Capsule 2. Familiarisation avec l'environnement de développement RStudio

Simon LaRue

08/03/2023

Installation de R et RStudio

Télécharger R

l'installation de R se fait via le site du Cran (cran.r-project.org/ (https://cran.r-project.org/)). Selectionner les liens *Download R for Windows*, suivit de *base* et finalement *Download R-#.#.# for Windows* pour télécharger le fichier d'installation.

Pour les utilisateurs de Mac, vous pouvez vous rendre sur cran.r-project.org/bin/macosx/ (https://cran.r-project.org/bin/macosx/). Deux options s'offrent à vous et vous devrez choisir la version en fonction du processeur de votre ordinateur (M1 ou Intel).

À noter, qu'au moment de la réalisation de cette capsule, la version de R la plus récente était la 4.2.2, ce qui risque de changer si une nouvelle version devient disponible. Pour mettre à jour R, il suffit de télécharger la version la plus récente en procédant de la même façon qu'a notre première installation.

Télécharger RStudio

Pour télécharger RStudio, on se rend à l'adresse posit.co/download/rstudio-desktop/ (https://posit.co/download/rstudio-desktop/). Il suffit de sélectionner le lien qui correspond à notre système d'exploitation pour télécharger le fichier d'installation.

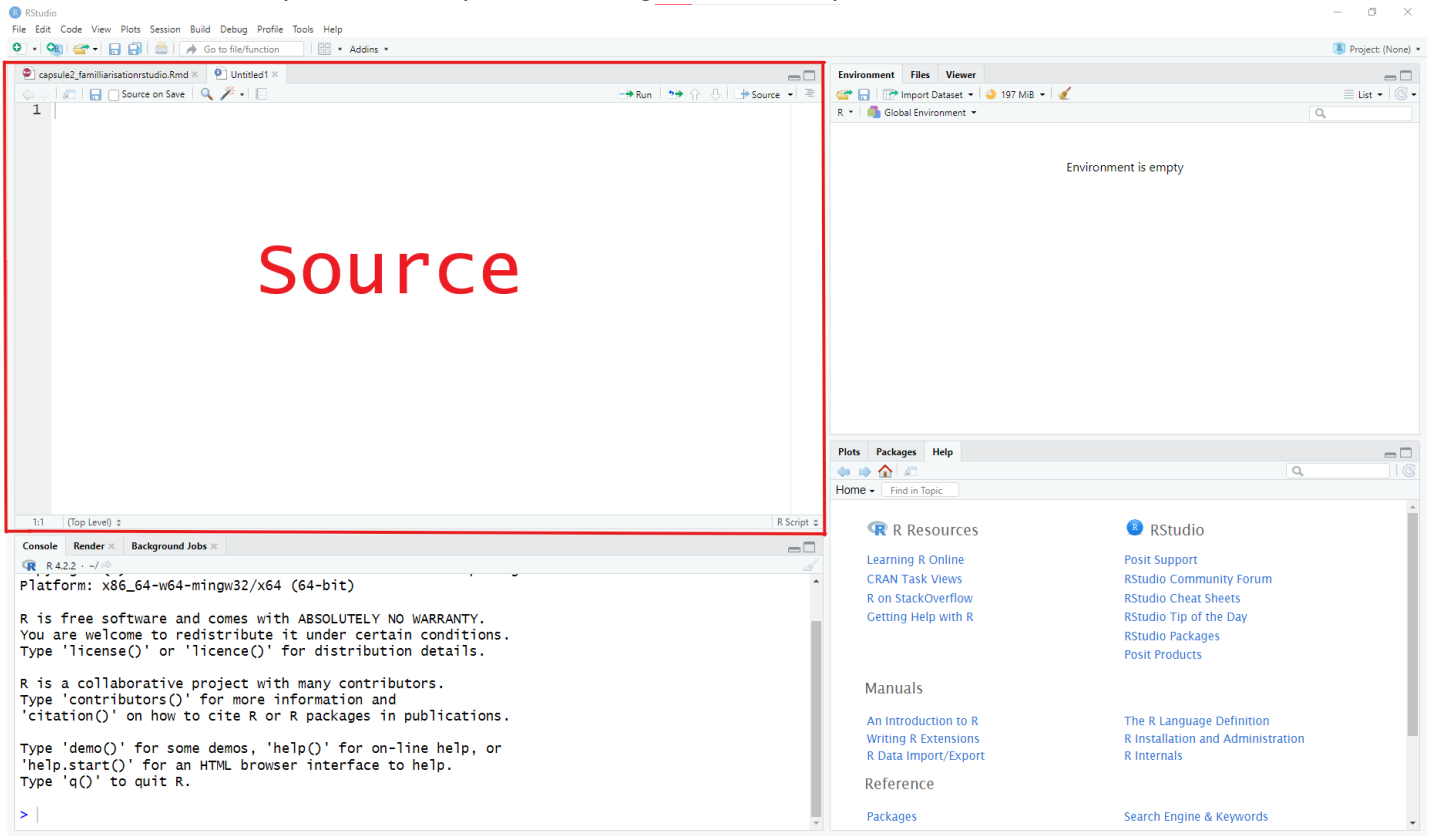
Présentation globale de l'interface

Interface

L'interface de RStudio est composée de quatre sections regroupant plusieurs fenêtres.

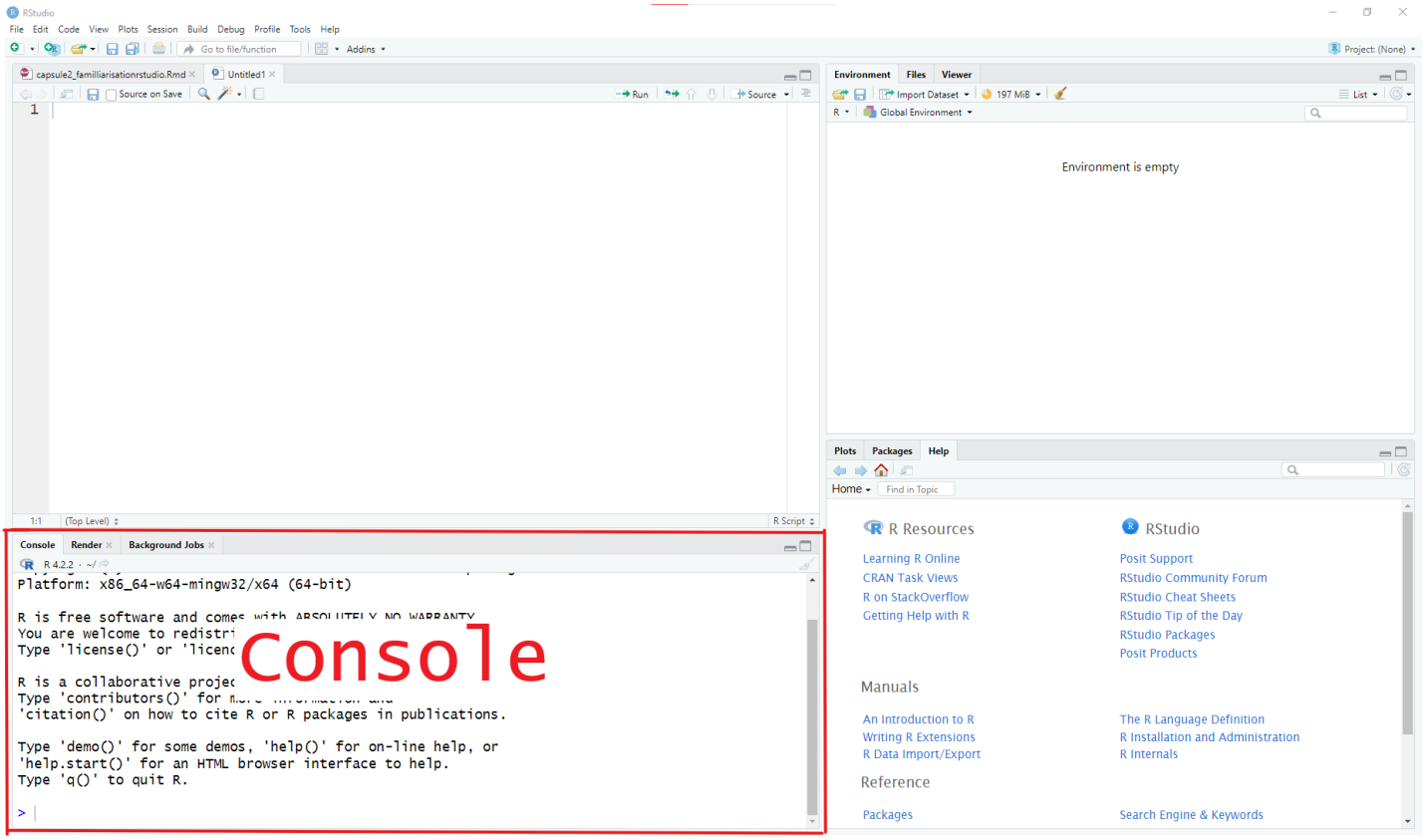
La première section est pour les fichiers sources. C'est à cet endroit que nous pouvons créer des scriptes et visualiser des documents textes et des tableaux. Les scriptes sont des fichiers textes qui permettent d'enregistrer des lignes de codes et pourrons être réutilisés à chaque session de travail. Autrement-dit, c'est l'endroit où nous allons travailler le plus. Sur Windows, des raccourcis clavier intéressant pour cette fenêtre compte ctrl+enter qui

permet d’exécuter une ligne de codes directement dans la console. Le raccourci crtl+s est notre meilleur ami et une bonne habitude à prendre, car il permet d’enregistré notre scripte.



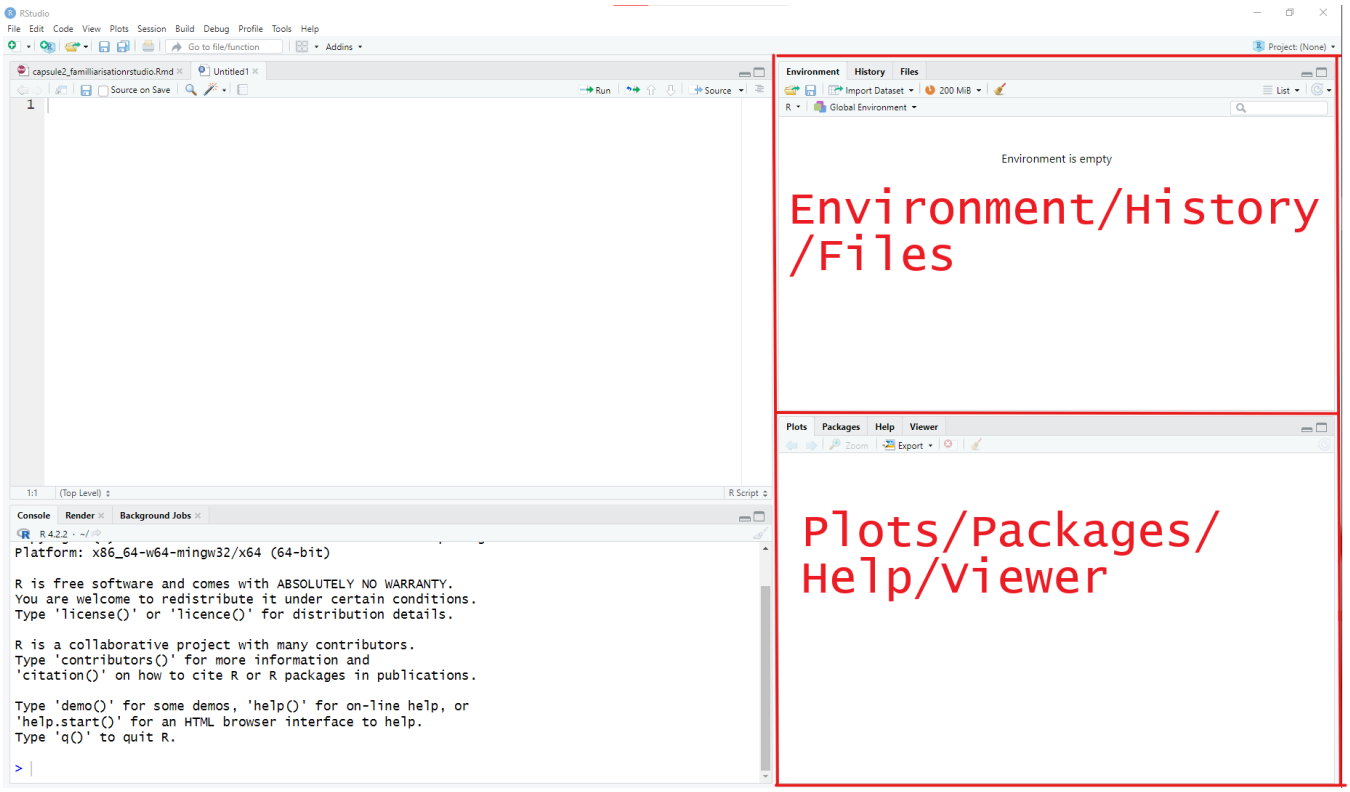
La deuxième section contient la console de R. C’est là que les commandes provenant d’un scripte ou écrit directement dans la console vont être exécutées et les résultats vont être affichés. La console va produire les résultats de codes R, ce qui inclut la manipulation de jeux de données, la création de graphiques et les différents

calcules. On peut aussi se servir de la console R comme une calculatrice en réalisant des opérations élémentaires.



Les deux dernières sections contiennent plusieurs fenêtres qui vont faciliter le travail avec l'interface.

- Environment: fenêtre qui garde une liste des objets enregistrés dans la session actuelle. Lorsqu'on affecte un objet a un nom, ça va nous l'afficher à cet endroit avec un peu d'informations sur l'objet en question. Par exemple, j'assigne le vecteur `(1,2,3,4)` au nom `vecteur_1` comme suit: `vecteur_1 <- c(1,2,3,4)`. Maintenant un objet portant le nom de `vecteur_1` contient les valeurs numériques 1,2,3,4 apparait dans l'environnement. Cette fenêtre nous permet donc de nous rappeler des objets que nous manipulons dans notre session.
- History: fenêtre qui garde une liste des commandes qui ont été exécutées. Elle permet de nous rappeler des commandes qui ont été exécutées durant notre session.
- Help: fenêtre qui affiche la documentation ou l'aide des fonctions. On peut accéder à la documentation d'une fonction via la commande `help()`, avec la commande `?"nom de la fonction"` ou en appuyant sur F1 après avoir surlignée le nom de la fonction dans un scripte. Par exemple, `help("mean")` ou `?mean` permet d'afficher la documentation de la fonction `mean()`.
- Plot: fenêtre dans laquelle s'affiche les graphiques réalisés avec R. Elle contient une icône Zoom qui permet de mieux visualiser les figures dans une plus grande fenêtre et une icône pour exporter des graphiques.
- Files: système de fichiers qui permet de naviguer dans les dossiers de l'ordinateur. Cette fenêtre peut être utilisée pour retrouver et ouvrir des documents.
- Packages: fenêtre dans laquelle s'affiche la liste des *packages* installés et chargés dans la session. Une icône permet d'installer facilement de nouveaux *packages*. Il est aussi possible de retrouver la documentation des librairies.
- Viewer: fenêtre qui permet de visualiser du contenu web local. Utile lorsque nous développons une application avec `shiny`, un package de développement web, ou nous travaillons avec des fichiers html.



Personnalisation

Sous l'onglet Tools > Global Options… On retrouve plusieurs options pour personnaliser l'interface de RStudio. Notamment, on retrouve:

- Sous l'onglet "Générale", l'option pour changer la version de R.
- L'onglet "Code" permet de personnaliser certaine fonctionnalité lors de l'écriture de code, comme la largeur des tabulations, des options d'auto-détections et même pour les plus fou, l'option "rainbow parentheses.
- L'onglet "Appearence" permet de changer le thème et la police d'écriture de l'interface.
- L'onglet "Panel Layout" permet de changer dans quelle section se retrouve les différentes fenêtres.

Installation des modules

Les modules, communément appelés *packages* en anglais, sont des extensions qui contiennent des fonctions, jeux de données, et documents. Ils permettent une utilisation plus facile des fonctionnalités présentes sur R, ou d'en ajouter de nouvelles. La diversité de *packages* disponibles est l'une des principales forces de R. Par exemple, l'un des *packages* les plus connus, `ggplot2`, facilite grandement la visualisation de données sur R. Entre autres, il est plus facile de déterminer le type de graphique, modifier les axes, les couleurs, etc. Ce sujet sera plus amplement abordé dans les capsules 11 et 12.

Il existe des *packages* déjà présents sur R, et d'autres qui doivent être téléchargés séparément. L'ensemble des *packages* existants sont répertoriés sur le site CRAN (cran.r-project.org (https://cran.r-project.org/)), et peuvent être catégorisés selon le type d'utilisation qu'on souhaite en faire (onglet Task Views). Par exemple, on voit ici un

ensemble de *packages* favorisant le développement d'essais cliniques.

CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

CRAN Task View: Clinical Trial Design, Monitoring, and Analysis

Maintainer: W.G. Zhang, R.G. Zhang, Ed Zhang
Contact: ClinicalTrials.TaskView at yahoo.com
Version: 2021-12-29
URL: https://CRAN.R-project.org/view=ClinicalTrials
Source: https://github.com/cran-task-views/ClinicalTrials/
Contributions: Suggestions and improvements for this task view are very welcome and can be made through issues or pull requests on GitHub or via e-mail to the maintainer address. For further details see the Contributing guide.
Citation: W.G. Zhang, R.G. Zhang, Ed Zhang (2021). CRAN Task View: Clinical Trial Design, Monitoring, and Analysis. Version 2021-12-29. URL https://CRAN.R-project.org/view=ClinicalTrials.
Installation: The packages from this task view can be installed automatically using the ctv package. For example, `ctv::install.views("ClinicalTrials", coreOnly = TRUE)` installs all the core packages or `ctv::update.views("ClinicalTrials")` installs all packages that are not yet installed and up-to-date. See the CRAN Task View Initiative for more details.

This task view gathers information on specific R packages for design, monitoring and analysis of data from clinical trials. It focuses on including packages for clinical trial design and monitoring in general plus data analysis packages for a specific type of design. Also, it gives a brief introduction to important packages for analyzing clinical trial data. Please refer to task views ExperimentalDesign, Survival, Pharmacokinetics, Meta-analysis for more details on these topics.

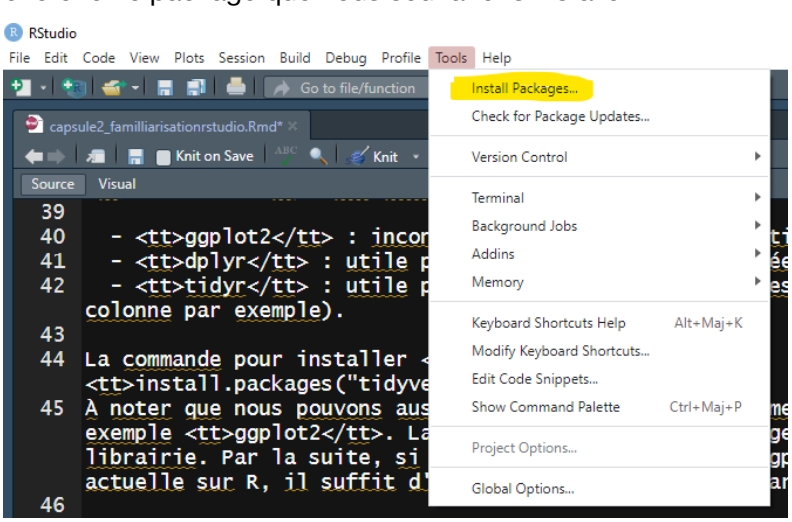
Contributions are always welcome and encouraged, either via e-mail to the maintainer or by submitting an issue or pull request in the GitHub repository linked above.

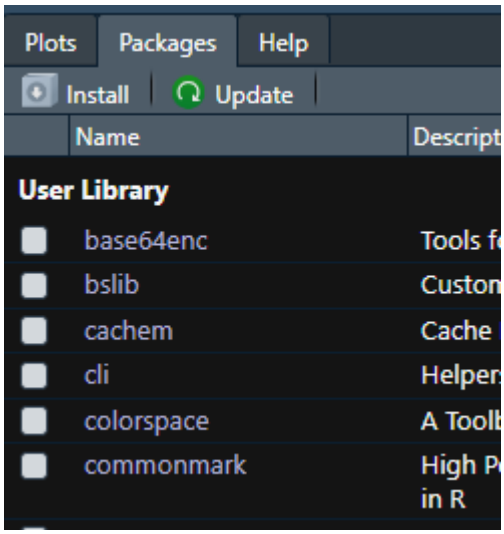
Design and Monitoring

- TrialSize This package has more than 80 functions from the book *Sample Size Calculations in Clinical Research* (Chow & Wang & Shao, 2007, 2nd ed., Chapman &Hall/CRC).
- asd This Package runs simulations for adaptive seamless designs using early outcomes for treatment selection.
- bcrm This package implements a wide variety of one and two-parameter Bayesian CRM designs. The program can run interactively, allowing the user to enter outcomes after each cohort has been recruited, or via simulation to assess operating characteristics.
- blockrand creates randomizations for block random clinical trials. It can also produce a PDF file of randomization cards.
- clusterPower Calculate power for cluster randomized trials (CRTs) that compare two means, two proportions, or two counts using closed-form solutions. In addition, calculate power for cluster randomized crossover trials using Monte Carlo methods. For more information, see Reich et al. (2012) doi:10.1371/journal.pone.0035564 .
- conf.design This small package contains a series of simple tools for constructing and manipulating confounded and fractional factorial designs.
- crmPack Implements a wide range of model-based dose escalation designs, ranging from classical and modern continual reassessment methods (CRMs) based on dose-limiting toxicity endpoints to dual-endpoint designs taking into account a biomarker/efficacy outcome. The focus is on Bayesian inference, making it very easy to setup a new design with its own JAGS code. However, it is also possible to implement 3+3 designs for comparison or models with non-Bayesian estimation. The whole package is written in a modular form in the S4 class system, making it very flexible for adaptation to new models, escalation or stopping rules.
- CRTSize This package contains basic tools for the purpose of sample size estimation in cluster (group) randomized trials. The package contains traditional power-based methods, empirical smoothing (Rotondi and Donner, 2009), and updated meta-analysis techniques (Rotondi and Donner, 2011).
- cosa Implements bound constrained optimal sample allocation (BCOSA) framework described in Bulus & Dong (2019) for power analysis of multilevel regression discontinuity designs (MRDDs) and multilevel randomized trials (MRTs) with continuous outcomes. Separate tools for statistical power and minimum detectable effect size computations are provided.
- dfcrm This package provides functions to run the CRM and TITE-CRM in phase I trials and calibration tools for trial planning purposes.
- DTAT Dose Titration Algorithm Tuning (DTAT) is a methodologic framework allowing dose individualization to be conceived as a continuous learning process that begins in early-phase clinical trials and continues throughout drug development, on into clinical practice. This package includes code that researchers may use to reproduce or extend key results of the DTAT research programme, plus tools for trialists to design and simulate a '3+3/PC' dose-finding study.
- ewoc An implementation of a variety of escalation with overdose control designs introduced by Babb, Rogatko and Zacks (1998) doi:10.1002/(SICI)1097-0258(19980530)17:10%3C1103::AID-SIM793%3E3.0.CO;2-9 . It calculates the next dose as a clinical trial proceeds as well as performs simulations to obtain operating characteristics.
- experiment contains tools for clinical experiments, e.g., a randomization tool, and it provides a few special analysis options for clinical trials.
- FrF2 This package creates regular and non-regular Fractional Factorial designs. Furthermore, analysis tools for Fractional Factorial designs with 2-level factors are offered (main effects and interaction plots for all factors simultaneously, cube plot for looking at the simultaneous effects of three factors, full or half normal plot, alias structure in a more readable format than with the built-in function alias). The package is currently subject to intensive development. While much of the intended functionality is already available, some changes and improvements are still to be expected.
- GroupSeq performs computations related to group sequential designs via the alpha spending approach, i.e., interim analyses need not be equally spaced, and their number need not be specified in advance.
- Keyboard We developed a package 'Keyboard' for designing single-agent, drug-combination, or phase I/II dose-finding clinical trials. The 'Keyboard' designs are novel early phase trial designs that can be implemented simply and transparently, similar to the 3+3 design, but yield excellent performance, comparable to those of more-complicated, model-based designs. The 'Keyboard' package provides tools for designing, conducting, and analyzing single-agent, drug-combination, and phase I/II dose-finding clinical trials.

Sur RStudio, si nous connaissons déjà le nom du package que nous souhaitons utiliser, la commande `install.packages("nom du package")` va permettre de l'installer à partir du site CRAN. Prenez note qu'une connexion Internet est nécessaire pour le téléchargement. Après avoir installé le package, il est possible de le rendre disponible pour la session actuelle, grâce à la commande `library("nom du package")`. Notez que cette commande est valable uniquement pour la session actuelle. Si nous ouvrons une nouvelle session, nous aurons à entrer la commande une nouvelle fois, mais pas à réinstaller le package en question. La commande `install.packages("nom du package")` permet aussi de mettre à jour les librairies à leur version la plus récente.

Pour installer une librairie, il est aussi possible de passer par le bouton *Install* de la fenêtre "Packages" ou encore par l'option *Install Packages…* de l'onglet "Tools". Dans les deux cas, une fenêtre s'ouvre dans laquelle on peut chercher le package que nous souhaitons installer.





Voici une collection de *packages*, qu'on juge essentielle à télécharger. Il s'agit de `tidyverse`, qui renferme entre autres les *packages* suivants :

- `ggplot2` : incontournable pour la visualisation des données.

- `dplyr` : utile pour la manipulation de données.
- `tidyr` : utile pour bien ordonner ses données (organisation en ligne et colonne par exemple).

La commande pour installer `tidyverse` est : `install.packages("tidyverse")`. À noter que nous pouvons aussi télécharger individuellement chaque package. Prenons comme exemple `ggplot2`. La commande `install.packages("ggplot2")` installe la librairie. Par la suite, si nous voulons utiliser `ggplot2` dans notre session actuelle sur R, il suffit d'entrer la commande `library("ggplot2")` comme illustré ci-dessous:

```
install.packages(ggplot2)
library(ggplot2)
```

Utilisation de la console

Comme mentionné plus haut, la console <R est comme une calculatrice. Il est possible de réaliser des sommes avec l'opérateur "+", des différences avec l'opérateur "-", des multiplications avec l'opérateur "*" et des divisions avec l'opérateur "/". Ainsi, lorsque j'exécute la commande `2 + 3` dans la console, elle me retourne bien 5.

```
2 + 3
```

```
## [1] 5
```

L'opérateur "^" permet de calculer les puissances d'un nombre.

```
4^3
```

```
## [1] 64
```

```
27^(1/3) # racine cubique
```

```
## [1] 3
```

La fonction `sqrt()` permet aussi de calculer la racine carré d'un nombre.

```
sqrt(49)
```

```
## [1] 7
```

D'autres fonctions comme `log()` et `exp()` permettent le calcule de le logarithme et l'exponentielle d'un nombre respectivement.

```
exp(3)
```

```
## [1] 20.08554
```

```
log(16)
```

```
## [1] 2.772589
```