
Notes de leçon | Projets RStudio

GRAPH Network & OMS, soutenu par le Fonds Mondial

January 2024

Ce cours a été créé par le Réseau GRAPH, une organisation à but non lucratif basée à l'Institut de santé globale de l'Université de Genève, en collaboration avec l'Organisation mondiale de la Santé, dans le cadre d'une subvention du Fonds mondial pour créer des cours afin de renforcer les capacités nationales en matière d'analyse épidémiologique.

Premiers pas: projets RStudio	
Objectifs d'apprentissage	
Introduction	
Création d'un nouveau Projet RStudio	
Création de sous-dossiers de projet	
Ajout d'un jeu de données au dossier "data"	
Création d'un script dans le dossier "scripts"	
Importation de données depuis le dossier "data"	
Exportation des données vers le dossier "outputs"	
Exportation des graphiques vers le dossier "outputs"	
Partager un projet	
Conclusion	

Premiers pas: projets RStudio

Objectifs d'apprentissage

1. Vous pouvez configurer un projet RStudio et créer des sous-répertoires pour les données d'entrée, les scripts et les résultats analytiques.
2. Vous pouvez importer et exporter des données dans un projet RStudio.
3. Vous comprenez la différence entre les chemins de fichiers relatifs et absolus.
4. Vous reconnaissez la valeur des Projets pour organiser et partager vos analyses.

Introduction

Précédemment, vous avez parcouru certaines des étapes essentielles de l'analyse des données, de l'importation des données au calcul des statistiques de base. Mais vous avez sauté une étape cruciale : la mise en place d'un *projet* d'analyse de données.

Les analystes de données expérimentés conservent tous les fichiers associés à une analyse spécifique — données d'entrée, scripts R et résultats analytiques — ensemble dans un seul dossier. Ces dossiers sont appelés *projets* (p minuscule), et RStudio les prend en charge via RStudio *Projets* (P majuscule).

Dans cette leçon, vous apprendrez à utiliser ces projets RStudio pour organiser votre analyse de données de manière cohérente et améliorer la reproductibilité de votre travail. Vous répliquerez une partie de l'analyse que vous avez effectuée lors de la dernière leçon d'exploration des données, mais dans le contexte d'un projet RStudio.

Allons-y.

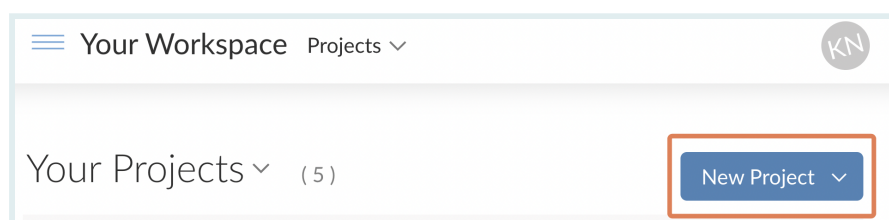
Création d'un nouveau Projet RStudio

La création d'un nouveau projet RStudio est différente si vous êtes sur un ordinateur local et si vous êtes sur RStudio Cloud. Passez à la section qui vous concerne.

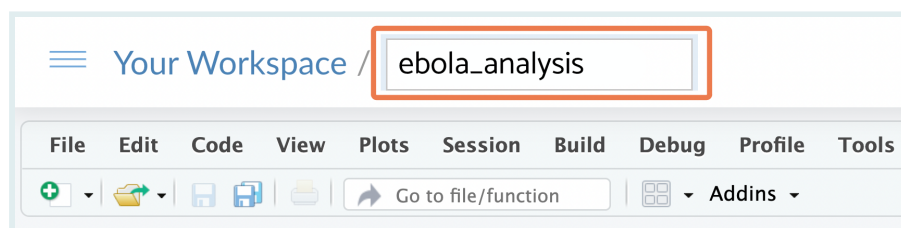
Sur RStudio Cloud

Si vous utilisez RStudio Cloud, vous avez probablement *déjà* créé un projet, car vous ne pouvez pas faire d'analyse sans projets.

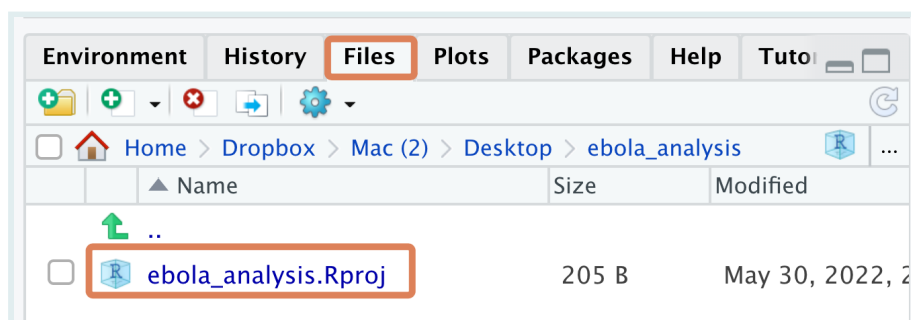
Les étapes sont assez simples : accédez à votre page d'accueil Cloud, rstudio.cloud, et cliquez sur le bouton "New Project".



Nommez votre projet quelque chose comme `analyse_ebola` ou `analyse_ebola_proj` si vous avez déjà un projet nommé `analyse_ebola`.



Le projet RStudio que vous venez de créer n'est qu'un dossier sur un ordinateur virtuel, qui contient un fichier `.Rproj` (et peut-être un fichier `.RHistory`). Vous devriez pouvoir voir ce fichier `.Rproj` dans l'onglet Files de RStudio :



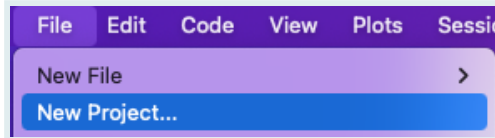
KEY POINT



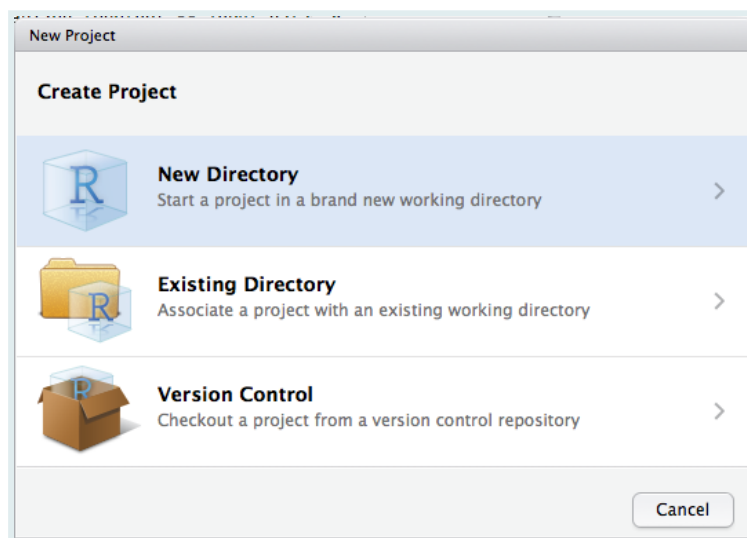
Le fichier `.Rproj` est ce qui transforme un dossier ordinaire de l'ordinateur en un "Projet RStudio".

Sur un ordinateur local

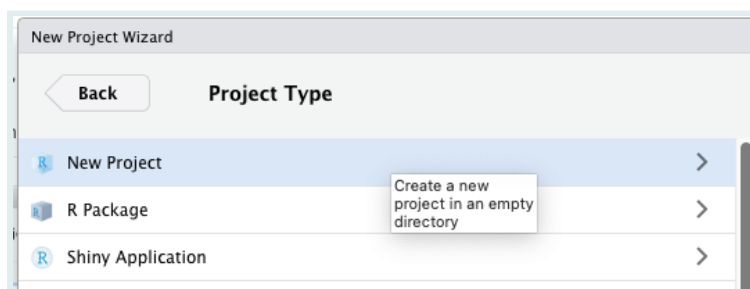
Si vous êtes sur un ordinateur local, ouvrez RStudio, puis dans le menu RStudio, allez dans “File > New Project”. Vos options peuvent être légèrement différentes des captures d’écran ci-dessous en fonction de votre système d’exploitation.



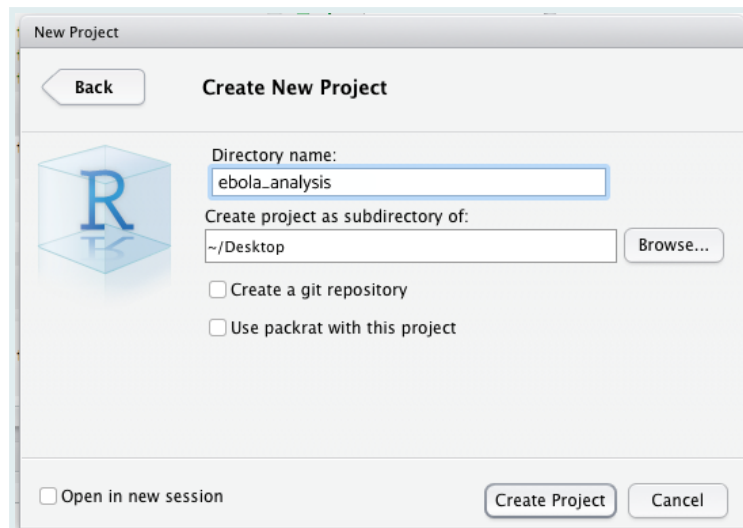
Choisissez “New directory”



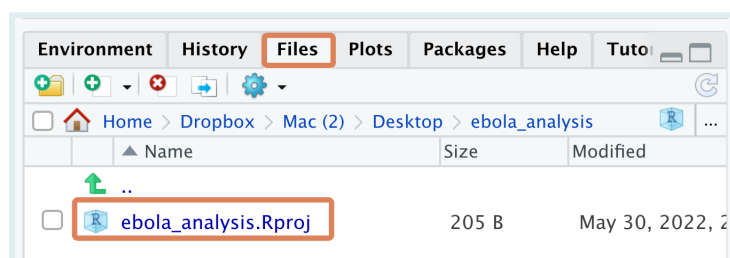
Choisissez ensuite “New Project”:



Vous pouvez appeler votre projet quelque chose comme “analyse Ebola” et en faire un “sous-répertoire” d’un dossier facile à trouver, tel que votre bureau. (L’expression “Créer un projet en tant que sous-répertoire de” semble effrayante, mais ce n’est pas le cas ; RStudio demande simplement : “Où dois-je mettre le dossier du projet” ?)



Le projet RStudio que vous avez créé n'est qu'un dossier contenant un fichier .Rproj (et peut-être un fichier .RHistory). Vous devriez pouvoir voir ce fichier .Rproj dans l'onglet Files de RStudio :



Cliquez sur le fichier .Rproj pour ouvrir votre projet

KEY POINT



Le fichier .Rproj est ce qui transforme un dossier ordinaire de l'ordinateur en un "Projet RStudio".

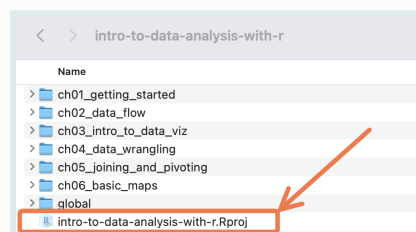
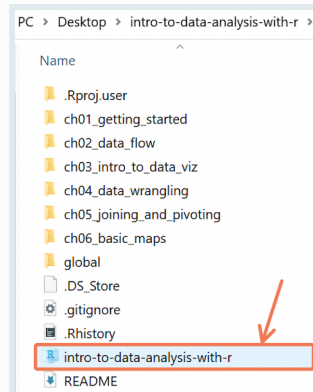
Désormais, pour ouvrir votre projet, vous devez double-cliquer sur ce fichier .Rproj à partir du Finder/Explorateur de fichiers de votre ordinateur.

Sous Windows, voici un exemple de ce à quoi ressemble un fichier .Rproj dans l'explorateur de fichiers :

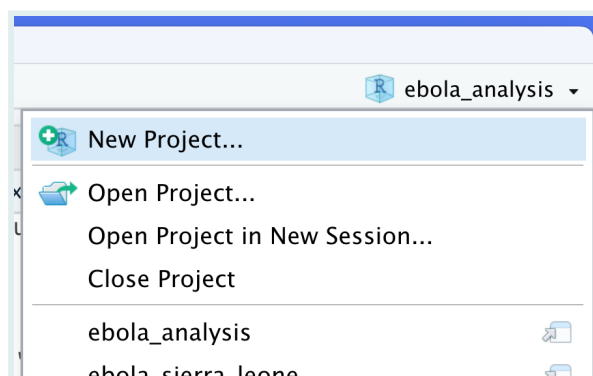
KEY POINT



Sur macOS, voici un exemple de ce à quoi ressemble un fichier .Rproj dans le Finder :

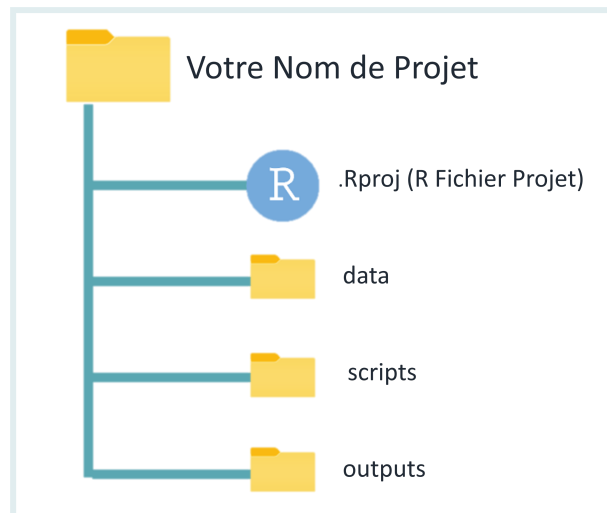


Notez également qu'il y a un en-tête en haut à droite de la fenêtre RStudio qui vous indique quel projet vous avez actuellement ouvert. En cliquant sur cet en-tête, vous pouvez accéder à d'autres options du projet. Vous pouvez notamment créer un nouveau projet, fermer un projet et ouvrir des projets récents.



Création de sous-dossiers de projet

Les projets d'analyse de données comportent généralement au moins trois sous-dossiers : un pour les données, un autre pour les scripts et un troisième pour les résultats, comme indiqué ci-dessous :



Regardons les sous-dossiers un par un :

- **data:** Il contient les fichiers de données source (brutes) que vous utiliserez dans l'analyse. Il peut s'agir de fichiers CSV ou Excel, par exemple.
- **scripts:** Ce sous-dossier est l'endroit où vous conservez vos scripts R. Vous pouvez également enregistrer les fichiers RMarkdown dans ce dossier. (Vous découvrirez bientôt les fichiers RMarkdown.)
- **outputs:** Ici, vous enregistrez les résultats de votre analyse, comme les graphiques et les tableaux récapitulatifs. Ces résultats doivent être *jetable*s et *reproductibles*. Autrement dit, vous devriez pouvoir régénérer les résultats en exécutant le code dans vos scripts. Vous le comprendrez mieux bientôt.

Créez maintenant ces trois sous-dossiers, “data”, “scripts” et “outputs” dans votre dossier RStudio projet. Pour ce faire, Vous devez utiliser le bouton “New Folder” (Nouveau dossier) dans l'onglet Files de RStudio:



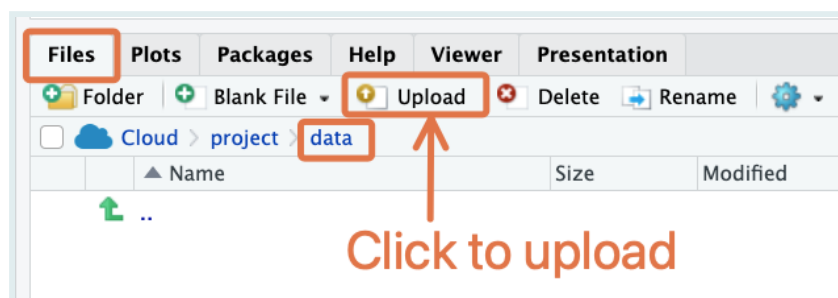
Ajout d'un jeu de données au dossier “data”

Ensuite, vous devez déplacer le jeu de données Ebola que vous avez téléchargé dans la leçon précédente vers le sous-dossier “data” nouvellement créé (vous pouvez retélécharger ce jeu de données sur bit.ly/ebola-data si vous ne parvenez pas à retrouver l'endroit où vous l'avez stocké).

La procédure de déplacement de ce jeu de données vers le dossier “data” est différente pour les utilisateurs de RStudio Cloud et ceux utilisant un ordinateur local. Passez à la section qui vous concerne.

Sur RStudio Cloud

Si vous êtes sur RStudio Cloud, l'ajout du jeu de données à votre dossier "data" est simple. Naviguez simplement jusqu'au dossier dans l'onglet Files, puis cliquez sur le bouton "Upload":

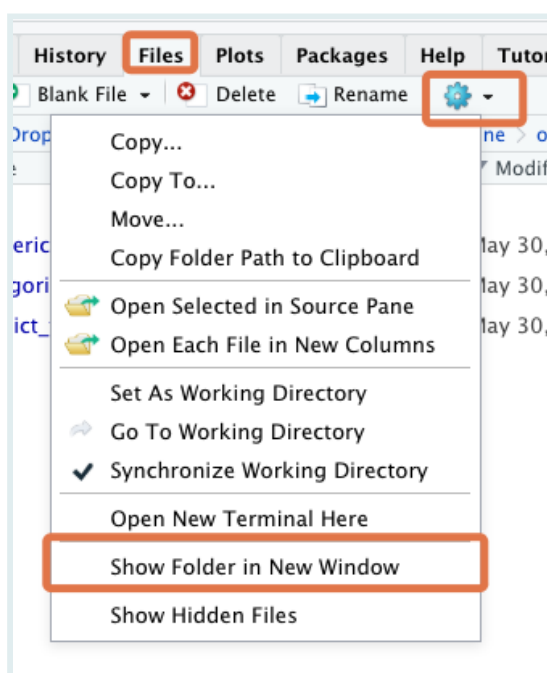


Cela fera apparaître une boîte de dialogue dans laquelle vous pourrez sélectionner le fichier à télécharger.

Sur un ordinateur local

Sur un ordinateur local, cette étape doit être effectuée avec l'Explorateur de fichiers/Finder de votre ordinateur.

- Tout d'abord, localisez le dossier project avec l'Explorateur de fichiers/Finder de votre ordinateur. Si vous rencontrez des difficultés pour le localiser, RStudio peut vous aider : allez dans l'onglet "Files", cliquez sur "More" (l'icône d'engrenage), puis cliquez sur "Show Folder in New Window".

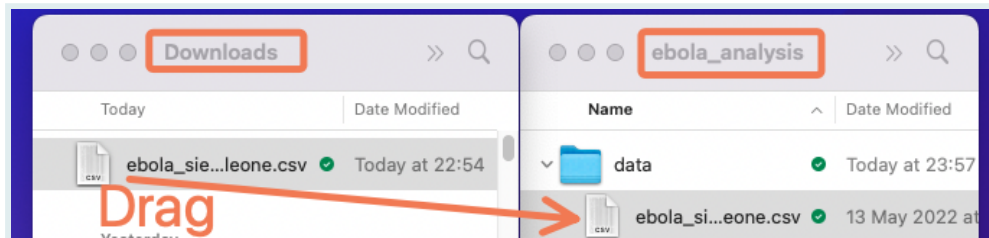


Vous accéderez ainsi au dossier Projet dans l'Explorateur de fichiers/Finder de votre ordinateur.

- Maintenant, déplacez le jeu de données Ebola que vous avez téléchargé dans la leçon précédente vers le sous-dossier “data” nouvellement créé.

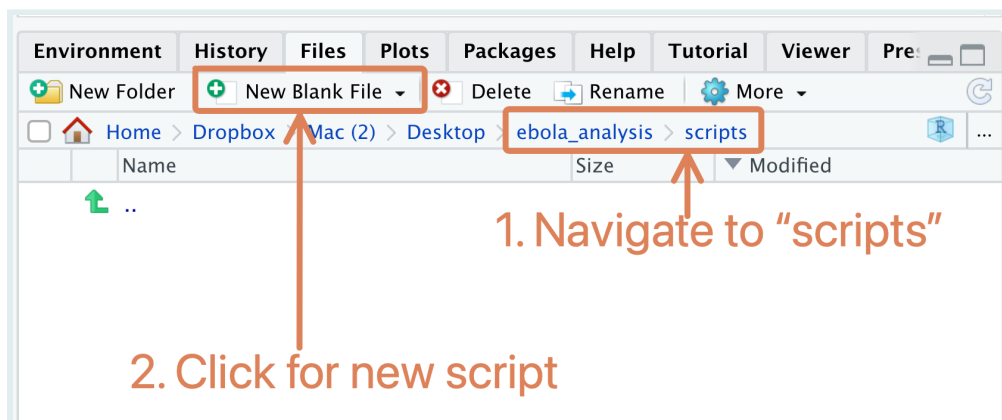
Voici à quoi ressemblerait le déplacement du fichier sur macOS :

##



Création d'un script dans le dossier “scripts”

Ensuite, créez et enregistrez un nouveau script R dans le dossier “scripts”. Vous pouvez appeler cela “analyse_principale” ou quelque chose de similaire. Pour créer un nouveau script R dans un dossier, accédez d'abord à ce dossier dans l'onglet Files, puis cliquez sur le bouton “New Blank File” et sélectionnez “Script R” dans la liste déroulante :



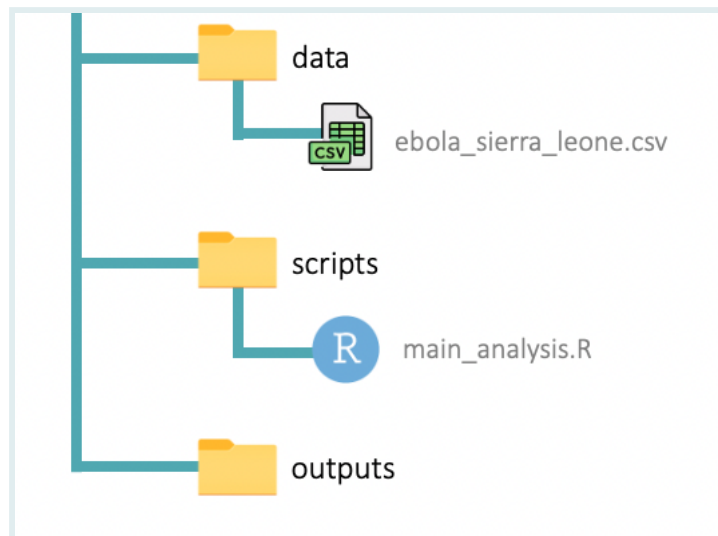
SIDE NOTE



Notez que ceci est différent de ce que vous avez fait jusqu'à présent lors de la création d'un nouveau script (avant, vous utilisiez l'option de menu “File > New File > New Script”). L'ancienne méthode est toujours valable; mais ce bouton “New Blank File” sera probablement plus rapide pour vous.

Excellent travail jusqu'à présent ! Maintenant, votre dossier project devrait avoir la structure ci-dessous, avec le jeu de données “ebola_sierra_leone.csv” dans le

dossier “data” et le script “analyse_principale.R” (toujours vide) dans le dossier “scripts”:



Il s’agit d’un processus que vous devez suivre au début de chaque projet d’analyse de données : créez un projet RStudio, créez les sous-dossiers nécessaires et placez vos jeux de données et vos scripts dans les sous-dossiers appropriés. Cela peut être un peu pénible, mais cela sera payant à long terme.

Le reste de cette leçon vous apprendra comment effectuer votre analyse dans le contexte de cette configuration ou creation de dossier. À la fin, vous aurez un flux global de données et de résultats qui ressemble au schéma ci-dessous :

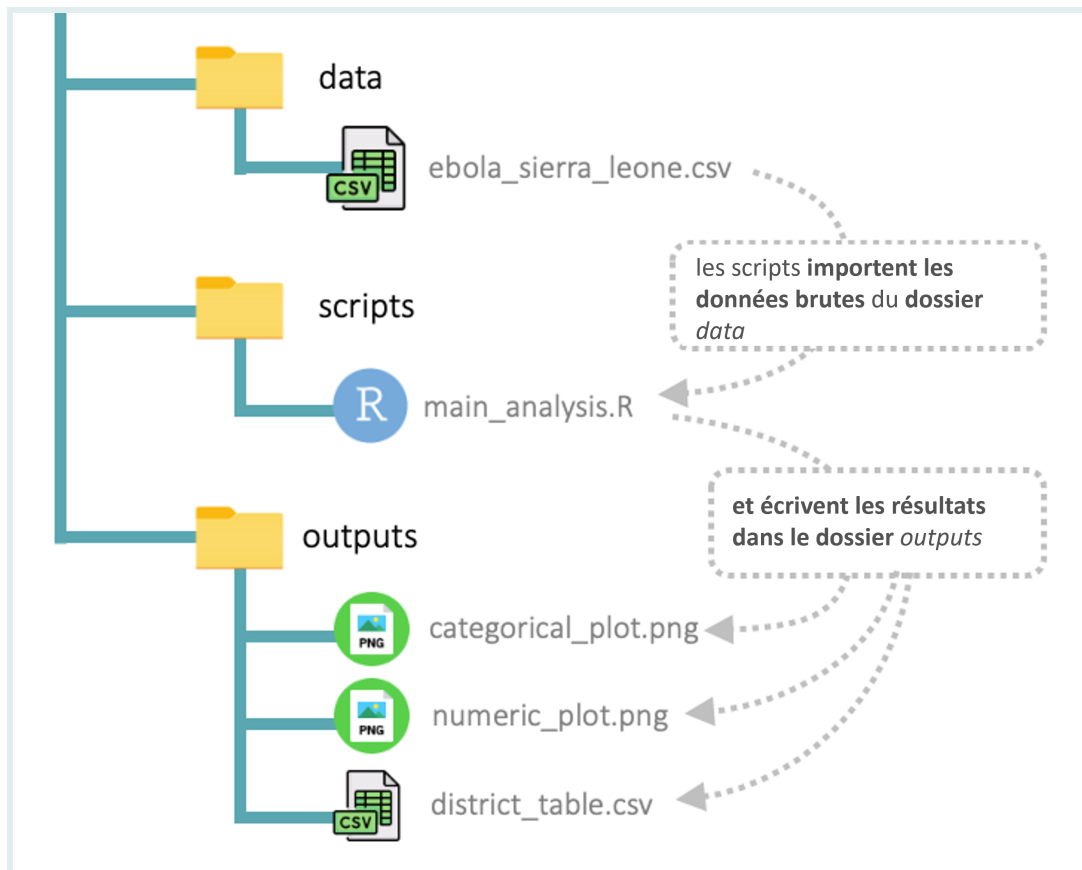


Figure : Flux de données dans un projet R. Les scripts du dossier “scripts” importent les données du dossier “data” et exportent les données et les graphiques vers le dossier “outputs”

Vous devriez vous référer à ce diagramme au fur et à mesure que vous avancez dans les sections ci-dessous, afin de vous aider à vous orienter.

Importation de données depuis le dossier “data”

Nous utiliserons l'extrait de code ci-dessous pour illustrer le flux de données dans un projet. Copiez et collez cet extrait dans votre script “analyse_principale.R” (mais ne l'exécutez pas encore). Le code reproduit certaines parties de l'analyse de la leçon sur l'exploration des données.

```

# se Ebola Sierra Leone
# Exemple-de-Nom Doe
# 01-01

# Charger les packages ----
install.packages("pacman")
pacman::p_load(
  dplyr,
  tidyr,
  readr,
  # nouveau package que nous utiliserons bientôt

# Charger les données ----
ebola_sierra_leone <- read_csv("") # DONNÉES EN ATTENTE ! NOUS LE METTRONS À JOUR CI-
  DESSOUS.

# Créer les départements ----
ebola_sierra_leone$departement <- tabyl(ebola_sierra_leone, district)
ebola_sierra_leone$departement

# Visualiser les variables nominales ----
ebola_sierra_leone$departement <- show_plot(inspect_cat(ebola_sierra_leone$departement))
ebola_sierra_leone$departement

# Visualiser les variables numériques ----
ebola_sierra_leone$departement <- show_plot(inspect_num(ebola_sierra_leone$departement))
ebola_sierra_leone$departement

```

Commencez par exécuter la section “Charger les packages” pour installer et/ou charger les packages nécessaires.

Passez ensuite à la section “Charger les données”, qui ressemble à ceci :

```

# Charger les données ----
ebola_sierra_leone <- read_csv("") # DONNÉES EN ATTENTE ! NOUS LE METTRONS À JOUR CI-
  DESSOUS.

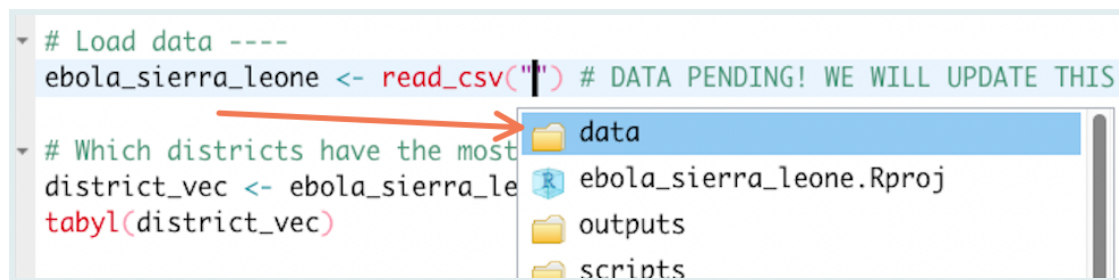
```

Ici, vous souhaitez importer le jeu de données Ebola que vous avez précédemment placé dans le dossier “data” du projet. Pour ce faire, vous devez fournir le chemin du fichier de ce jeu de données comme premier argument de `read_csv()`.

Comme vous utilisez un projet RStudio, ce chemin peut être obtenu très facilement : placez votre curseur entre les guillemets dans la fonction `read_csv()` et appuyez sur la touche `Tab` de votre clavier. Vous devriez voir une liste des sous-dossiers disponibles dans votre projet. Quelque chose comme ça :

```
# Load data ----
ebola_sierra_leone <- read_csv("") # DATA PENDING! WE WILL UPDATE THIS

# Which districts have the most
district_vec <- ebola_sierra_leone
tabyl(district_vec)
```



Cliquez sur le dossier “data”, puis appuyez à nouveau sur `Tab`. Puisque vous n’avez qu’un seul fichier dans le dossier “data”, RStudio devrait automatiquement remplir son nom. Vous devriez maintenant voir :

```
ebola_sierra_leone <- read_csv("data/ebola_sierra_leone.csv")
```

Merveilleux! Exécutez maintenant cette ligne de code pour importer les données.

Si cela réussit, vous devriez voir apparaître les données dans l’onglet Environment de RStudio :



Chemins relatifs

Le chemin que vous avez utilisé ici, “data/ebola_sierra_leone.csv”, est appelé un chemin *relatif*, car il est relatif à la *racine* (ou à la *base*) de votre projet.

KEY POINT



Comment R sait-il où se trouve la racine de votre projet ? C’est là qu’intervient le fichier `.Rproj`. Ce fichier, qui réside dans le dossier “analyse_ebola”, indique à R “ici ! Ici ! Je suis dans le dossier ‘analyse_ebola’, c’est donc la racine !”. Ainsi, vous n’avez qu’à spécifier les composants de chemin qui sont *plus profonds* que cette racine.

Les projets RStudio et les chemins relatifs qu’ils vous permettent d’utiliser sont importants pour la reproductibilité. Les projets qui utilisent des chemins relatifs peuvent être exécutés sur l’ordinateur de n’importe qui, et le code d’importation et d’exportation devrait fonctionner sans problème. Cela signifie que vous pouvez envoyer à quelqu’un un dossier de projet RStudio et que le code devrait s’exécuter sur sa machine comme il s’est exécuté sur la vôtre !

KEY POINT



Ce ne serait pas le cas si vous utilisiez un chemin *absolu*, quelque chose comme “~/Bureau/mon_analyse_de_donnees/formation_r/ebola_sierra_leone.csv”, dans votre script. Les chemins absolus donnent l’adresse complète d’un fichier et ne fonctionnent généralement pas sur l’ordinateur de quelqu’un d’autre, où les fichiers et les dossiers seront organisés différemment.



Notez que si vous utilisez RStudio Cloud, vous êtes *forcés* d’utiliser des chemins relatifs, car vous ne pouvez pas accéder au système de fichiers général de l’ordinateur virtuel ; vous ne pouvez travailler que dans des dossiers de projet spécifiques.

Utilisation de `here::here()`

Comme vous avez pu le constater, les projets RStudio simplifient le processus d’importation des données et améliorent la reproductibilité de votre analyse, principalement parce qu’ils vous permettent d’utiliser des chemins relatifs.

Mais il y a une étape supplémentaire que nous recommandons lorsque vous utilisez des chemins relatifs : plutôt que de laisser votre chemin *nu*, enveloppez-le dans la fonction `here()` du package `{here}`.

Ainsi, dans la section d’importation de données de votre script, modifiez l’entrée de `read_csv()` de `"data/ebola_sierra_leone.csv"` à `here("data/ebola_sierra_leone.csv")` :

```
ebola_leone <- read_csv(here("data/ebola_sierra_leone.csv"))
```

Quel est l’intérêt d’envelopper le chemin dans `here()` ? Eh bien, techniquement, il n’y a pas de réel intérêt à le faire dans un script *R* ; le code d’importation fonctionne bien sans cela. Mais cela *sera* nécessaire lorsque vous commencerez à utiliser les scripts *RMarkdown* (qui vous seront bientôt présentés), car les chemins non enveloppés dans `here()` sont problématiques dans le contexte *RMarkdown*.

Donc, pour garder les choses cohérentes, nous vous recommandons toujours d’utiliser `here()` pour pointer vers des chemins, que ce soit dans un script *R* ou un script *RMarkdown*.

Exportation des données vers le dossier “outputs”

L’importation de données n’est pas le seul avantage de RStudio Projects ; l’exportation de données est également simplifiée lorsque vous utilisez Projects. Voyons maintenant ce qu’il en est.

Dans la section “Cas par départements” de votre script, vous devriez avoir :

```
## Cas par départements ----
district_tab <- tabyl(ebola_sierra_leone, district)
district_tab
```

Exécutez ce code maintenant ; vous devriez obtenir le tableau suivant :

```
##      district      n percent
##      Bo          2    0.010
##      Kailahun 155    0.775
##      Kambia      1    0.005
##      Kenema      34    0.170
##      Kono         2    0.010
##      Port Loko    2    0.010
##      Western Urban 4    0.020
```

Maintenant, imaginez que vous souhaitiez exporter ce tableau au format CSV. Ce serait bien s’il existait un dossier spécifique désigné pour ces exportations. Eh bien, c’est le cas ! Il s’agit du dossier “outputs” que vous avez créé précédemment. Exportons votre tableau dans ce dossier. Saisissez le code ci-dessous (mais ne l’exécutez pas encore) :

```
write_csv(x = district_tab, file = "")
```

Avec la fonction `write_csv()`, vous allez “écrire” (ou “enregistrer”) le tableau `district_tab` sous forme de fichier CSV.

L’argument `x` de `write_csv()` prend en compte l’objet à sauvegarder (dans ce cas `district_tab`). Et l’argument `file` prend en compte le chemin du fichier cible. Ce chemin de fichier cible peut être un simple chemin relatif : “outputs/departement_tableau.csv”. (Et, comme mentionné précédemment, nous devrions envelopper le chemin dans `here()`.) Saisissez ceci et exécutez-le maintenant :

```
write_csv(x = district_tab, file = here("outputs/departement_tableau.csv"))
```

Le chemin “outputs/departement_tableau.csv” indique à `write_csv()` d’enregistrer le graphique dans un fichier CSV nommé “departement_tableau” dans le dossier “outputs” du projet.

SIDE NOTE



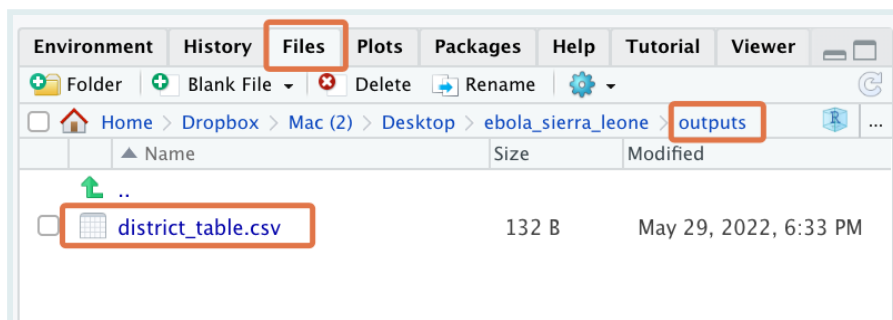
Vous pouvez remplacer “departement_tableau.csv” par tout autre nom approprié, par exemple “tableau de freq par départements.csv” :

SIDE NOTE

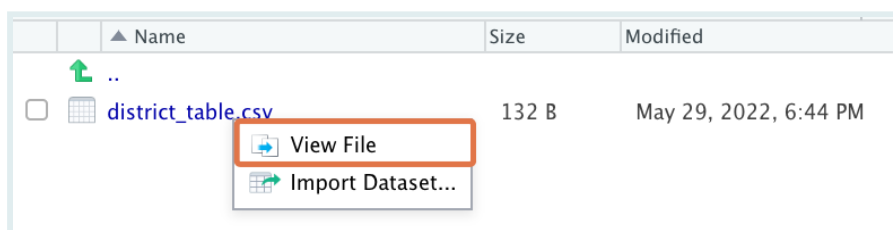


```
sv(x = district_tab, file = here("outputs/tableau de freq par  
départements.csv"))
```

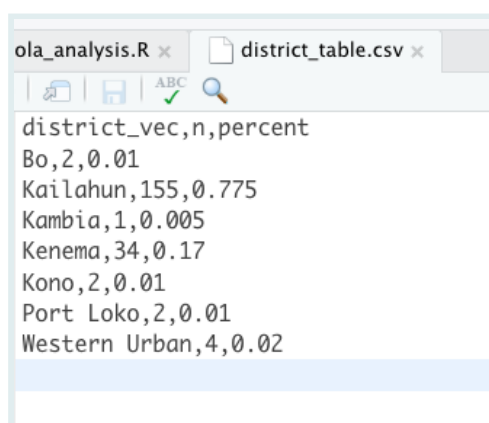
Bon travail! Maintenant, si vous allez dans l'onglet Files et naviguez jusqu'au dossier outputs de votre projet, vous devriez voir ce fichier nouvellement créé :



Vous pouvez cliquer sur le fichier pour le visualiser dans RStudio en tant que fichier CSV brut :



Cela devrait faire apparaître une fenêtre de visualisation RStudio :

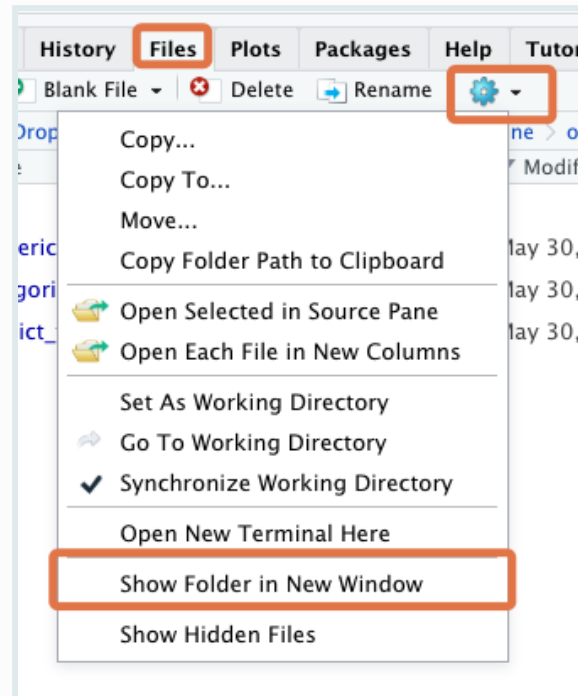


Si vous souhaitez plutôt visualiser le CSV dans Microsoft Excel, vous pouvez accéder au même fichier dans le Finder/Explorateur de fichiers de votre ordinateur et double-cliquer dessus à partir de là.

REMINDER

Pour localiser votre dossier de projet dans le Finder/Explorateur de fichiers de votre ordinateur, allez dans l'onglet "Files", cliquez sur l'icône d'engrenage, puis cliquez sur "Show Folder in New Window".

REMINDER



Si vous êtes sur RStudio cloud, vous ne pourrez pas visualiser le CSV dans Microsoft Excel tant que vous ne l'aurez pas "exporté". Utilisez l'option de menu "Export" dans l'onglet Files. Si cette option n'est pas immédiatement visible, cliquez sur l'icône d'engrenage pour afficher les options "More", puis faites défiler l'écran pour trouver l'option "Export".

Remplacement des données

Si vous avez besoin de mettre à jour le CSV de sortie, vous pouvez simplement réexécuter la fonction `write_csv()` avec l'objet de données mis à jour.

Pour tester cela, remplacez la section "Cas par département" de votre script par le code suivant. Il utilise la fonction `arrange()` pour organiser le tableau dans l'ordre du nombre de cas, `n` :

```
par département ----
c_tab <- tabyl(ebola_sierra_leone, district)
c_tab_arranged <- arrange(district_tab, -n)
c_tab_arranged
```

(`-n` signifie “trier par ordre décroissant de la variable `n`” ; nous vous présenterons correctement la fonction `arrange` plus tard.)

La sortie devrait être :

```
##      district    n percent
##      Kailahun 155   0.775
##      Kenema   34   0.170
##      Western Urban 4   0.020
##      Bo       2   0.010
##      Kono     2   0.010
##      Port Loko 2   0.010
##      Kambia   1   0.005
```

Vous pouvez maintenant remplacer l’ancien fichier “`departement_tableau.csv`” en réexécutant la fonction `write_csv` avec l’objet `district_tab_arrange` :

```
write_csv(x = district_tab_arrange, file = here("outputs/departement_tableau.csv"))
```

Pour vérifier que le jeu de données a bien été mis à jour, observez la section “Modified” dans l’onglet Files de RStudio :



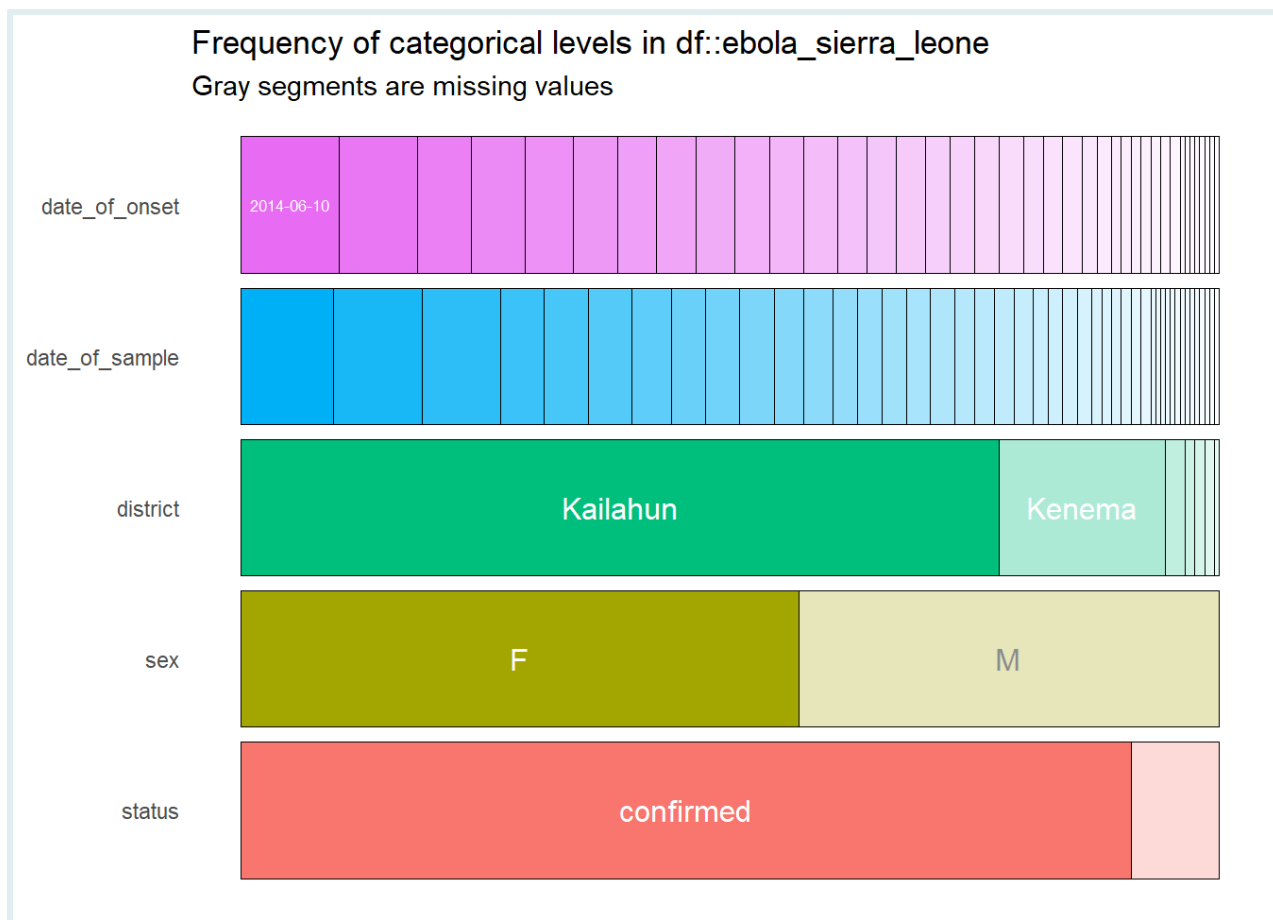
Exportation des graphiques vers le dossier “outputs”

Enfin, regardons l’exportation de graphiques dans le contexte d’un projet RStudio.

Dans la section “Visualiser les variables nominales” de votre script, vous devez avoir :

```
Visualiser les variables nominales ----
n_graph<- show_plot(inspect_cat(ebola_sierra_leone))
n_graph
```

L’exécution de ces lignes de code devrait vous donner ce résultat :



Sous ces lignes, tapez la commande `ggsave()` ci-dessous (mais ne l'exécutez pas encore) :

```
filename = "", plot = vars_nom_graph)
```

Cette commande utilise la fonction `ggsave()` pour exporter la figure `vars_nom_graph`. L'argument `plot` de `ggsave()` prend en compte l'objet à enregistrer (dans ce cas `vars_nom_graph`), et l'argument `filename` prend en compte le chemin du fichier cible pour le graphique.

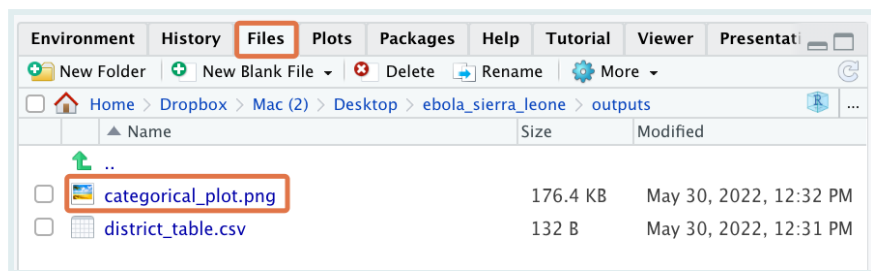
Comme vous l'avez vu lors de l'exportation de données, ce chemin de fichier cible est assez simple car vous travaillez dans un projet RStudio. Dans ce cas, vous avez :

```
filename = "outputs/graphique_nominal.png", plot = vars_nom_graph)
```

Exécutez maintenant cette commande `ggsave()`. Le chemin "outputs/graphique_nominal.png" indique à `ggsave()` d'enregistrer le graphique en tant que fichier PNG nommé "graphique_nominal" dans le dossier "outputs" du projet.

Pour voir ce graphique nouvellement enregistré, accédez à l'onglet Files. Vous pouvez cliquer dessus pour l'ouvrir avec le visualiseur d'images par défaut de votre

ordinateur :



Notez également que la fonction `ggsave()` vous permet d'enregistrer des graphiques dans plusieurs formats d'image. Par exemple, vous pourriez plutôt écrire :

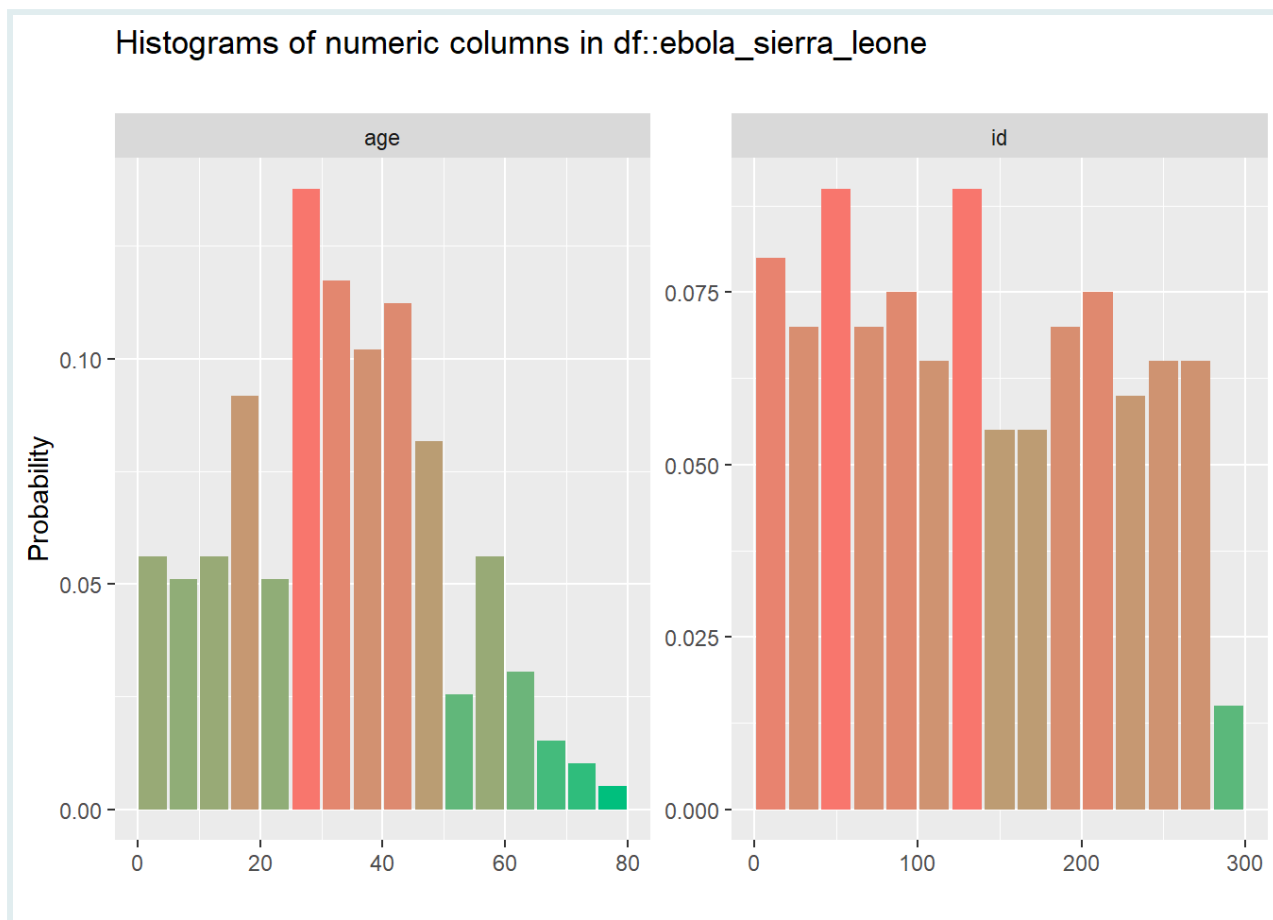
```
filename = "outputs/graphique_nominal.pdf", plot = vars_nom_graph)
```

pour enregistrer le graphique au format PDF. Exécutez `?ggsave` pour voir quels autres formats sont possibles.

Exportons maintenant le deuxième graphique, le résumé numérique. Dans la section de votre script intitulée “Visualiser les variables numériques”, vous devriez avoir :

```
Visualiser les variables numériques ----
n_graph <- show_plot(inspect_num(ebola_sierra_leone))
n_graph
```

L'exécution de ces lignes de code devrait vous donner ce résultat :



Pour exporter ce graphique, saisissez et exécutez le code suivant :

```
filename = "outputs/graph_numerique.png", plot = vars_num_graph)
```

Merveilleux!

Partager un projet

Les projets sont également parfaits pour partager votre analyse avec des collaborateurs.

Vous pouvez compresser votre dossier project et l'envoyer à un collègue par e-mail ou via un service de partage de fichiers comme Dropbox. Le collègue peut alors décompresser le dossier, cliquer sur le fichier .Rproj pour ouvrir le projet dans RStudio, et refaire et modifier toutes vos étapes d'analyse.

C'est une configuration convenable, mais l'envoi de projets dans les deux sens n'est pas toujours idéal pour une collaboration à long terme. C'est pourquoi les analystes expérimentés utilisent une technologie appelée *git* pour collaborer sur des projets. Mais ce sujet est un peu trop avancé pour ce cours ; nous le détaillerons dans un prochain cours. Si vous êtes impatient, vous pouvez consulter ce chapitre de livre : https://intro2r.com/github_r.html

Conclusion

Toutes nos félicitations! Vous savez maintenant comment configurer et utiliser les projets RStudio !

J'espère que vous voyez l'intérêt d'organiser vos scripts d'analyse, vos données et vos résultats de cette manière. Les projets sont une manière cohérente de structurer vos analyses et facilitent la réexamen, la révision et le partage de votre travail. Ils constitueront la base d'une grande partie de votre travail en tant qu'analyste de données à l'avenir.

C'est tout pour le moment. Rendez-vous à la prochaine leçon.

Contributeurs

Les membres de l'équipe suivants ont contribué à cette leçon :



KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about world improvement



LAURE NGUEMO

Data Science Education Officer
Gets very excited at the mention of data, especially health related data

Références

Certains éléments de cette leçon ont été adaptés à partir des sources suivantes :

- Wickham, H., & Grolemund, G. (n.d.). *R pour science des données*. 8 Workflow : projets | R pour Data Science. Extrait le 31 mai 2022 de <https://r4ds.had.co.nz/workflow-projects.html>

This work is licensed under the [Creative Commons Attribution Share Alike](#) license.

