# Epi R handbook

<https://epirhandbook.com/transition-to-r.html>

4 Transition vers R

Ci-dessous, nous fournissons des conseils et des ressources si vous passez à R.

R a été introduit à la fin des années 1990 et a depuis augmenté considérablement sa portée. Ses capacités sont si étendues que les alternatives commerciales ont réagi aux développements de R afin de rester compétitives ! ([lisez cet article comparant R, SPSS, SAS, STATA et Python](https://www.inwt-statistics.com/read-blog/comparison-of-r-python-sas-spss-and-stata.html)).

De plus, R est beaucoup plus facile à apprendre qu'il ne l'était il y a 10 ans. Auparavant, R avait la réputation d'être difficile pour les débutants. C'est maintenant beaucoup plus facile avec des interfaces utilisateur conviviales comme RStudio, un code intuitif comme le tidyverse et de nombreuses ressources didactiques.

**Ne vous laissez pas intimider - venez découvrir le monde de R !**

Une image contenant texte, intérieur

Description générée automatiquement

4.1 Depuis Excel

Passer d'Excel directement à R est un objectif tout à fait réalisable. Cela peut sembler intimidant, mais vous pouvez le faire !

Il est vrai qu'une personne possédant de solides compétences en Excel peut effectuer des activités très avancées avec seulement Excel, même en utilisant des outils de script comme VBA. Excel est utilisé dans le monde entier et est un outil essentiel pour un épidémiologiste. Cependant, le compléter avec R peut considérablement améliorer et étendre vos flux de travail.

Avantages

Vous constaterez que l'utilisation de R offre d'immenses avantages en termes de gain de temps, d'analyse plus cohérente et précise, de reproductibilité, de partageabilité et de correction d'erreurs plus rapide. Comme tout nouveau logiciel, il existe une « courbe » de temps d'apprentissage que vous devez investir pour vous familiariser. Les dividendes seront importants et un immense champ de nouvelles possibilités s'ouvrira à vous avec R.

Excel est un logiciel bien connu qui peut être facile à utiliser pour un débutant pour produire des analyses et des visualisations simples avec « pointer-cliquer ». En comparaison, cela peut prendre quelques semaines pour se familiariser avec les fonctions et l'interface R. Cependant, R a évolué ces dernières années pour devenir beaucoup plus convivial pour les débutants.

De nombreux processus de travail dans Excel reposent sur la mémoire et la répétition - il y a donc beaucoup de risques d'erreur. De plus, le nettoyage des données, la méthodologie d'analyse et les équations utilisées sont généralement cachés. Il peut falloir beaucoup de temps à un nouveau collègue pour apprendre ce que fait un classeur Excel et comment le résoudre. Avec R, toutes les étapes sont explicitement écrites dans le script et peuvent être facilement visualisées, modifiées, corrigées et appliquées à d'autres ensembles de données.

**Pour commencer votre transition d'Excel vers R, vous devez ajuster votre état d'esprit de plusieurs manières importantes :**

Des données ordonnées

Utilisez des données « ordonnées » lisibles par votre ordinateur au lieu de données « lisibles par l'homme » désordonnées. Ce sont les trois principales exigences pour des données « ordonnées », comme expliqué dans ce tutoriel sur les [données « ordonnées » dans R](https://r4ds.had.co.nz/tidy-data.html):

* Chaque variable doit avoir sa propre colonne
* Chaque observation doit avoir sa propre ligne
* Chaque valeur doit avoir sa propre cellule

Pour les utilisateurs d'Excel, pensez au rôle que jouent les [« tableaux » Excel](https://exceljet.net/excel-tables) dans la normalisation des données et pour rendre le format plus prévisible.

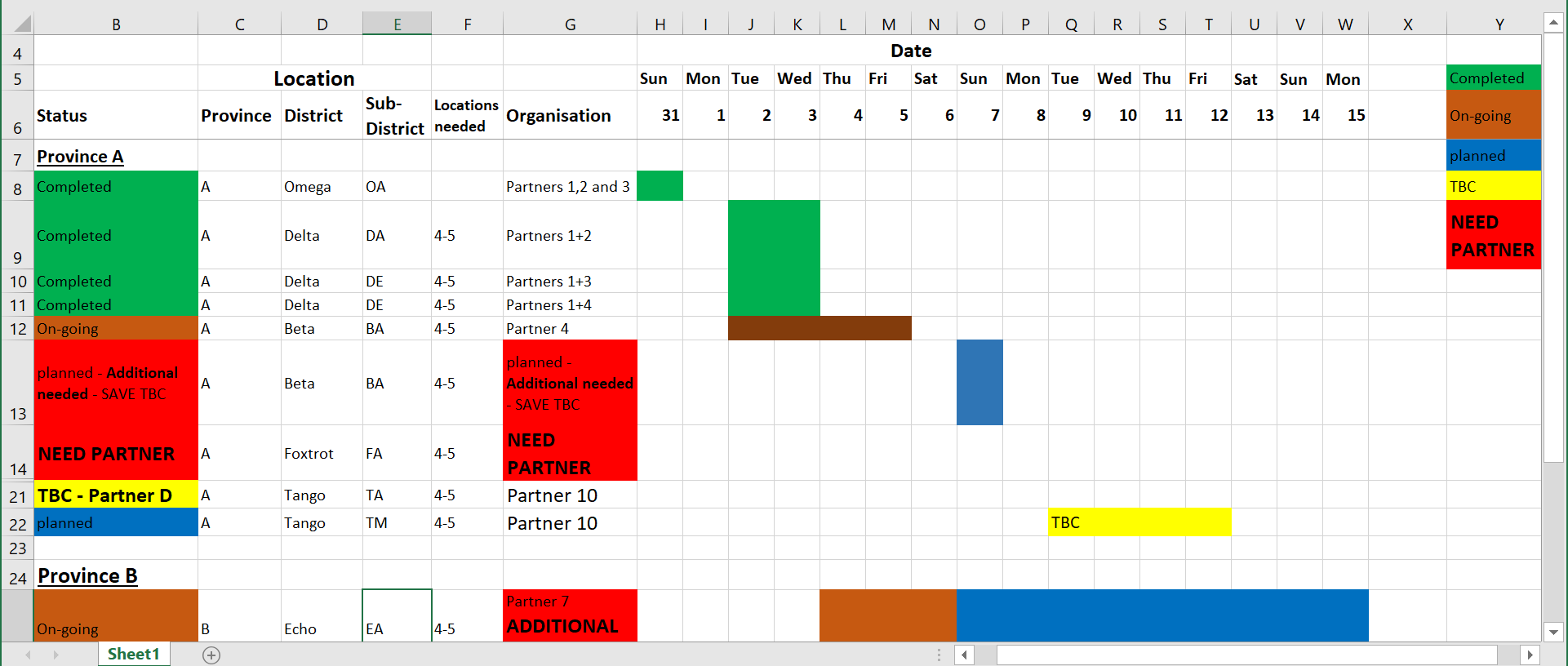
Un exemple de données « ordonnées » serait le registre de cas utilisée dans ce manuel - chaque variable est contenue dans une colonne, chaque observation (un cas) a sa propre ligne et chaque valeur se trouve dans une seule cellule. Ci-dessous, vous pouvez voir les 50 premières lignes du registre:

Une image contenant texte, table

Description générée automatiquement

*La principale raison pour laquelle on rencontre des données non ordonnées est que de nombreuses feuilles de calcul Excel sont conçues pour donner la priorité à la lecture facile par les humains, et non à la lecture facile par les ordinateurs/logiciels.*

Pour vous aider à voir la différence, voici quelques exemples fictifs de données non ordonnées qui donnent la priorité à la lisibilité humaine par rapport à la lisibilité par ordinateur:



*Problèmes :* Dans la feuille de calcul ci-dessus, il existe des cellules fusionnées qui ne sont pas facilement digérées par R. La ligne à considérer comme « en-tête » n'est pas claire. Un dictionnaire basé sur les couleurs se trouve à droite et les valeurs des cellules sont représentées par des couleurs - ce qui n'est pas non plus facilement interprété par R (ni par les humains daltoniens !). De plus, différentes informations sont regroupées dans une seule cellule (plusieurs organisations partenaires travaillant dans un même domaine, ou le statut « à confirmer » dans la même cellule que « Partenaire D »).

Une image contenant table

Description générée automatiquement

*Problèmes* : Dans la feuille de calcul ci-dessus, il existe de nombreuses lignes et colonnes vides supplémentaires dans l'ensemble de données - cela entraînera des maux de tête pour le nettoyage des données dans R. De plus, les coordonnées GPS sont réparties sur deux lignes pour un centre de traitement donné. En passant, les coordonnées GPS sont dans deux formats différents !

Les ensembles de données « ordonnés » ne sont peut-être pas aussi lisibles à l'œil nu, mais ils facilitent grandement le nettoyage et l'analyse des données ! Les données ordonnées peuvent être stockées sous différents formats, par exemple « longs » ou « larges » (voir la page sur les [données pivotantes](https://epirhandbook.com/pivoting-data.html#pivoting-data)), mais les principes ci-dessus sont toujours respectés.

Les fonctions

Le mot R "fonction" est peut-être nouveau, mais le concept existe également dans Excel sous forme de *formules*. Les formules dans Excel nécessitent également une syntaxe précise (par exemple, le placement de points-virgules et de parenthèses). Tout ce que vous avez à faire est d'apprendre quelques nouvelles fonctions et comment elles fonctionnent ensemble dans R.

Scipts

Au lieu de cliquer sur des boutons et de faire glisser des cellules, vous écrirez chaque étape et procédure dans un « script ». Les utilisateurs d'Excel peuvent être familiers avec les « macros VBA » qui utilisent également une approche de script.

*Le script R se compose d'instructions étape par étape.* Cela permet à n'importe quel collègue de lire le script et de voir facilement les étapes que vous avez suivies. Cela permet également de déboguer les erreurs ou les calculs inexacts. Voir la section [R basics](https://epirhandbook.com/r-basics.html#r-basics) sur les scripts pour des exemples.

Voici un exemple de script R :

Une image contenant texte

Description générée automatiquement

Ressources « Excel vers R »

Voici quelques liens vers des tutoriels pour vous aider à passer à R à partir d'Excel :

* [R contre Excel](https://www.northeastern.edu/graduate/blog/r-vs-excel/)
* [Cours RStudio en R pour les utilisateurs d'Excel](https://rstudio-conf-2020.github.io/r-for-excel/)

Interaction R-Excel

R propose des moyens robustes d'importer des classeurs Excel, de travailler avec les données, d'exporter/enregistrer des fichiers Excel et de travailler avec des feuilles Excel.

Il est vrai que certains formats Excel esthétiques peuvent se perdre dans la conversion (par exemple, italique, texte latéral, etc.). Si votre flux de travail nécessite le transfert de documents entre R et Excel tout en conservant le formatage Excel d'origine, essayez des packages tels que **openxlsx**.

4.2 Depuis Stata

**Venir à R de Stata**

De nombreux épidémiologistes apprennent d'abord à utiliser Stata, et il peut sembler intimidant de passer à R. Cependant, si vous êtes un utilisateur confortable de Stata, le saut dans R est certainement plus gérable que vous ne le pensez. Bien qu'il existe des différences clés entre Stata et R dans la façon dont les données peuvent être créées et modifiées, ainsi que dans la façon dont les fonctions d'analyse sont mises en œuvre, après avoir appris ces différences clés, vous serez en mesure de traduire vos compétences.

Vous trouverez ci-dessous quelques conversions clés entre Stata et R, qui peuvent vous être utiles lorsque vous consultez ce guide.

**Notes générales**

| **STATA** | **R** |
| --- | --- |
| You can only view and manipulate one dataset at a time | You can view and manipulate multiple datasets at the same time, therefore you will frequently have to specify your dataset within the code |
| Online community available through <https://www.statalist.org/> | Online community available through [RStudio](https://community.rstudio.com/), [StackOverFlow](https://stackoverflow.com/questions/tagged/r), and [R-bloggers](https://www.r-bloggers.com/) |
| Point and click functionality as an option | Minimal point and click functionality |
| Help for commands available by help [command] | Help available by [function]? or search in the Help pane |
| Comment code using \* or /// or /\* TEXT \*/ | Comment code using # |
| Almost all commands are built-in to Stata. New/user-written functions can be installed as **ado** files using **ssc install** [package] | R installs with **base** functions, but typical use involves installing other packages from CRAN (see page on [R basics](https://epirhandbook.com/r-basics.html#r-basics)) |
| Analysis is usually written in a **do** file | Analysis written in an R script in the RStudio source pane. R markdown scripts are an alternative. |

**Working directory**

| **STATA** | **R** |
| --- | --- |
| Working directories involve absolute filepaths (e.g. “C:/usename/documents/projects/data/”) | Working directories can be either absolute, or relative to a project root folder by using the **here** package (see [Import and export](https://epirhandbook.com/import-and-export.html#import-and-export)) |
| See current working directory with **pwd** | Use [getwd()](https://rdrr.io/r/base/getwd.html) or [here()](https://here.r-lib.org/reference/here.html) (if using the **here** package), with empty parentheses |
| Set working directory with **cd** “folder location” | Use setwd(“folder location”), or set\_here("folder location) (if using **here** package) |

**Importing and viewing data**

| **STATA** | **R** |
| --- | --- |
| Specific commands per file type | Use [import()](https://rdrr.io/pkg/rio/man/import.html) from **rio** package for almost all filetypes. Specific functions exist as alternatives (see [Import and export](https://epirhandbook.com/import-and-export.html#import-and-export)) |
| Reading in csv files is done by **import delimited** “filename.csv” | Use [import("filename.csv")](https://rdrr.io/pkg/rio/man/import.html) |
| Reading in xslx files is done by **import excel** “filename.xlsx” | Use [import("filename.xlsx")](https://rdrr.io/pkg/rio/man/import.html) |
| Browse your data in a new window using the command **browse** | View a dataset in the RStudio source pane using [View(dataset)](https://rdrr.io/r/utils/View.html). *You need to specify your dataset name to the function in R because multiple datasets can be held at the same time. Note capital “V” in this function* |
| Get a high-level overview of your dataset using **summarize**, which provides the variable names and basic information | Get a high-level overview of your dataset using [summary(dataset)](https://rdrr.io/r/base/summary.html) |

**Basic data manipulation**

| **STATA** | **R** |
| --- | --- |
| Dataset columns are often referred to as “variables” | More often referred to as “columns” or sometimes as “vectors” or “variables” |
| No need to specify the dataset | In each of the below commands, you need to specify the dataset - see the page on [Cleaning data and core functions](https://epirhandbook.com/cleaning-data-and-core-functions.html#cleaning-data-and-core-functions) for examples |
| New variables are created using the command **generate** *varname* = | Generate new variables using the function mutate(varname = ). See page on [Cleaning data and core functions](https://epirhandbook.com/cleaning-data-and-core-functions.html#cleaning-data-and-core-functions) for details on all the below **dplyr** functions. |
| Variables are renamed using **rename** *old\_name new\_name* | Columns can be renamed using the function rename(new\_name = old\_name) |
| Variables are dropped using **drop** *varname* | Columns can be removed using the function select() with the column name in the parentheses following a minus sign |
| Factor variables can be labeled using a series of commands such as **label define** | Labeling values can done by converting the column to Factor class and specifying levels. See page on [Factors](https://epirhandbook.com/factors.html#factors). Column names are not typically labeled as they are in Stata. |

**Descriptive analysis**

| **STATA** | **R** |
| --- | --- |
| Tabulate counts of a variable using **tab** *varname* | Provide the dataset and column name to [table()](https://rdrr.io/r/base/table.html) such as [table(dataset$colname)](https://rdrr.io/r/base/table.html). Alternatively, use count(varname) from the **dplyr** package, as explained in [Grouping data](https://epirhandbook.com/grouping-data.html#grouping-data) |
| Cross-tabulaton of two variables in a 2x2 table is done with **tab** *varname1 varname2* | Use table(dataset$varname1, dataset$varname2 or count(varname1, varname2) |

While this list gives an overview of the basics in translating Stata commands into R, it is not exhaustive. There are many other great resources for Stata users transitioning to R that could be of interest:

* <https://dss.princeton.edu/training/RStata.pdf>
* <https://clanfear.github.io/Stata_R_Equivalency/docs/r_stata_commands.html>
* <http://r4stats.com/books/r4stata/>

4.3 From SAS

**Coming from SAS to R**

SAS is commonly used at public health agencies and academic research fields. Although transitioning to a new language is rarely a simple process, understanding key differences between SAS and R may help you start to navigate the new language using your native language. Below outlines the key translations in data management and descriptive analysis between SAS and R.

**General notes**

| **SAS** | **R** |
| --- | --- |
| Online community available through [SAS Customer Support](https://support.sas.com/en/support-home.html) | Online community available through RStudio, StackOverFlow, and R-bloggers |
| Help for commands available by help [command] | Help available by [function]? or search in the Help pane |
| Comment code using \* TEXT ; or /\* TEXT \*/ | Comment code using # |
| Almost all commands are built-in. Users can write new functions using SAS macro, SAS/IML, SAS Component Language (SCL), and most recently, procedures Proc Fcmp and Proc Proto | R installs with **base** functions, but typical use involves installing other packages from CRAN (see page on [R basics](https://epirhandbook.com/r-basics.html#r-basics)) |
| Analysis is usually conducted by writing a SAS program in the Editor window. | Analysis written in an R script in the RStudio source pane. R markdown scripts are an alternative. |

**Working directory**

| **SAS** | **R** |
| --- | --- |
| Working directories can be either absolute, or relative to a project root folder by defining the root folder using %let rootdir=/root path; %include “&rootdir/subfoldername/filename” | Working directories can be either absolute, or relative to a project root folder by using the **here** package (see [Import and export](https://epirhandbook.com/import-and-export.html#import-and-export)) |
| See current working directory with %put %sysfunc(getoption(work)); | Use [getwd()](https://rdrr.io/r/base/getwd.html) or [here()](https://here.r-lib.org/reference/here.html) (if using the **here** package), with empty parentheses |
| Set working directory with libname “folder location” | Use setwd(“folder location”), or set\_here("folder location) if using **here** package |

**Importing and viewing data**

| **SAS** | **R** |
| --- | --- |
| Use Proc Import procedure or using Data Step Infile statement. | Use [import()](https://rdrr.io/pkg/rio/man/import.html) from **rio** package for almost all filetypes. Specific functions exist as alternatives (see [Import and export](https://epirhandbook.com/import-and-export.html#import-and-export)) |
| Reading in csv files is done by using Proc Import datafile=”filename.csv” out=work.filename dbms=CSV; run; OR using [Data Step Infile statement](http://support.sas.com/techsup/technote/ts673.pdf) | Use [import("filename.csv")](https://rdrr.io/pkg/rio/man/import.html) |

Reading in xslx files is done by using Proc Import datafile=”filename.xlsx” out=work.filename dbms=xlsx; run; OR using [Data Step Infile statement](http://support.sas.com/techsup/technote/ts673.pdf)|Use import(“filename.xlsx”) Browse your data in a new window by opening the Explorer window and select desired library and the dataset|View a dataset in the RStudio source pane using View(dataset). You need to specify your dataset name to the function in R because multiple datasets can be held at the same time. Note capital “V” in this function

**Basic data manipulation**

| **SAS** | **R** |
| --- | --- |
| Dataset columns are often referred to as “variables” | More often referred to as “columns” or sometimes as “vectors” or “variables” |
| No special procedures are needed to create a variable. New variables are created simply by typing the new variable name, followed by an equal sign, and then an expression for the value | Generate new variables using the function mutate(). See page on [Cleaning data and core functions](https://epirhandbook.com/cleaning-data-and-core-functions.html#cleaning-data-and-core-functions) for details on all the below **dplyr** functions. |
| Variables are renamed using rename \*old\_name=new\_name\* | Columns can be renamed using the function rename(new\_name = old\_name) |
| Variables are kept using \*\*keep\*\*=varname | Columns can be selected using the function select() with the column name in the parentheses |
| Variables are dropped using \*\*drop\*\*=varname | Columns can be removed using the function select() with the column name in the parentheses following a minus sign |
| Factor variables can be labeled in the Data Step using Label statement | Labeling values can done by converting the column to Factor class and specifying levels. See page on [Factors](https://epirhandbook.com/factors.html#factors). Column names are not typically labeled. |
| Records are selected using Where or If statement in the Data Step. Multiple selection conditions are separated using “and” command. | Records are selected using the function [filter()](https://rdrr.io/r/stats/filter.html) with multiple selection conditions separated either by an AND operator (&) or a comma |
| Datasets are combined using Merge statement in the Data Step. The datasets to be merged need to be sorted first using Proc Sort procedure. | **dplyr** package offers a few functions for merging datasets. See page [Joining Data](https://epirhandbook.com/joining-data.html#joining-data) for details. |

**Descriptive analysis**

| **SAS** | **R** |
| --- | --- |
| Get a high-level overview of your dataset using Proc Summary procedure, which provides the variable names and descriptive statistics | Get a high-level overview of your dataset using [summary(dataset)](https://rdrr.io/r/base/summary.html) or skim(dataset) from the **skimr** package |
| Tabulate counts of a variable using proc freq data=Dataset; Tables varname; Run; | See the page on [Descriptive tables](https://epirhandbook.com/descriptive-tables.html" \l "descriptive-tables). Options include [table()](https://rdrr.io/r/base/table.html) from **base** R, and tabyl() from **janitor** package, among others. Note you will need to specify the dataset and column name as R holds multiple datasets. |
| Cross-tabulation of two variables in a 2x2 table is done with proc freq data=Dataset; Tables rowvar\*colvar; Run; | Again, you can use [table()](https://rdrr.io/r/base/table.html), tabyl() or other options as described in the [Descriptive tables](https://epirhandbook.com/descriptive-tables.html#descriptive-tables) page. |

**Some useful resources:**

[R for SAS and SPSS Users (2011)](https://www.amazon.com/SAS-SPSS-Users-Statistics-Computing/dp/1461406846/ref=sr_1_1?dchild=1&gclid=EAIaIQobChMIoqLOvf6u7wIVAhLnCh1c9w_DEAMYASAAEgJLIfD_BwE&hvadid=241675955927&hvdev=c&hvlocphy=9032185&hvnetw=g&hvqmt=e&hvrand=16854847287059617468&hvtargid=kwd-44746119007&hydadcr=16374_10302157&keywords=r+for+sas+users&qid=1615698213&sr=8-1)

[SAS and R, Second Edition (2014)](https://www.amazon.com/SAS-Management-Statistical-Analysis-Graphics-dp-1466584491/dp/1466584491/ref=dp_ob_title_bk)

4.4 Data interoperability

See the [Import and export](https://epirhandbook.com/import-and-export.html#import-and-export) page for details on how the R package **rio** can import and export files such as STATA .dta files, SAS .xpt and.sas7bdat files, SPSS .por and.sav files, and many others.