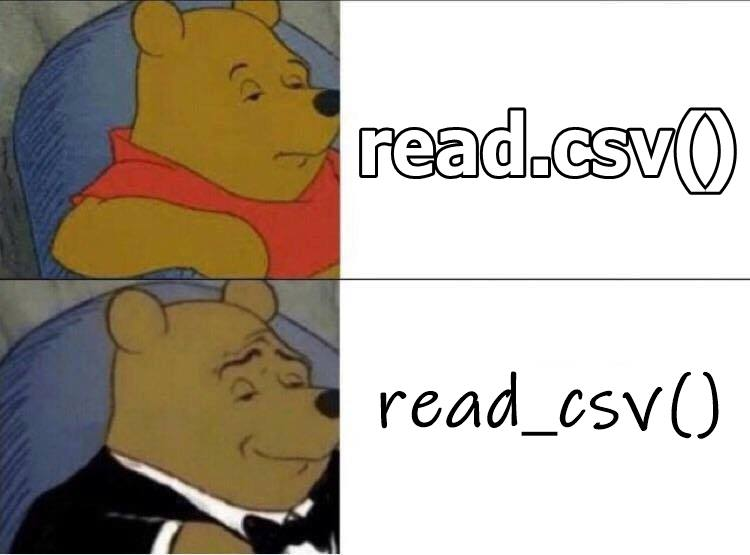
Chapitre 2 : Manipulation des données

*Traduit par Brice Beffara-Bret.*



2.1 Objectifs d’apprentissage

1. Comprendre l’utilisation des différentes types de données de base
2. Comprendre et utiliser les différents types de conteneurs de base (listes, vecteurs)
3. Créer des vecteurs et les enregistrer comme variables
4. Comprendre les opérations sur les vecteurs
5. Créer un tableau de données
6. Importer des données à partir de fichier CSV et Excel

2.2 Ressources

Chapitre 11: [Importation des données](https://r4ds.had.co.nz/data-import.html) de [*R for Data Science*](https://r4ds.had.co.nz/) (EN)

[Antisèche RStudio sur l’importation des données](https://github.com/rstudio/cheatsheets/raw/master/source/pdfs/data-import-cheatsheet.pdf) (EN)

[Fichier de données *Scottish babynames*](https://www.nrscotland.gov.uk/files//statistics/babies-first-names-full-list/summary-records/babies-names16-all-names-years.csv) (EN)

[Développer une analyse dans R/RStudio: *Scottish babynames* (1/2)](https://www.youtube.com/watch?v=lAaVPMcMs1w)(EN)

[Développer une analyse dans R/RStudio: *Scottish babynames* (2/2)](https://www.youtube.com/watch?v=lzdTHCcClqo)(EN)

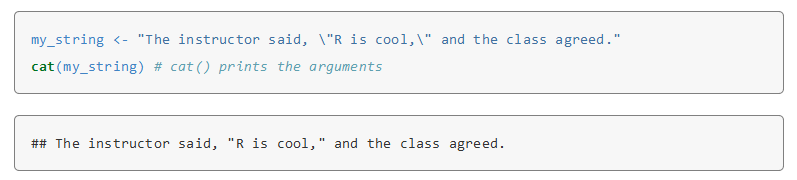
2.3 Les types de données de base

Il y a 5 principaux types de données de base dans R (en réalité il en existe plus que cela mais ces cinq sont ceux qu’il est nécessaire de connaître pour ce cours).

|  |  |  |
| --- | --- | --- |
| Type (EN) | description | exemple |
| double | valeur en virgule flottante | .333337 |
| integer | entier | -1, 0, 1 |
| numeric | N’importe quel nombre réel | 1, .5, -.222 |
| boolean | assertion vrai ou faux | VRAI, FAUX |
| character | texte, chaîne de caractères | "hello world", ‘salut’ |

Il existe aussi un type spécifique de données appelé **facteur**. Ce type de variable vous donnera probablement mal à la tête tôt ou tard mais nous pouvons continuer sans pour l’instant.

Les chaînes de caractères peuvent inclure à peu près n’importe quoi, y compris de guillemets. Cependant, si vous voulez inclure un guillemet il faut l’”échapper” en utilisant l’antislash :



Notez que si vous entrez juste un nombre simple comme **10** par exemple, il est stocké en tant que donnée “double”, même s’il ne comporte pas de décimale. Si vous voulez que ce nombre soit de type “integer” il faut lui ajouter **L** en suffixe (10L).

Si vous souhaitez connaître le type de donnée d’un élément utilisez la fonction “**class**”.



2.4 Les types de conteneurs de base

2.4.1 Les vecteurs

Les vecteurs font partie des structures de données clefs dans R. Dans R, un vecteur est comme un vecteur en mathématiques : un ensemble d'éléments ordonnés. Tous les éléments du vecteur doivent être du même type de données (numeric, character, factor). Vous pouvez créer un vecteur en insérant les éléments dans **c(...)** comme montré ci-dessous.



**2.4.1.1 Sélectionner des valeurs à partir d’un vecteur**

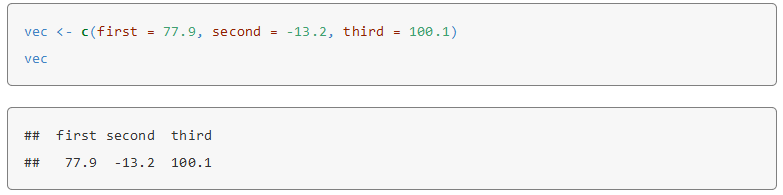
Souvenez-vous de la dernière leçon. Vous pouvez créer une vecteur en utilisant l’opérateur “**c()**” (il s’agit de la manière la plus simple de le faire, mais vous pouvez aussi utiliser la fonction “**vecteur()**”). Si on souhaitait choisir des valeurs spécifiques au sein d’un vecteur à partir de leur position, on pourrait construire un vecteur de nombres de la manière suivante :



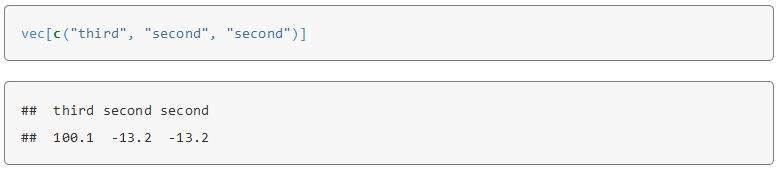
Et ensuite les extraire en utilisant l’opérateur “**[ ]**” (qui est l’opérateur “*extraction*”) sur le vecteur “**LETTERS**” déjà intégré de base dans R.



Vous pouvez aussi créer des vecteurs “nommés” où chaque élément a un nom. Par exemple :

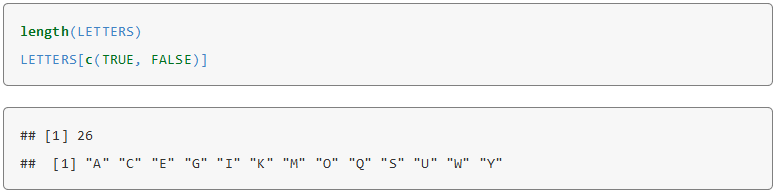


On peut ensuite accéder aux éléments par leur nom en utilisant un vecteur de type “character” à l’intérieur des crochets. On peut les mettre dans l’ordre souhaité et éventuellement les répéter :



On peut obtenir le vecteur de noms en utilisant la fonction “**names()**”, et nous pouvons les spécifier ou les changer en utilisant une approche du type **names(vec2) <- c("n1", "n2", "n3")**.

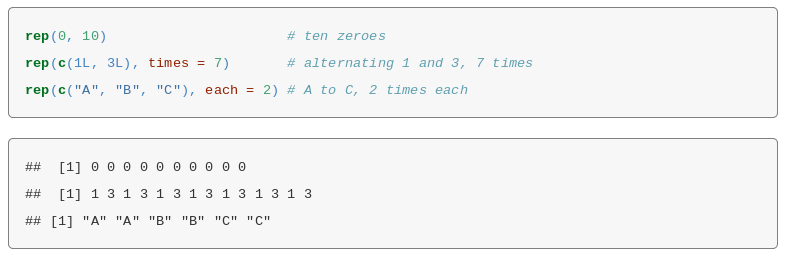
Une autre manière d’accéder aux éléments est d’utiliser un vecteur de type “logique” à l’intérieur des crochets. Cela va extraire les éléments du vecteur “logique” pour lesquels les éléments correspondant du vecteur sont **TRUE**. Si le vecteur “logique” n’a pas la même longueur que le vecteur original, il se répètera. Vous pouvez trouver la longueur du vecteur en utilisant la fonction “**length()**”.



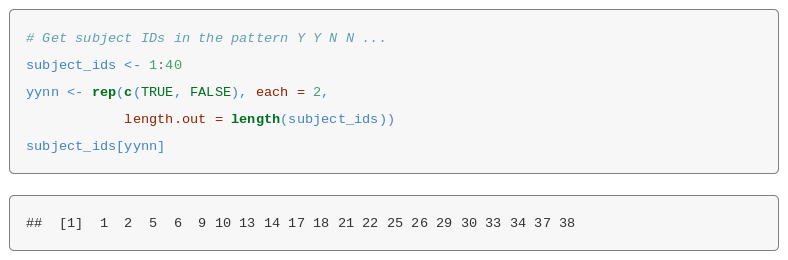
**2.4.1.2 Répéter des séquences**

Voici quelques astuces utiles pour éviter de rentrer les valeurs manuellement lors de la création d’un vecteur. Souvenez-vous que dans la commande **“x:y”** l’opérateur **“:”** renvoie une séquence d’integers de **“x:y”**.

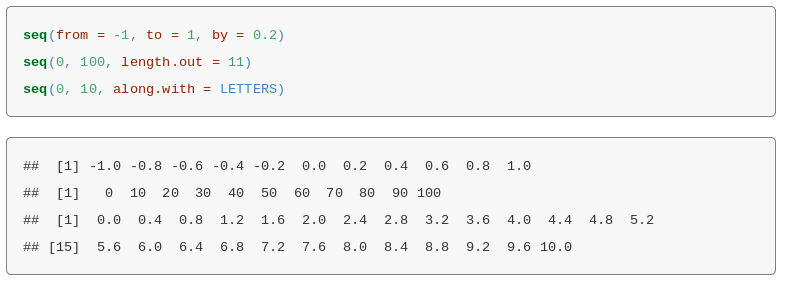
Comment faire si vous voulez répéter un vecteur plusieurs fois ? Vous pourriez les entrer manuellement (ce qui serait relativement pénible) ou bien utiliser la fonction **“rep()”** qui peut répéter des facteurs de différentes manières.



La fonction **“rep”** est utile pour créer des vecteurs de valeur logiques (**VRAI** / **FAUX** ou **1** / **0**) pour sélectionner des valeurs à partir d’un autre vecteur.



Qu’en serait-il si vous vouliez créer une séquence mais à partir d’éléments qui ne sont pas des integer ? Vous pourriez utiliser la fonction **“seq()”**. Regardez les exemples ci-dessous et tentez de déduire ce que font les arguments.

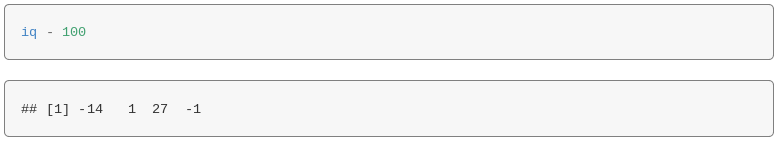


**2.4.1.3 Opérations vectorielles**

R effectue des calculs sur des vecteurs d’une manière spéciale. Prenons un exemple en utilisant les z-scores. Un z-score est un score de déviation (un score moins une moyenne) divisé par un écart-type. Disons que nous disposons d’un ensemble de quatre scores de QI.



Si on veut soustraire la moyenne à partir de ces quatre scores, on utilise simplement le code suivant :

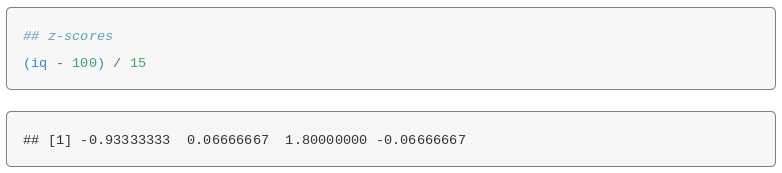


Ce code soustrait 100 de chaque élément du vecteur. R postule automatiquement que c’est ce que vous vouliez faire; c’est ce qu’on appelle une *opération vectorielle* et cela rend possible d’exprimer les opérations de manière plus efficace.

Pour calculer des z-scores on utilise la formule suivante :



Où X sont les scores, 𝜇 la moyenne, et 𝜎 l’écart-type. On peut exprimer cette formule dans R de la manière suivante :



Vous pouvez voir que R a calculé les quatre z-score dans une seule ligne de code. Très efficace !

**2.4.1.4 Exercices**

1. Le vecteur **“letters”** intégré de base dans R contient des lettres de l’alphabet anglais. Utiliser un vecteur d’indexation d’integers pour extraire les lettres composant le mot ‘cat’.
2. LA fonction **“colors”** renvoie tous les noms de couleurs que connaît R. Quelle est la longueur du vecteur renvoyé par cette fonction ? (Utiliser le code pour répondre à cette question)
3. Utiliser la fonction **“runif(1000, 0, 1)”** va renvoyer 1000 nombres à partir d’une distribution uniforme de 0 à 1, ce qui simule les p-valeurs que vous obtiendriez à partir de 1000 expériences dans le cas où l’hypothèse nulle est vraie. Enregistrer ce résultat sous le nom **pvals**. Créer un vecteur logique appelé **is\_sig** étant **VRAI** si le nombre d’éléments de **pvals** correspondant est inférieur à .05, et **FAUX** dans les autres cas (indice: opérations vectorielles de la session précédente), puis utiliser ce vecteur logique pour extraire ces p-valeurs. Enfin, calculer la proportion de ces p-valeurs qui étaient significatives.

2.4.2 Les listes

Rappelez-vous que les vecteurs peuvent seulement contenir des données du même type. Comment faire alors pour stocker des données de types différents ? Pour cela, il faudrait alors utiliser une **liste**. On définit une liste en utilisant la fonction **“list()”**.



You can refer to elements of a list by



2.4.3 Les données tabulaires

La plupart des éléments avec lesquels vous allez travailler dans ce cours seront des *données tabulaires*, c’est à dire, des données agencées en forme de tableau.

Tout comme les listes, les données tabulaires permettent à des données de différents types (chaînes de caractères, integers, logiques, etc.) mais avec la contrainte de conserver le même nombre d’éléments dans chaque “colonne” du tableau (élément de la liste). Le tableau par défaut est appelé “**data.frame”** dans R alors que la version ‘tidtverse’ est appelée **“tibble”**. Les tibbles sont plus facile à manipuler dons nous utiliserons ce type de tableau. Pour en savoir plus à propos des différences entre ces deux structures de données, voir **“vignette(“tibble”)”**.

Les données tabulaires deviendront particulièrement importantes lorsque nous parlerons du rangement des données à la leçon 4, qui consistera en un ensemble de principes simples pour structurer les données.

Si on créer un tibble à partir de rien, on peut utiliser la fonction **“tibble()”**, et entrer les données à l’intérieur. Notez que si on veut qu’une valeur se répète plusieurs fois, il suffit simplement de spécifier un vecteur de un élément ; R va étendre ce vecteur pour remplir le tableau. Toutes les colonnes du tibble doivent avoir la même longueur ou une longueur de 1.

Si on veut utiliser la fonction **“tibble()”**, nous devons charger soit le package tibble soit le package tidyverse (qui lui-même charge tibble en plus d’autres packages). Choisissons la deuxième option.



On peut obtenir les informations à propos des dimensions du tableau en utilisant la fonction **“ncol()”** (nombre de colonnes) **“nrow()”** (nombre de lignes), ou **“dim()”** (un vecteur avec le nombre de lignes et le nombre de colonnes).



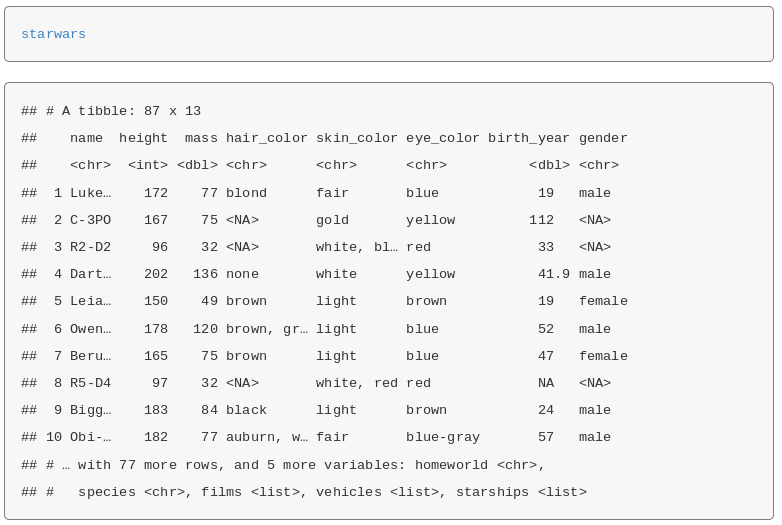
2.4.3.1 Visualiser votre tibble

Toujours, toujours, toujours. Regardez toujours vos données une fois que vous avez créé le tableau. Regardez aussi votre tableau de données à chaque fois que vous transformez votre tibble.

Il y a trois moyen de visualiser votre tibble: **“View()”** [\*NB: ‘V’ majuscule], **“print()”**, et **“glimpse()”**. Notez qu’il est rare d’afficher les données dans un script. Il s’agit plus d’une vérification, et vous devriez réserver cette commande pour la console.

La méthode **“print()”** peut être lancée explicitement mais il est plus courant de l’appeler en entrant simplement le nom de la variable dans une nouvelle ligne. Habituellement, on appelle **“print()”** uniquement si on veut contrôler la manière dont l’information est affichée.

Notez que par défaut, ce n’est pas le tableau entier qui est affichée mais juste les 20 première lignes. Regardons le tableau **“starwars”** inclus par défaut dans le tidyverse.

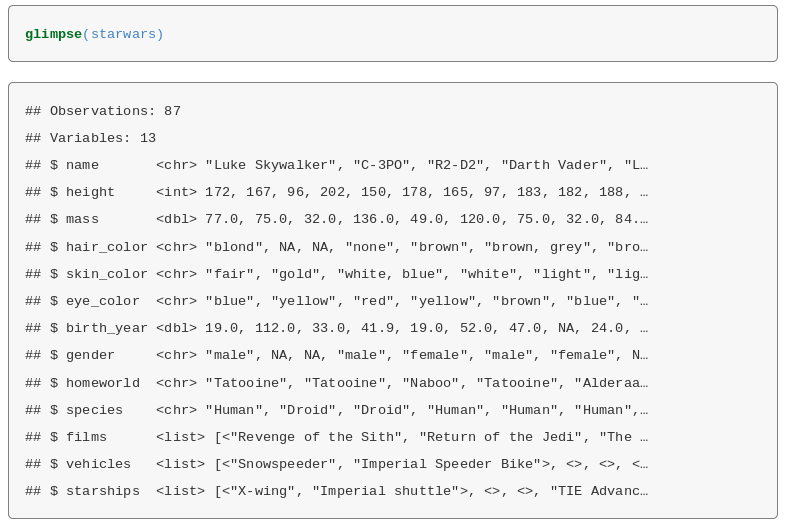


Vous pouvez voir qu’il s’agit d’un tableau à 87 lignes par 13 colonnes, que nous voyons uniquement les 10 premières lignes et 8 premières colonnes.

Si je veux voir les 87 lignes pour quelque raison que ce soir, il faut que j’utilise explicitement la fonction **“print()”**, en spécifiant l’argument **“n”** correspondant au nombre de lignes que je veux voir. Si je veux toutes les lignes, je dois simplement utiliser **“+Inf”**, le symbole pour un nombre “infini” de lignes.

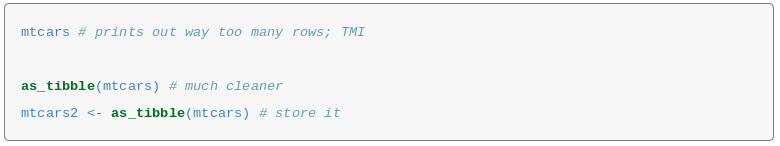


En revanche, on ne peut toujours pas voir toutes les colonnes. Si on veut vraiment les voir, on peut utiliser “glimpse”, qui renvoie une version transposée du tibble.



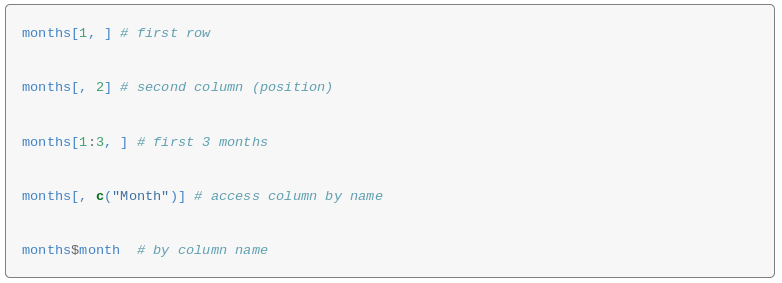
Une autre manière de visualiser le tableau de données est possible sous forme d’une sorte de feuille de calcul plus “graphique”. Cette visualisation est donnée par la commande “View()” (‘V’ majuscule). Cette visualisation peut être utile lorsqu’elle est lancée via la console. En revanche, n’utilisez pas cette commande dans un script car lorsque le script est lancé, une fenêtre pop-up va s’afficher ce qui peut être gênants pour les utilisateur·ices.

Notez que les objets **data.frame** sont affichés d’une manière différente en comparaison aux objets **tibble**. Si vous affichez un objet **data.frame** avec des centaines de milliers de lignes, vous n’aurez pas simplement un aperçu… vous allez spammer votre console avec les lignes de vos données. Si vous voulez transformer votre data.frame en tibble pour que l’affichage soit plus élégant, utilisez simplement la fonction **“as\_tibble()”**.



2.4.3.2 Accéder aux lignes et colonnes

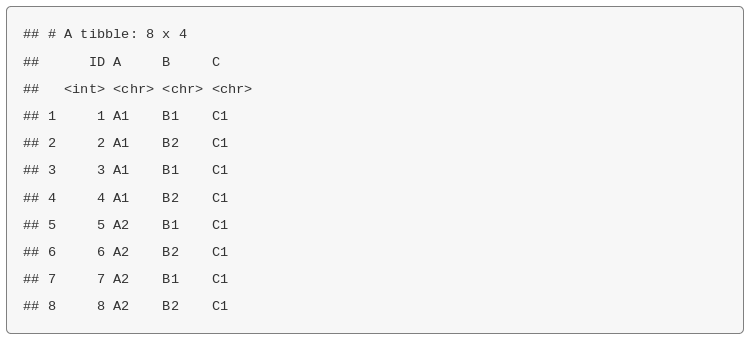
Il y a plusieurs moyens de base dans R pour accéder à des lignes ou colonnes spécifiques. Il est utile de connaître ces méthodes, mais vous allez apprendre des méthodes plus faciles (et plus lisibles) quand on commencera le cours sur la préparation des données. Des exemples des fonctions de base dans R sont proposés ici à titre de référence.



Vous apprendrez les opérations sur les tableaux de données dans les cours sur tidyr et dplyr.

2.4.3.3 Exercices

1. Créer un tibble avec le nom, l’âge, et le sexe de 3 à 5 personnes dont vous connaissez ces informations.
2. Convertissez le jeu de données **iris** inclus par défaut dans R en tibble, et stockez le dans une variable **iris2**.
3. Créez un tibble ayant la structure du tableau ci-dessous, en utilisant un minimum d’entrées au clavier (indice: **“rep()”**). Stockez le dans la variable **my\_tbl**.



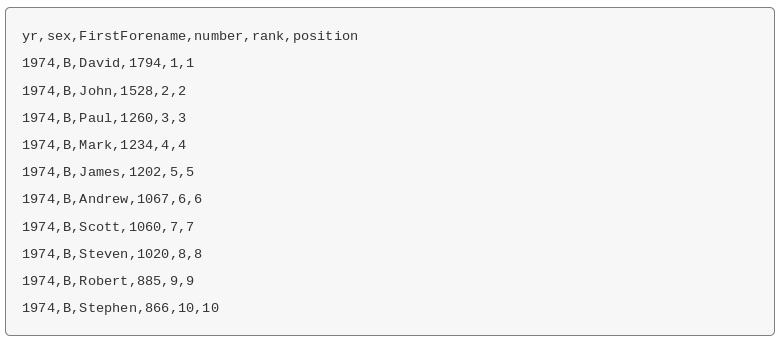
2.5 Importer des données

Il y a plusieurs types de fichiers différents avec lesquels il est possible de travailler lorsque l’on veut analyser des données. Ces différents types de fichiers sont habituellement identifiés par une *extension* de 3 lettres qui suivent un point après le nom du fichier. Voici quelques exemples de différents types de fichiers les et fonctions qu’il faudrait utiliser pour les lire ou bien les écrire.



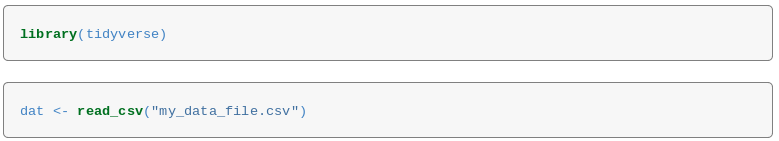
Note : Conformément aux conventions présentées plus haut dans la section des packages additionnels, **“readr::read\_csv()”** fait référence à la fonction **“read\_csv()”** du package **“readr”**, et **“readxl::read\_excel()”** fait référence à la fonction **“read\_excel()”** du package **“readxl”**.

En principe, le type de fichier le plus commun que vous rencontrerez sera le **.csv** (valeurs séparés par une virgule, en anglais, “comma-separated values”). Comme son nom l’indique, un fichier CSV distingue l’appartenance des valeurs aux variables en les séparant par des virgules, et les valeurs texte sont parfois mises entre guillemets. La première ligne du fichier contient normalement le nom des variables. Par exemple, voici les première lignes d’un fichier CSV contenant les noms des bébés écossais (voir la page des [archives nationales d’Ecosse](https://www.nrscotland.gov.uk/statistics-and-data/statistics/statistics-by-theme/vital-events/names/babies-first-names/babies-first-names-summary-records-comma-separated-value-csv-format)) :

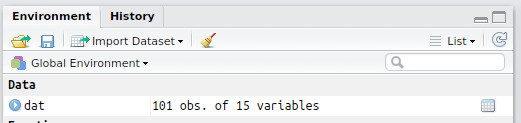


Il y a six variables dans ce jeu de données, et leurs noms sont donnés par la première ligne du fichier : **yr**, **sex**, **FirstForename**, **number**, **rank**, and **position**. Vous pouvez voir que les valeurs pour chacune de ces variables sont données dans l’ordre, séparées par des virgules, sur chaque ligne successive du fichier.

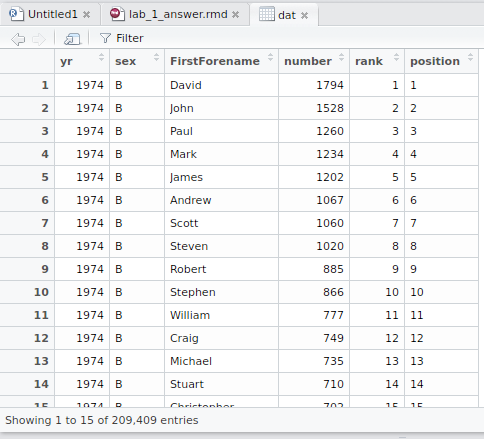
Lorsque vous lisez des fichiers CSV, il est préférable d’utiliser la fonction “**readr::read\_csv()**”. Le package readr est automatiquement chargé comme composant du package tidyverse, qui sera utilisé dans quasiment tous nos scripts. Notez qu’il est commun d’enregistrer le résultat de la fonction “**read\_csv()**” comme variable, de la manière suivante :



Un fois que vos données sont chargées, vous pouvez les visualiser dans le visionneur de données. Dans le volet supérieur droit de RStudio, sous l’onglet Environnement, vous verrez l’objet **dat** listé.



Si vous cliquez sur l’icône View (), une vue tabulaire plus “graphique” de votre tableau apparaîtra dans le volet supérieur gauche de RStudio.



Cela vous permet de vérifier que vos données ont été importées correctement. Vous pouvez fermer cette fenêtre lorsque vous avez fini de l’utiliser. Cela ne supprimera pas l'objet.

2.5.1 Ecrire les données

Si vous avez des données que vous voulez enregistrer comme fichier CSV, utilisez “**readr::write\_csv()**” comme suit.



Vos données seront alors enregistrées au format CSV dans votre répertoire de travail.

2.6 Exercices

Télécharger les [exercices](https://psyteachr.github.io/msc-data-skills/exercises/02_data_exercise.Rmd) (EN). Consulter les [solutions](https://psyteachr.github.io/msc-data-skills/exercises/02_data_answers.Rmd) (EN) uniquement lorsque vous avez essayé de répondre à toutes les questions.