(1, 2, 3, 4, 5, 6)
> mean(x)
[1] 3.5
```

```
> x <- c(1, 2, 3, 4, 5, 6)
> x %>% mean()
[1] 3.5
```

Plus intuitif que l'encastrement des fonctions qui se lit de droite à gauche.
Bonne pratique : aller à la ligne après chaque pipe (%>%)

```
>  round(exp(diff(log(x))), 1)
[1] 2.0 1.5 1.3 1.2 1.2
```

```
>  x %>%
+    log() %>%
+    diff() %>%
+    exp() %>%
+    round(1)
[1] 2.0 1.5 1.3 1.2 1.2
```

Dplyr

Dplyr est le package dédié à la manipulation des données. Les verbes principaux sont :

- select() pour sélectionner des variables par leur nom

```
lieuxTournage <- data %>%
  select(-c("Coordonnée en X", "Coordonnée en Y", "geo_shape", "geo_point_2d"))
```

- filter() pour filtrer des observations selon leur valeur.

```
emily <- data %>%
  filter(TITRE == "Emily in Paris")
```

- arrange() pour changer l'odre des colonnes.

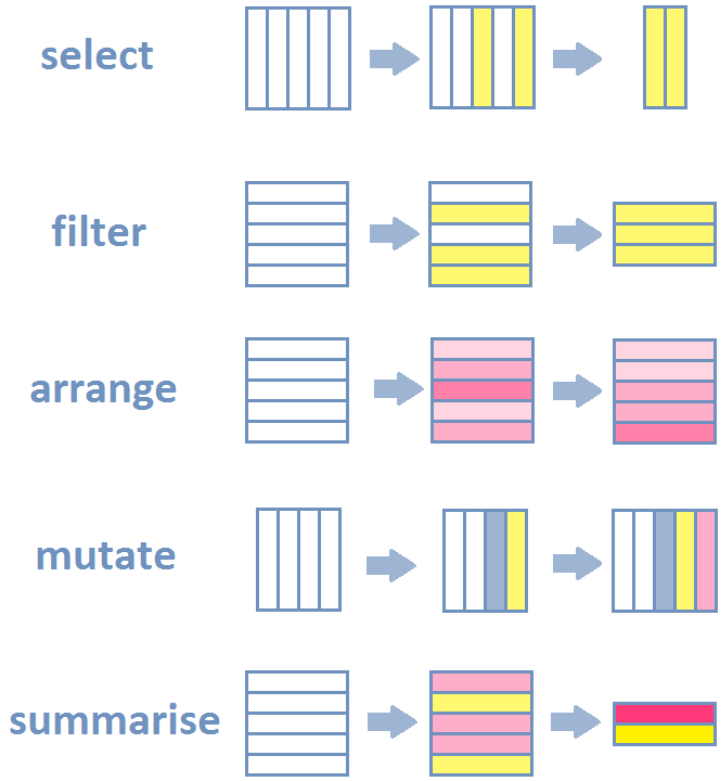
```
tournageMax <- lieuxTournage %>%
  arrange(desc(occurence))
```

- mutate() pour modifier ou ajouter de nouvelles variables

```
lieuxTournage <- lieuxTournage %>%
  mutate(TYPE = as.factor(TYPE))
```

- summarise() pour réduire les observations (moyenne, somme).

```
nbreTournage <- lieuxTournage %>%
  summarise(N_TOURNAGE = n())
```



Source

Dplyr

Il existe d'autres verbes :

- group_by() pour grouper les observations selon une variable
- count() pour compter les observations

```
(real <- lieuxTournage %>%
  group_by(REAL) %>%
  count())
```

- rename() pour changer le nom des variables

```
lieuxTournage <- lieuxTournage %>%
  rename(ID = `Identifiant du lieu`)
```

Avec le pipe, on peut facilement nettoyer et manipuler un jeu de données en ne créant qu'un seul objet

```
tournageMax <- lieuxTournage %>%
  group_by(TITRE) %>%
  count(name = "occurence") %>%
  arrange(desc(occurence)) %>%
  ungroup() %>%
  slice(1:10)
```

Feuille de triche de Dplyr

Différence Dplyr et Rbase

Plusieurs manière d'arriver à la même fin.

Pour sélectionner une variable :

Tidyverse

```
> select(penguins, species, island)
# A tibble: 344 x 2
   species island
   <fct>   <fct>
 1 Adelie  Torgersen
 2 Adelie  Torgersen
 3 Adelie  Torgersen
 4 Adelie  Torgersen
 5 Adelie  Torgersen
 6 Adelie  Torgersen
 7 Adelie  Torgersen
 8 Adelie  Torgersen
 9 Adelie  Torgersen
10 Adelie  Torgersen
# ... with 334 more rows
```

Rbase

```
> penguins[,c("species", "island")]
# A tibble: 344 x 2
   species island
   <fct>   <fct>
 1 Adelie  Torgersen
 2 Adelie  Torgersen
 3 Adelie  Torgersen
 4 Adelie  Torgersen
 5 Adelie  Torgersen
 6 Adelie  Torgersen
 7 Adelie  Torgersen
 8 Adelie  Torgersen
 9 Adelie  Torgersen
10 Adelie  Torgersen
# ... with 334 more rows
```

Pour filtrer une observation selon une condition :

```
> filter(penguins, flipper_length_mm == 181)
> penguins[penguins$flipper_length_mm == 181,]
```

Tidyverse
Rbase

```
# A tibble: 7 x 9
  species   island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex      year body_mass_kg
  <fct>     <fct>              <dbl>         <dbl>             <int>       <int> <fct>   <int>        <dbl>
1 Adelie    Torgersen           39.1          18.7               181        3750 male     2007         3.75
2 Adelie    Torgersen           38.9          17.8               181        3625 female   2007         3.62
3 Adelie    Dream               37.6          19.3               181        3300 female   2007         3.3
4 Adelie    Biscoe              36.5          16.6               181        2850 female   2008         2.85
5 Adelie    Biscoe              38.1          17                 181        3175 female   2009         3.18
6 Chinstrap Dream               58            17.8               181        3700 female   2007         3.7
7 Chinstrap Dream               42.4          17.3               181        3600 female   2007         3.6
```

Pour la création (colonne, variable) : pas besoin de créer la nouvelle colonne en dur

Tidyverse

```
> mutate(penguins, body_mass_kg = body_mass_g/1000)
# A tibble: 344 x 9
   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex      year body_mass_kg
   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>   <int>        <dbl>
 1 Adelie  Torgersen           39.1          18.7               181        3750 male     2007         3.75
 2 Adelie  Torgersen           39.5          17.4               186        3800 female   2007         3.8
 3 Adelie  Torgersen           40.3          18                 195        3250 female   2007         3.25
 4 Adelie  Torgersen           NA            NA                  NA          NA NA       2007        NA
 5 Adelie  Torgersen           36.7          19.3               193        3450 female   2007         3.45
 6 Adelie  Torgersen           39.3          20.6               190        3650 male     2007         3.65
 7 Adelie  Torgersen           38.9          17.8               181        3625 female   2007         3.62
 8 Adelie  Torgersen           39.2          19.6               195        4675 male     2007         4.68
 9 Adelie  Torgersen           34.1          18.1               193        3475 NA       2007         3.48
10 Adelie  Torgersen           42            20.2               190        4250 NA       2007         4.25
# ... with 334 more rows
```

Rbase

```
> penguins$body_mass_kg <- penguins$body_mass_g/1000
> head(penguins)
# A tibble: 6 x 9
  species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex      year body_mass_kg
  <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>   <int>        <dbl>
1 Adelie  Torgersen           39.1          18.7               181        3750 male     2007         3.75
2 Adelie  Torgersen           39.5          17.4               186        3800 female   2007         3.8
3 Adelie  Torgersen           40.3          18                 195        3250 female   2007         3.25
4 Adelie  Torgersen           NA            NA                  NA          NA NA       2007        NA
5 Adelie  Torgersen           36.7          19.3               193        3450 female   2007         3.45
6 Adelie  Torgersen           39.3          20.6               190        3650 male     2007         3.65
```

[D'autres exemples d'équivalences Tidyverse / Rbase]

StringR

StringR est le package dédié aux chaînes de caractères (*string*).

On peut facilement :

- détecter et extraite des patterns (str_detect(), str_which(), str_subset(), str_extract())

```
> str_subset(real$REAL, "Alexandra")
[1] "Alexandra LECLERE" "Alexandra Leclère"
```

- gérer les longueur (str_length(), str_pad(), str_trunc() et str_trim())

```
lieuxTournage %>%
  mutate(TAILLE = str_length(TITRE)) %>%
```

- modifier les chaînes de caractères (str_replace(), str_to_lower(), str_to_upper(), str_to_title())

```
lieuxTournage <- lieuxTournage %>%
  mutate(TITRE = str_to_lower(TITRE),
         REAL = str_to_title(REAL),
         PROD = str_to_upper(PROD))
```

- Le package regorge de nombreuses autres fonctionnalités : Feuille de trich de StringR

Lubridate

Lubridate est un package facilitant la manipulation du temps à travers des vecteurs avec des types dédiés (date, POSXict...) :

- Facilement en extraire les éléments (le jour, la semaine, le mois, l'année...)

```
> lieuxTournage %>%
+   mutate(JOUR_DEB = wday(DATE_DEB, label = TRUE)) %>%
+   select(DATE_DEB, JOUR_DEB)
# A tibble: 10,766 x 2
   DATE_DEB   JOUR_DEB
   <date>     <ord>
 1 2019-10-17 "jeu\\."
 2 2019-11-05 "mar\\."
 3 2019-11-13 "mer\\."
 4 2021-05-18 "mar\\."
 5 2021-05-25 "mar\\."
 6 2021-07-20 "mar\\."
 7 2019-11-13 "mer\\."
 8 2019-09-20 "ven\\."
 9 2019-08-19 "lun\\."
10 2019-08-21 "mer\\."
# ... with 10,756 more rows
```

- Faire des *maths* entre deux vecteurs de même types (addition, soustraction...)

```
(lieuxTournage <- lieuxTournage %>%
  mutate(DUREE = DATE_FIN - DATE_DEB + 1))
```

```
$ DUREE     : 'difftime' num [1:10766] 1 1 1 1
```

```
DUREE
<drt>
1 da~
1 da~
1 da~
1 da~
1 da~
1 da~
1 da~
1 da~
2 da~
1 da~
1 da~
```

Unité : jour

Et beaucoup d'autres choses : [feuille de triche]

TidyR

TidyR est le package pour organiser (*tidy*) sa donnée avec des fonctions comme :

- drop_na() qui supprime les lignes contenant des valeurs manquantes (NA)

```
> dim(penguins)
[1] 344   9
> penguins <- drop_na(penguins)
> dim(penguins)
[1] 333   9
```

- separate() pour séparer une colonne de texte en plusieurs colonnes et permettant d'utiliser des expressions régulières (*équivalent de str_split() de stringR*).

- pivot_longer() et pivot_wider() pour passer un tableau d'un format court à un format long et inversement

```
> tournageByArrType
# A tibble: 91 x 3
   CODE  TYPE          N_TOURNAGE
   <chr> <fct>              <int>
 1 75001 Long métrage         256
 2 75001 Série TV             304
 3 75001 Série Web             13
 4 75001 Téléfilm              60
 5 75002 Long métrage         157
 6 75002 Série TV              96
 7 75002 Série Web              2
 8 75002 Téléfilm               5
 9 75003 Long métrage         105
10 75003 Série TV              93
# ... with 81 more rows
```

```
> (tournageByArrType_wild <- tournageByArrType %>%
+    pivot_wider(names_from = TYPE,
+                values_from = N_TOURNAGE))
# A tibble: 27 x 5
   CODE  `Long métrage` `Série TV` `Série Web` Téléfilm
   <chr>          <int>      <int>       <int>    <int>
 1 75001            256        304          13       60
 2 75002            157         96           2        5
 3 75003            105         93          15        6
 4 75004            332        221          13       24
 5 75005            349        209          10       33
 6 75006            225        145          10       24
 7 75007            315        212          13       22
 8 75008            346        237          28       43
 9 75009            298        253          10       19
10 75010            363        248          29       37
# ... with 17 more rows
```

[Feuille de triche]

TidyR

?pivot_wider()

Passage d'un tableau long à un tableau court : on diminue le nombre de lignes et on étend l'information sur plusieurs colonnes

data

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

```
pivot_wider(data,
            names_from = "type",     #les nouvelles variables à mettre en colonne
            values_from = "count"    #les valeurs à répartir dans les nouvelles colonnes
            )
```

TidyR

?pivot_longer()

Passage d'un tableau court à un tableau long : on ajoute des lignes et on regroupe l'information contenue dans plusieurs colonnes dans deux variables

data

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

```
pivot_longer(data,
             cols = 2:3,              #l'info à réduire en une seule variable
             names_to = "year" ,      #facultatif
             values_to = "cases"      #facultatif
             )
```

Programme

- Rappel séance 1
- Introduction tidyverse
- **Import de données**
- Manipuler des données
- Export de données

Téléchargement du diaporama et des supports de la séance

Pour les exercices pratiques, les données sont à télécharger à l'adresse suivante :

https://elementr.gitpages.huma-num.fr/website/posts/seance3.html

Import de données

Importer une table au format csv avec la librairie readr

Deux solutions :

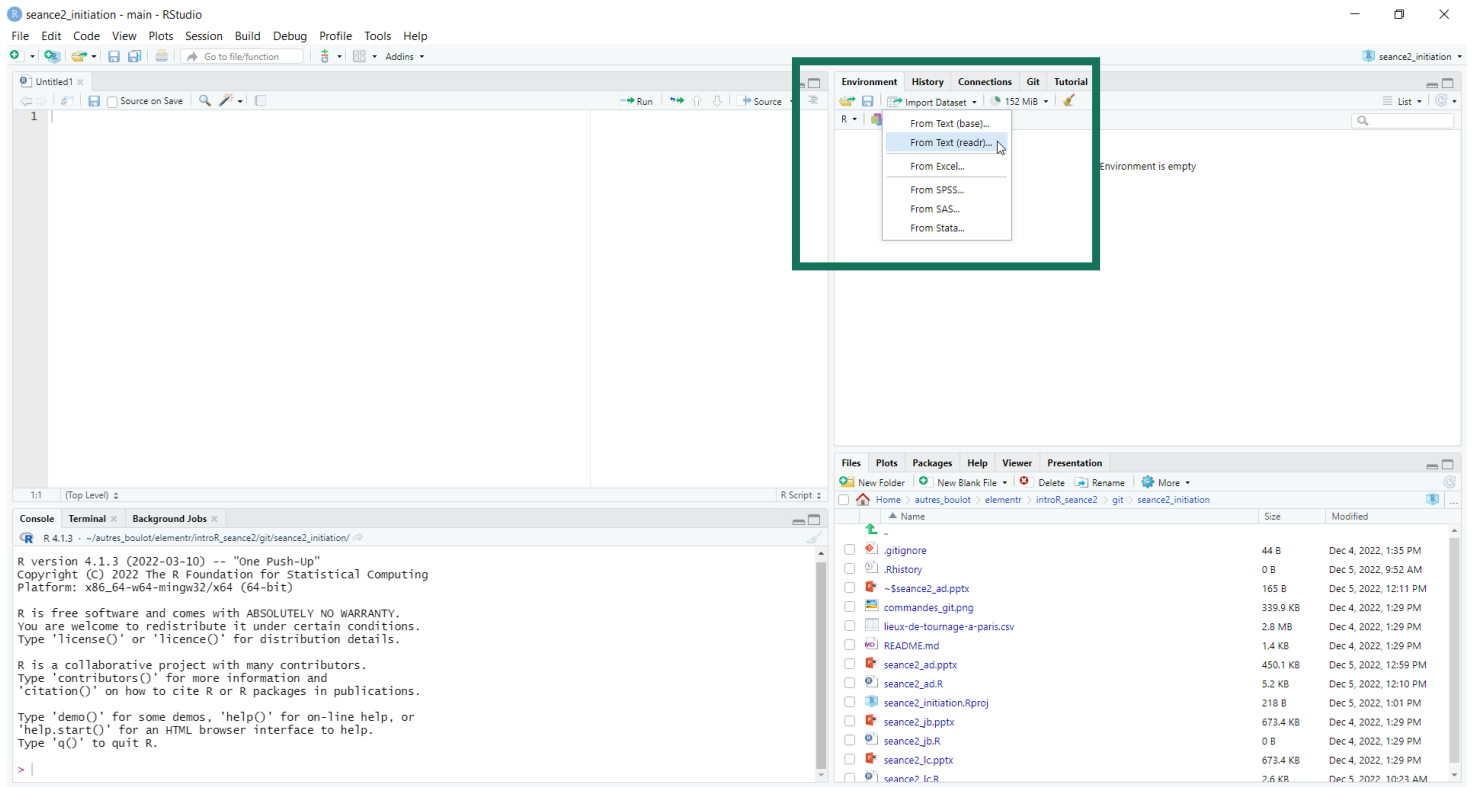
- Via la fenêtre d'import de RStudio
- En ligne de code

Import de données

Importer une table au format csv avec la librairie readr

Deux solutions :

- Via la fenêtre d'import de RStudio
- En ligne de code

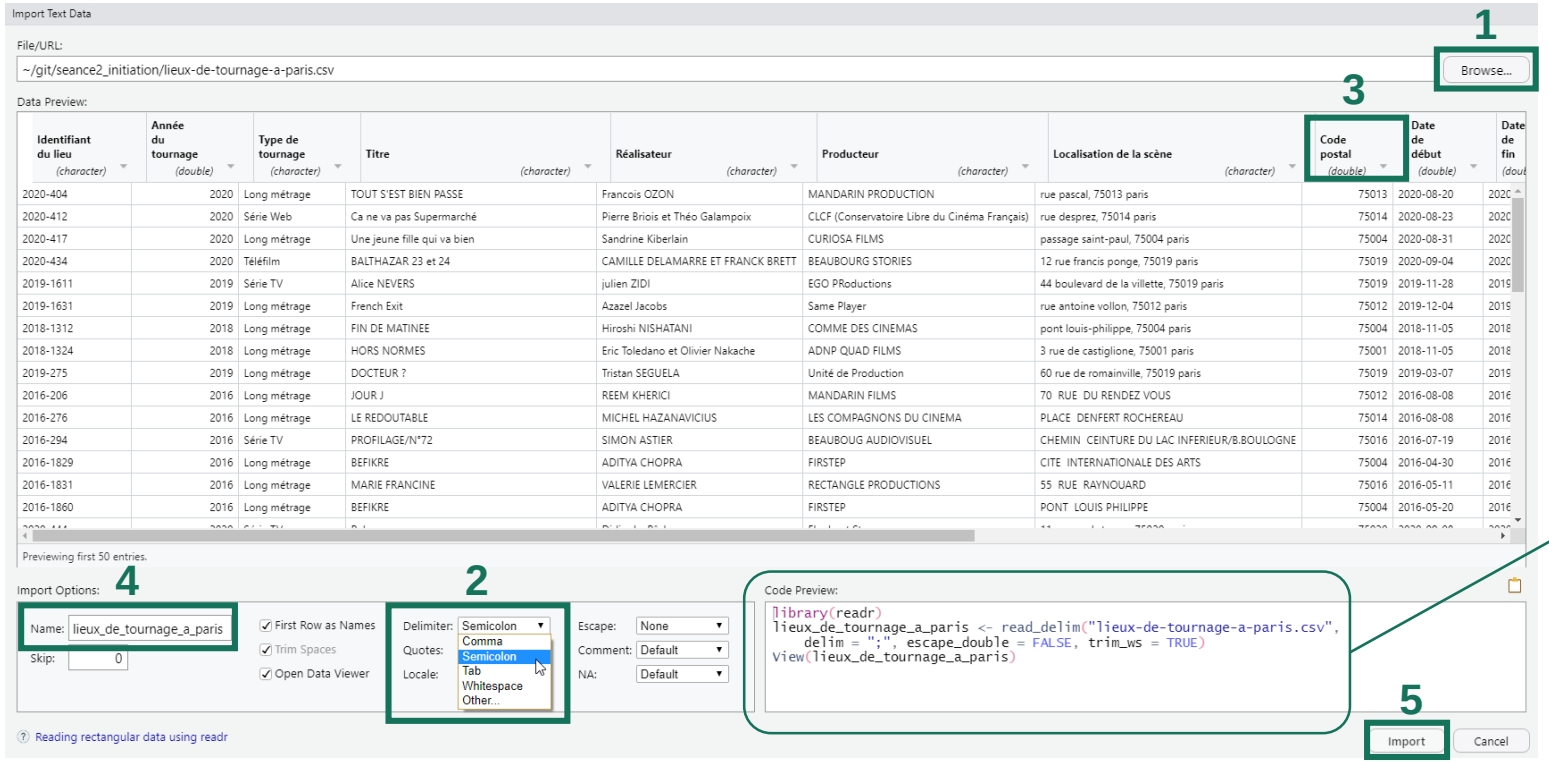


Import de données

Importer une table au format csv avec la librairie readr

Deux solutions :

- Via la fenêtre d'import de RStudio



1. Chercher le csv dans ses dossiers avec le bouton 'Browse'
2. Préciser le séparateur de champs. Dans notre cas : Semicolon
3. Modifier le type des variables. Par exemple Code postal en character
4. Modifier le nom de l'objet
5. Cliquer sur 'Import'

L'outil génère du code qui se met à jour à chaque clic-bouton. Il est possible de copier/coller ce code dans son script pour une meilleure reproductibilité

Import de données

Importer une table au format csv avec la librairie readr

Deux solutions :

- Via la fenêtre d'import de RStudio
- En ligne de code avec la fonction **read_csv2()**

```
library(readr)
data <- read_csv2(file = "lieux-de-tournage-a-paris.csv")
```

La structure de la table s'affichent dans la console

```
i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
Rows: 10766 Columns: 14

-- Column specification --------------------------------------------------------
Delimiter: ";"
chr  (7): Identifiant du lieu, Type de tournage, Titre, Réalisateur, Producteur, Localisation de la scène, geo...
dbl  (2): Année du tournage, Code postal
date (2): Date de début, Date de fin

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Import de données

Importer une table au format csv avec la librairie readr

Deux solutions :

- Via la fenêtre d'import de RStudio
- En ligne de code avec la fonction **read_csv2()**

```
data <- read_csv2(file = "lieux-de-tournage-a-paris.csv",
                  col_types = cols(`Code postal` = col_character()),
                  show_col_types = TRUE)
```

Les codes postaux ne sont plus en type 'double'

```
i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
Rows: 10766 Columns: 14

-- Column specification --------------------------------------------------------------------------------------------
Delimiter: ";"
chr  (8): Identifiant du lieu, Type de tournage, Titre, Réalisateur, Producteur, Localisation de la scène, Cod...
dbl  (1): Année du tournage
date (2): Date de début, Date de fin

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Programme

- Rappel séance 1
- Intro tydiverse
- Import de données
- **Manipuler des données**
- Export de données

Cas pratique : les lieux de tournage à Paris

La ville de Paris publie en open data les lieux de tournage de scène en extérieur à Paris depuis 2016.
Afin de manipuler les notions précédentes, nous vous avons preparé un script qui contient :

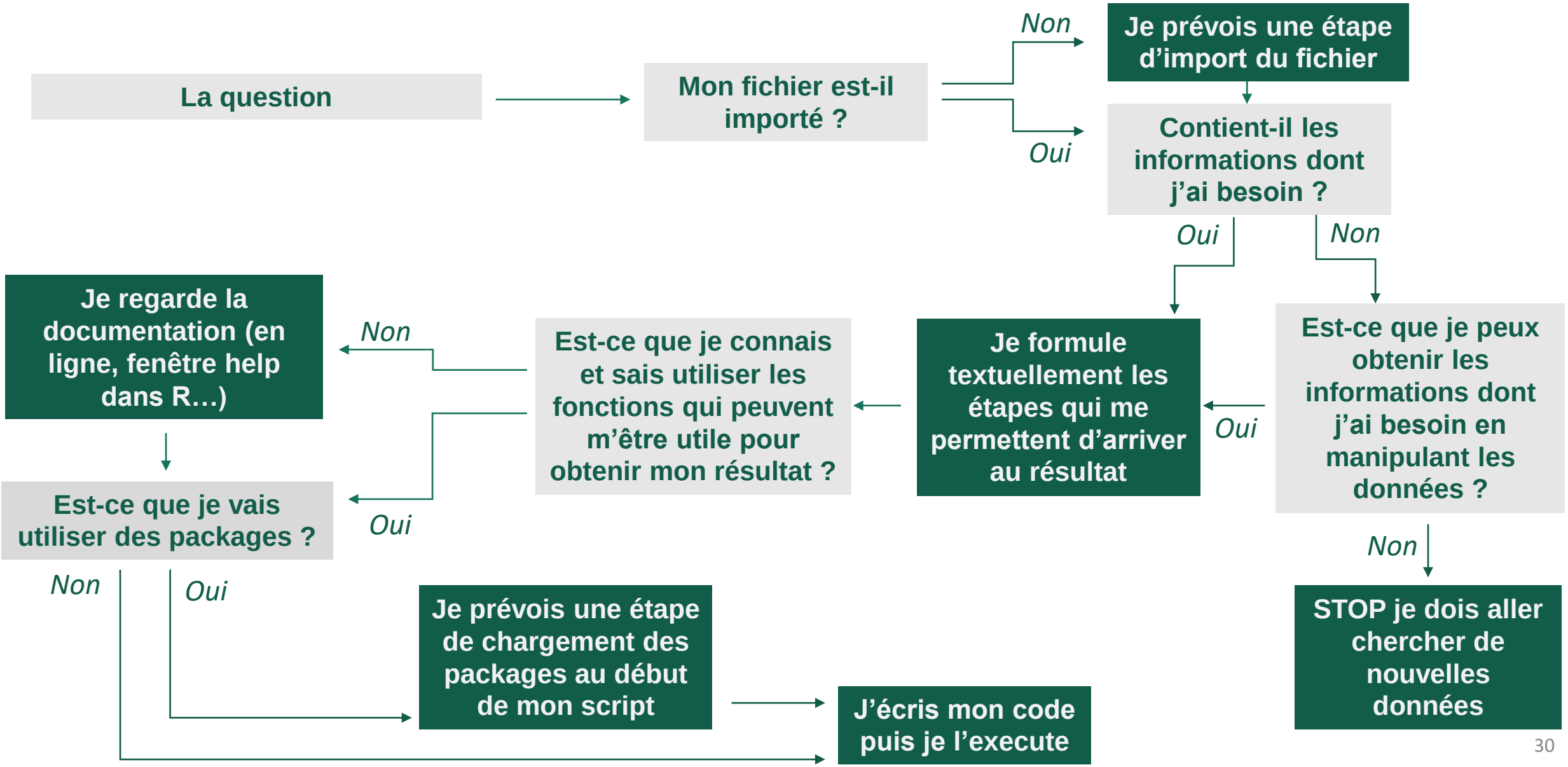
- Des questions auxquelles il faut répondre en retrouvant la ligne de code

Ex : Combien de tournages ont eu lieu dans le 15 ème arrondissement ?

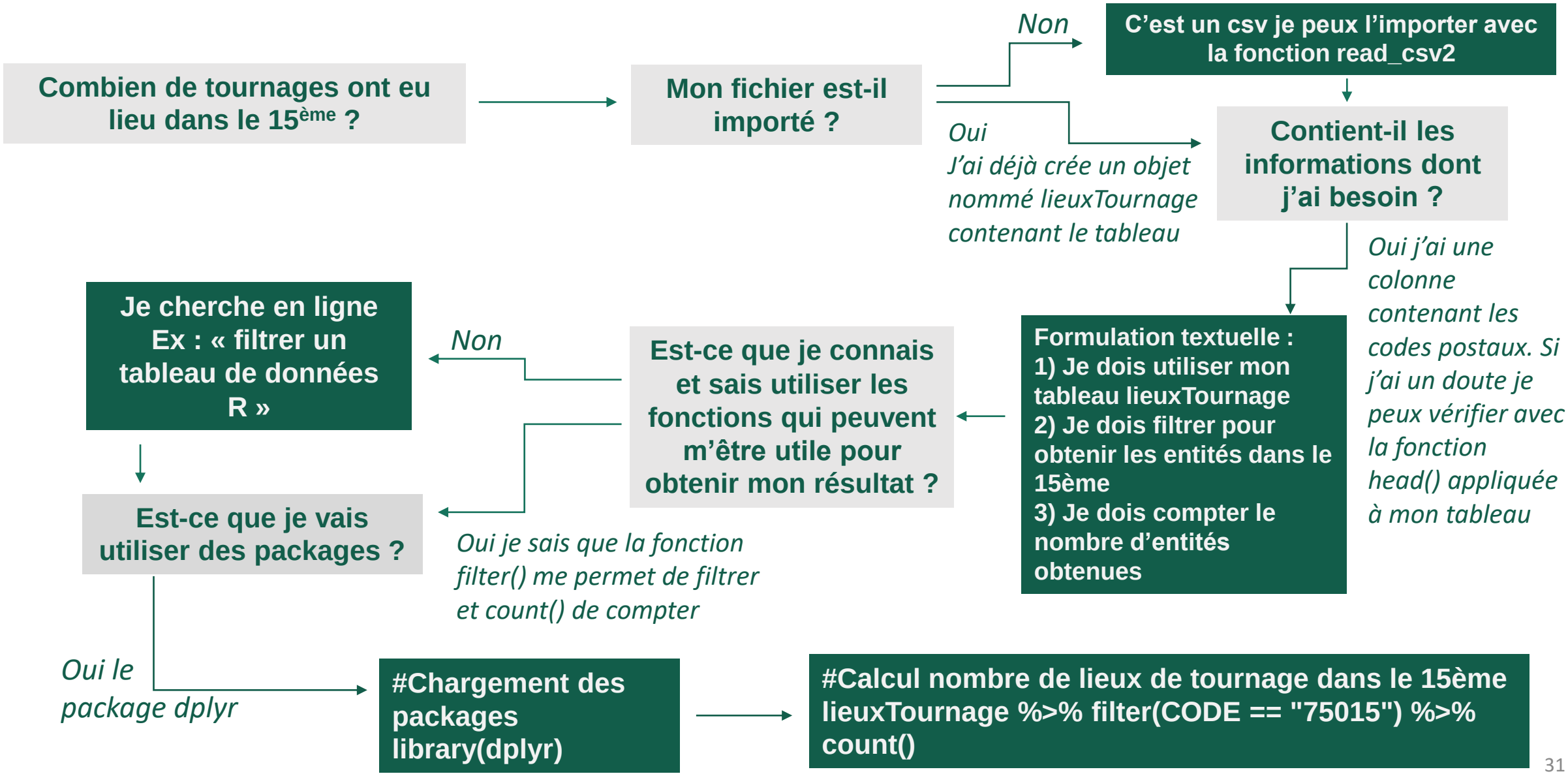
- Des lignes de code pour lesquels il faut retrouver le commentaire

```
#
data <- read.csv2("lieux-de-tournage-a-paris.csv", fileEncoding = "UTF-8")
```

Processus d'aide pour répondre aux questions



Exemple : Combien de tournages ont eu lieu dans le 15ème ?



Et si on traduisait en français ?

Si l'on reprend la ligne de code précédent :

```
lieuxTournage %>% filter(CODE == "75015") %>% count()
```

Je peux la « traduire » en français :

lieuxTournage : Je prends mon tableau « lieu de tournage »

%>% : et puis

filter(CODE == "75015") : je filtre ce tableau pour garder uniquement les entités dont la colonne CODE est égale à 75015

%>% : et puis

count() : je compte ces entités

Programme

- Rappel séance 1
- Introduction tidyverse
- Import de données
- Manipuler des données
- **Export de données**

Export de données

Exporter une table créé dans une session pour enregistrer, partager ou réutiliser ses données :

- Au format .csv avec la librairie readr
- Au format .rds, propre à R

Export de données

Exporter une table créé dans une session pour enregistrer, partager ou réutiliser ses données :

- Au format .csv avec la librairie readr
- Au format .rds, propre à R

L'écriture se fait avec la fonction **write_csv2()**

```
write_csv2(x = newData,
           file = "lieux_de_tournage_a_paris_clean.csv")
```

Export de données

Exporter une table créée dans une session pour enregistrer, partager ou réutiliser ses données :

- Au format .csv avec la librairie readr
- Au format .rds, propre à R

L'écriture se fait avec la fonction **saveRDS()**

```
saveRDS(object = newData,
        file = "lieux_de_tournage_a_paris_clean.rds")
```

La lecture se fera avec la fonction **readRDS()**

```
data <- readRDS(file = "lieux_de_tournage_a_paris_clean.rds")
```