Notes de leçon | Exploration de données: Ebola en Sierra Leone

GRAPH Network & OMS, soutenu par le Fonds Mondial

November 2023

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.

Objectifs d'apprentissage

1. Vous pouvez utiliser l'interface graphique de RStudio pour importer des données CSV dans R.

2. Vous pouvez expliquer le concept de reproductibilité.

3. Vous pouvez utiliser les fonctions `nrow()`, `ncol()` et `dim()` pour obtenir les dimensions d'un jeu de données, et la fonction `summary()` pour obtenir un résumé des variables du jeu de données.

4. Vous pouvez utiliser les fonctions `vis_dat()`, `inspect_num()` et `inspect_cat()` pour obtenir des résumés visuels d'un jeu de données.

5. Vous pouvez inspecter une variable numérique :

    - avec les fonctions récapitulatives `mean()` , `median()`, `max()`, `min()`, `length()` et `sum()` ;

    - avec le code ggplot2 généré par esquisse.

6. Vous pouvez inspecter une variable nominale :

    - avec les fonctions récapitulatives `table()` et `janitor::tabyl()` ;

- avec les fonctions graphiques `barplot()` et `pie()`.

Introduction

Grâce à vos connaissances nouvellement acquises sur les fonctions et les objets, vous disposez désormais des éléments de base nécessaires pour effectuer une analyse de données simple dans R. Alors commençons. L'objectif est de commencer à travailler avec des données le plus rapidement possible, avant même que vous ne vous sentiez prêt.

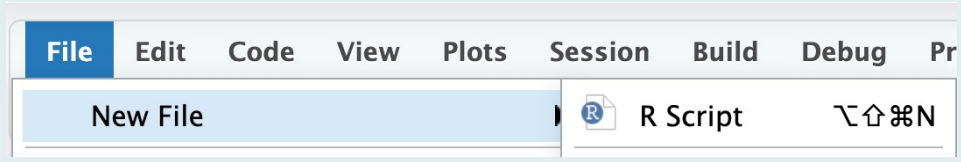
Ici, vous allez analyser un jeu de données sur les cas confirmés et suspects de fièvre hémorragique Ebola en Sierra Leone en mai et juin 2014 (Fang et al., 2016). Les données sont présentées ci-dessous :

Vous allez importer et explorer ce jeu de données, puis utiliser R pour répondre aux questions suivantes sur l'épidémie :

- **Quand le premier cas a-t-il été signalé ?**
- **Quel était l'âge médian des personnes concernées ?**
- **Les cas ont-ils été plus nombreux chez les hommes ou chez les femmes ?**
- **Quel district avait eu le plus de cas signalés ?**
- **À la fin du mois de juin 2014, l'épidémie était-elle en croissance ou en regression ?**

Configuration du script

Tout d'abord, ouvrez un nouveau script dans RStudio avec `File > New File > R Script`. (Si vous êtes sur RStudio, vous pouvez ouvrir n'importe lequel de vos projets créés précédemment.)



Ensuite, enregistrez le script avec `File> Save as` ou appuyez sur `Commande/Contrôle` + S pour afficher la boîte de dialogue Enregistrer le fichier. Enregistrez le fichier sous le nom "ebola_analysis" ou quelque chose de similaire

SIDE NOTE

Videz votre environnement au début de l'analyse

SIDE NOTE

Lorsque vous démarrez une nouvelle analyse, votre environnement R devrait normalement être vide. Vérifiez cela en ouvrant l'onglet *Environment* ; il devrait dire "Environment is empty". Si au contraire, il affiche des objets précédemment chargés, il est recommandé de redémarrer R en allant dans l'option de menu `Session > Restart R`

En-tête

Ajoutez un titre, un nom et une date au début du script, sous forme de commentaires de code. Il s'agit généralement d'une bonne pratique pour l'écriture de scripts R, car cela vous aide à vous donner, à vous et à vos collaborateurs, le contexte de votre script. Votre en-tête peut ressembler à ceci :

```
# Analyse Ebola Sierra Leone
# John Example-de-Nom Doe
# 2024-01-01
```

Les packages

Ensuite, utilisez la fonction `p_load()` de {pacman} pour charger les packages que vous utiliserez. Placez ceci sous un en-tête de section appelé "Load packages", avec quatre traits d'union, comme indiqué ci-dessous :

```
# Charger les packages ----
if(!require(pacman)) install.packages("pacman")
pacman::p_load(
  tidyverse, # méta-package
  inspectdf,
  plotly,
  janitor,
  visdat,
  esquisse
)
```

REMINDER

N'oubliez pas que le *signifiant complet* d'une fonction inclut à la fois le nom du package et le nom de la fonction, `package::function()`. Ce signifiant complet est pratique si vous souhaitez utiliser une fonction avant d'avoir chargé son package source. C'est le cas dans le morceau de code ci-dessus : nous voulons utiliser `p_load()` de {pacman} sans charger formellement le package {pacman}, nous saisissons donc `pacman::p_load()`

Nous pourrions aussi d'abord charger {pacman} avant d'utiliser la fonction p_load :

REMINDER

```
library(pacman) # charge d'abord {pacman}
p_load(tidyverse) # utilise `p_load` de {pacman} pour charger d'autres packages
```

(Rappelez-vous également que l'avantage de `p_load()` est qu'il installe automatiquement un package s'il n'est pas encore installé. Sans `p_load()`, vous devez d'abord installer le package avec `install.packages()` avant de pouvoir le charger avec `library()`.)

Importer des données dans R

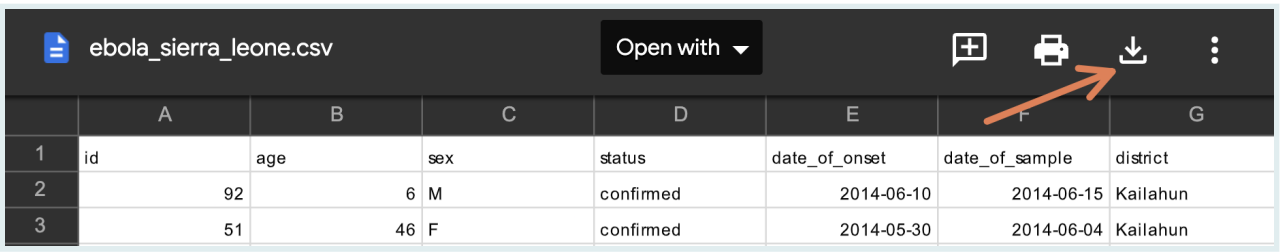
Maintenant que les packages nécessaires sont chargés, vous devez importer le jeu de données.

SIDE NOTE

À propos du jeu de données Ebola

Les données sur lesquelles vous allez travailler contiennent un échantillon d'informations sur les patients de l'épidémie d'Ebola de 2014-2016 en Sierra Leone. Elles proviennent d'un document de recherche qui a analysé la dynamique de transmission de cette épidémie. Les variables clés incluent le `statut` d'un cas, si le cas a été `confirmé` ou `suspecté` ; la `date d'apparition`, lorsque des symptômes de type Ebola sont apparus chez un patient ; et la `date_de_l'échantillon`, date à laquelle l'échantillon de test a été prélevé. Pour en savoir plus sur ces données, consultez la publication source ici : bit.ly/ebola-data-source. Ou recherchez le DOI suivant sur DOI.org : 10.1073/pnas.1518587113.

Accédez à bit.ly/view-ebola-data pour afficher le jeu de données sur lequel vous allez travailler. Cliquez ensuite sur l'icône de téléchargement en haut de la page pour le télécharger sur votre ordinateur.



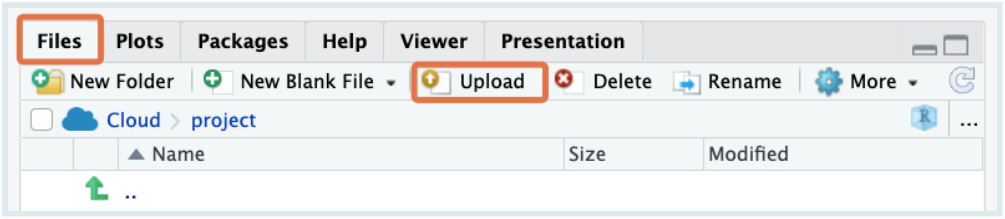
	A	B	C	D	E	F	G
1	id	age	sex	status	date_of_onset	date_of_sample	district
2	92	6	M	confirmed	2014-06-10	2014-06-15	Kailahun
3	51	46	F	confirmed	2014-05-30	2014-06-04	Kailahun

Vous pouvez laisser le jeu de données dans votre dossier de téléchargements ou le déplacer vers un endroit plus respectable ; les prochaines étapes fonctionneront

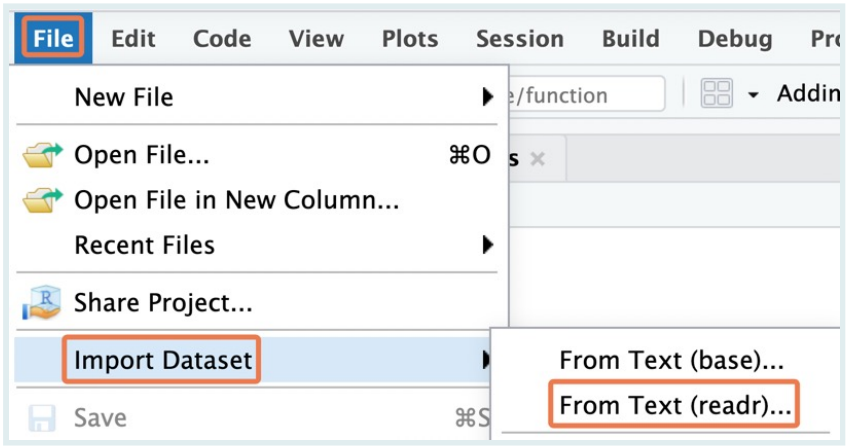
indépendamment de l'endroit où les données sont stockées. Dans la leçon suivante, vous apprendrez à organiser correctement vos projets d'analyse de données, et nous réfléchirons à la configuration idéale des dossiers pour le stockage des données.

RSTUDIO CLOUD

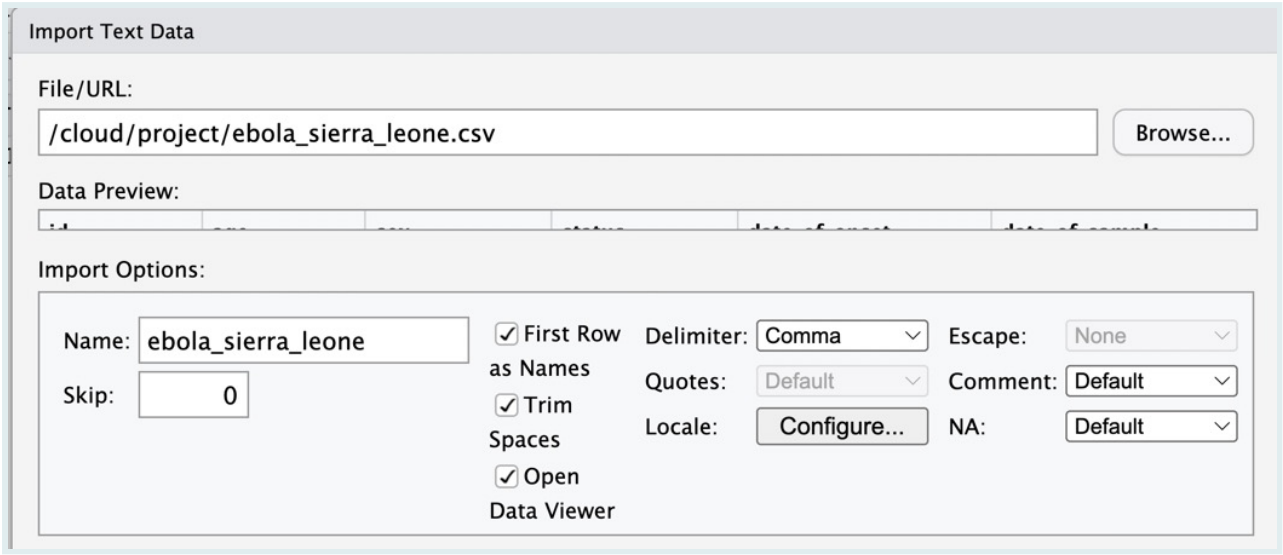
REMARQUE : Si vous utilisez RStudio Cloud, vous devez télécharger votre jeu de données sur le cloud. Faites cela dans l'onglet "Files" en cliquant sur le bouton "Upload".



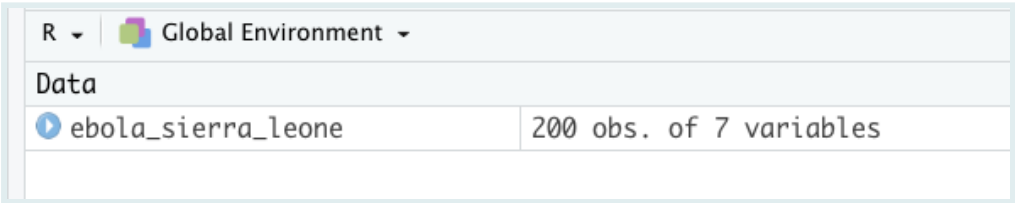
Ensuite, dans le menu de RStudio, allez dans `File > Import Dataset > From Text (readr)`.



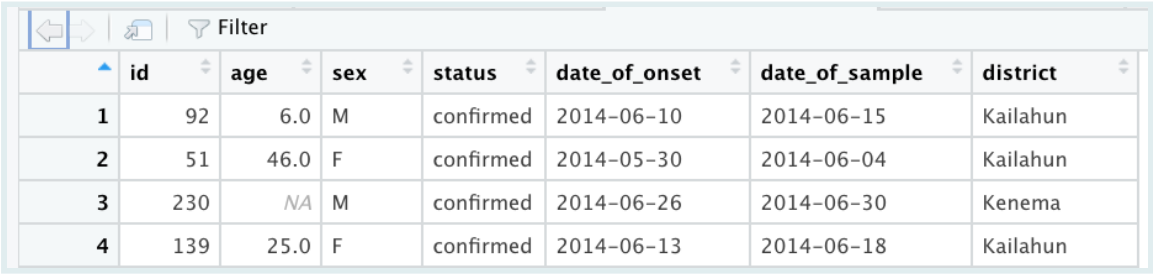
Parcourez les fichiers de l'ordinateur et naviguez jusqu'au jeu de données téléchargé. Cliquez pour l'ouvrir. Vous devriez voir une boîte de dialogue d'importation comme celle-ci :



Laissez tous les paramètres d'importation aux valeurs par défaut ; cliquez simplement sur "Import" en bas ; cela devrait charger le jeu de données dans R. Vous pouvez vous en rendre compte en regardant votre volet d'environnement, qui devrait maintenant comporter un objet appelé "ebola_sierra_leone" ou quelque chose de similaire :



RStudio devrait également avoir appelé la fonction `View()` sur votre jeu de données, vous devriez donc voir une vue familière de ces données sur une feuille de calcul :



	id	age	sex	status	date_of_onset	date_of_sample	district
1	92	6.0	M	confirmed	2014-06-10	2014-06-15	Kailahun
2	51	46.0	F	confirmed	2014-05-30	2014-06-04	Kailahun
3	230	*NA*	M	confirmed	2014-06-26	2014-06-30	Kenema
4	139	25.0	F	confirmed	2014-06-13	2014-06-18	Kailahun

Jetez maintenant un coup d'œil à votre console. Observez-vous que vos actions dans l'interface graphique ont en fait déclenché l'exécution d'un code R ? Copiez la ligne de code qui inclut la fonction `read_csv()`, en omettant le symbole >.

Collez le code copié dans votre script R et intitulez cette section “Charger les données”. Cela peut ressembler à ce qui suit (le chemin du fichier entre guillemets diffère d'un ordinateur à l'autre).

```
# Charger les données ----
ebola_sierra_leone <- read_csv("~/Téléchargements/ebola_sierra_leone.csv")
```

RECAP

Beau travail jusqu'à présent!

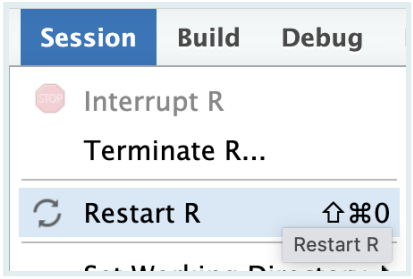
Votre script R devrait ressembler à ceci :

```
# Analyse Ebola Sierra Leone
# John Example-de-Nom Doe
# 2024-01-01

# Charger les  packages ----
if(!require(pacman)) install.packages("pacman")
pacman::p_load(
  tidyverse,
  inspectdf,
  plotly,
  janitor,
  visdat
)

# Charger les données ----
ebola_sierra_leone <-
read_csv("~/Téléchargements/ebola_sierra_leone.csv")
```

Introduction à la reproductibilité

Maintenant que le code d'importation des données se trouve dans votre script R, vous pouvez facilement réexécuter ce script à tout moment pour réimporter le jeu de données ; il ne sera pas nécessaire de refaire la procédure manuelle de pointer-cliquer pour l'importation des données.

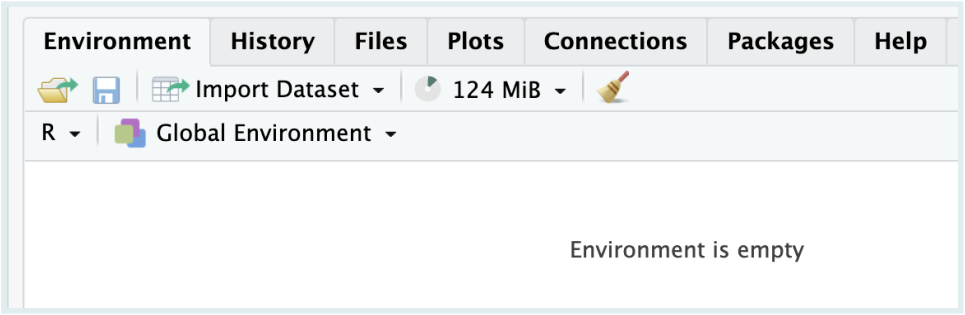
Essayez de redémarrer R et de réexécuter le script maintenant. Enregistrez votre script avec `Contrôle/Commande + s` , puis *redémarrez* R avec le menu RStudio, à `Session > Restart R`. Sur RStudio Cloud, l'option de menu ressemble à ceci :



Si le redémarrage réussit, votre console devrait afficher ce message :

```
Restarting R session...

>
```

Vous devriez également voir la phrase “Environment is empty” dans l’onglet Environnement, indiquant que le jeu de données que vous avez importé n’est plus stocké par R – vous commencez avec un nouvel espace de travail.



Pour relancer votre script, utilisez `Commande/Contrôle` + `a` pour surligner tout le code, puis `Commande/Contrôle` + `Enter` pour l’exécuter.

Si cela a fonctionné, félicitations ; vous avez les prémices de votre premier script d’analyse “reproductible” !

Que signifie “reproductible” ?

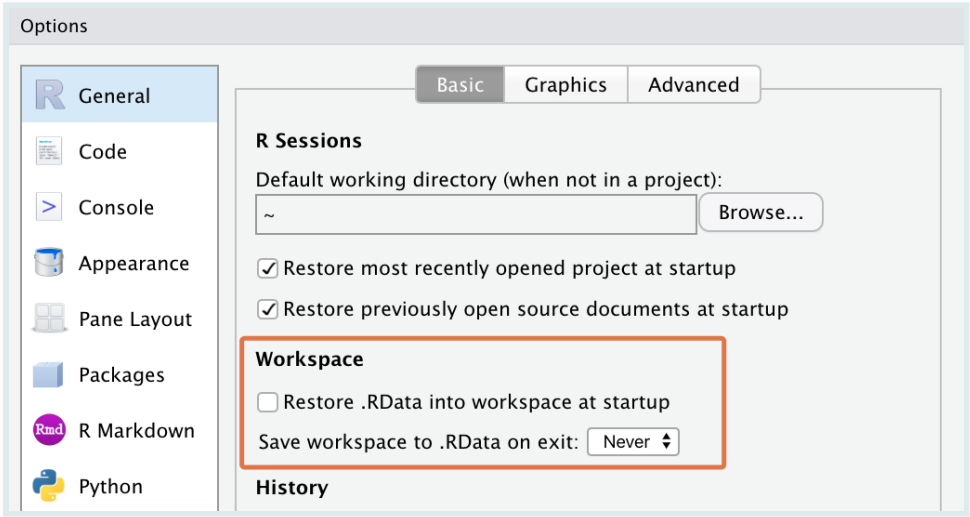
Lorsque vous faites des choses avec du code plutôt qu’en pointant et en cliquant, il est facile pour n’importe qui de réexécuter, ou de *reproduire* ces étapes, en réexécutant simplement votre script.

Bien que vous puissiez utiliser l’interface graphique de RStudio pour pointer et cliquer tout au long du processus d’importation de données, vous devez toujours copier le code pertinent dans votre script afin que votre script reste un enregistrement reproductible de toutes vos étapes d’analyse.

Bien sûr, votre script jusqu'à présent n'est pas encore *entièrement* reproductible, car le chemin dd'accès au jeu de données (celui qui ressemble à ceci : "...intro-a-lanalyse-de-donnees-avec-r/ch01_pour_commencer/data. .."") est spécifique à votre ordinateur uniquement. Plus tard, nous verrons comment utiliser des chemins d'accès relatifs, afin que le code d'importation de données puisse fonctionner sur n'importe quel ordinateur.

Si votre environnement n'était pas vide après le redémarrage de R, cela signifie que vous avez sauté une étape dans une leçon précédente. Faites ceci maintenant :

- Dans le menu de RStudio, allez à `Tools > Global Options` pour afficher la boîte de dialogue des options de RStudio.
- Ensuite, allez dans `General> Basic` et **décochez** la case qui dit "Restore .RData into workspace at startup".
- Pour l'option "save your workspace to .RData on exit", réglez-la sur "Never".



Exploration rapide des données

Passons maintenant en revue certaines étapes de base de l'exploration des données, en jetant un coup d'œil général sur le jeu de données. Vous devez placer cette section sous un titre tel que "Explorer les données" dans votre script.

Pour afficher les 6 lignes supérieures et inférieures du jeu de données, vous pouvez utiliser les fonctions `head()` et `tail()` :

```
# Explorer les données ----
head(ebola_sierra_leone)
```

```
## # A tibble: 6 × 7
##      id   age sex   status    date_of_onset date_of_sample
##   <dbl> <dbl> <chr> <chr>     <date>        <date>
## 1    92     6 M     confirmed 2014-06-10    2014-06-15
## 2    51    46 F     confirmed 2014-05-30    2014-06-04
## 3   230    NA M     confirmed 2014-06-26    2014-06-30
## 4   139    25 F     confirmed 2014-06-13    2014-06-18
## 5     8     8 F     confirmed 2014-05-22    2014-05-27
## 6   215    49 M     confirmed 2014-06-24    2014-06-29
## # ℹ 1 more variable: district <chr>
```

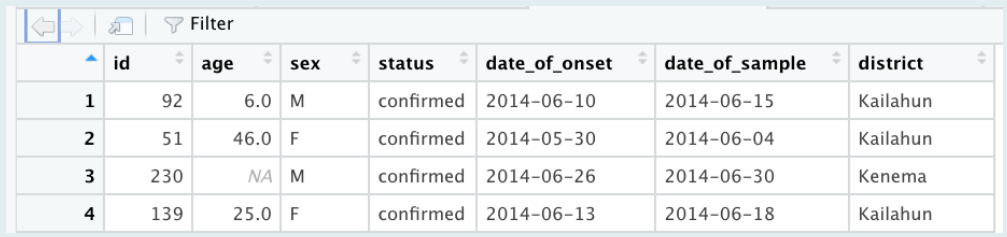
```
tail(ebola_sierra_leone)
```

```
## # A tibble: 6 × 7
##      id   age sex   status    date_of_onset date_of_sample
##   <dbl> <dbl> <chr> <chr>     <date>        <date>
## 1   214     6 F     confirmed 2014-06-24    2014-06-30
## 2    28    45 F     confirmed 2014-05-27    2014-06-01
## 3    12    27 F     confirmed 2014-05-22    2014-05-27
## 4   110     6 M     confirmed 2014-06-10    2014-06-15
## 5   209    40 F     confirmed 2014-06-24    2014-06-27
## 6    35    29 M     suspected 2014-05-28    2014-06-01
## # ℹ 1 more variable: district <chr>
```

Pour afficher l'intégralité du jeu de données, utilisez la fonction `View()`.

```
View(ebola_sierra_leone)
```

Cette fonction ouvrira à nouveau une vue familière des données sous forme de feuille de calcul :



Filter

	id	age	sex	status	date_of_onset	date_of_sample	district
1	92	6.0	M	confirmed	2014-06-10	2014-06-15	Kailahun
2	51	46.0	F	confirmed	2014-05-30	2014-06-04	Kailahun
3	230	NA	M	confirmed	2014-06-26	2014-06-30	Kenema
4	139	25.0	F	confirmed	2014-06-13	2014-06-18	Kailahun

Vous pouvez fermer cet onglet et revenir à votre script.

Les fonctions `nrow()`, `ncol()` et `dim()` vous donnent les dimensions de votre jeu de données :

```
nrow(ebola_sierra_leone) # nombre de lignes
```

```
## [1] 200
```

```
ncol(ebola_sierra_leone) # nombre de colonnes
```

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

```
dim(ebola_sierra_leone) # nombre de lignes et de colonnes
```

```
## [1] 200   7
```

REMINDER

Si vous n'êtes pas sûr de ce que fait une fonction, n'oubliez pas que vous pouvez obtenir de l'aide sur la fonction avec le symbole du point d'interrogation. Par exemple, pour obtenir de l'aide sur la fonction `ncol()`, exécutez :

```
?ncol
```

Une autre fonction souvent utile est `summary()` :

```
summary(ebola_sierra_leone)
```

```
##        id              age            sex
##  Min.   :  1.00   Min.   : 1.80   Length:200
##  1st Qu.: 62.75   1st Qu.:20.00   Class :character
```

```
##  Median :131.50   Median :35.00   Mode  :character
##  Mean   :136.72   Mean   :33.85
##  3rd Qu.:208.25   3rd Qu.:45.00
##  Max.   :285.00   Max.   :80.00
##                   NA's   :4
##     status          date_of_onset        date_of_sample
##  Length:200         Min.   :2014-05-18   Min.   :2014-05-23
##  Class :character   1st Qu.:2014-06-01   1st Qu.:2014-06-07
##  Mode  :character   Median :2014-06-13   Median :2014-06-18
##                     Mean   :2014-06-12   Mean   :2014-06-17
##                     3rd Qu.:2014-06-23   3rd Qu.:2014-06-29
##                     Max.   :2014-06-29   Max.   :2014-07-17
##
##    district
##  Length:200
##  Class :character
##  Mode  :character
##
##
##
##
```

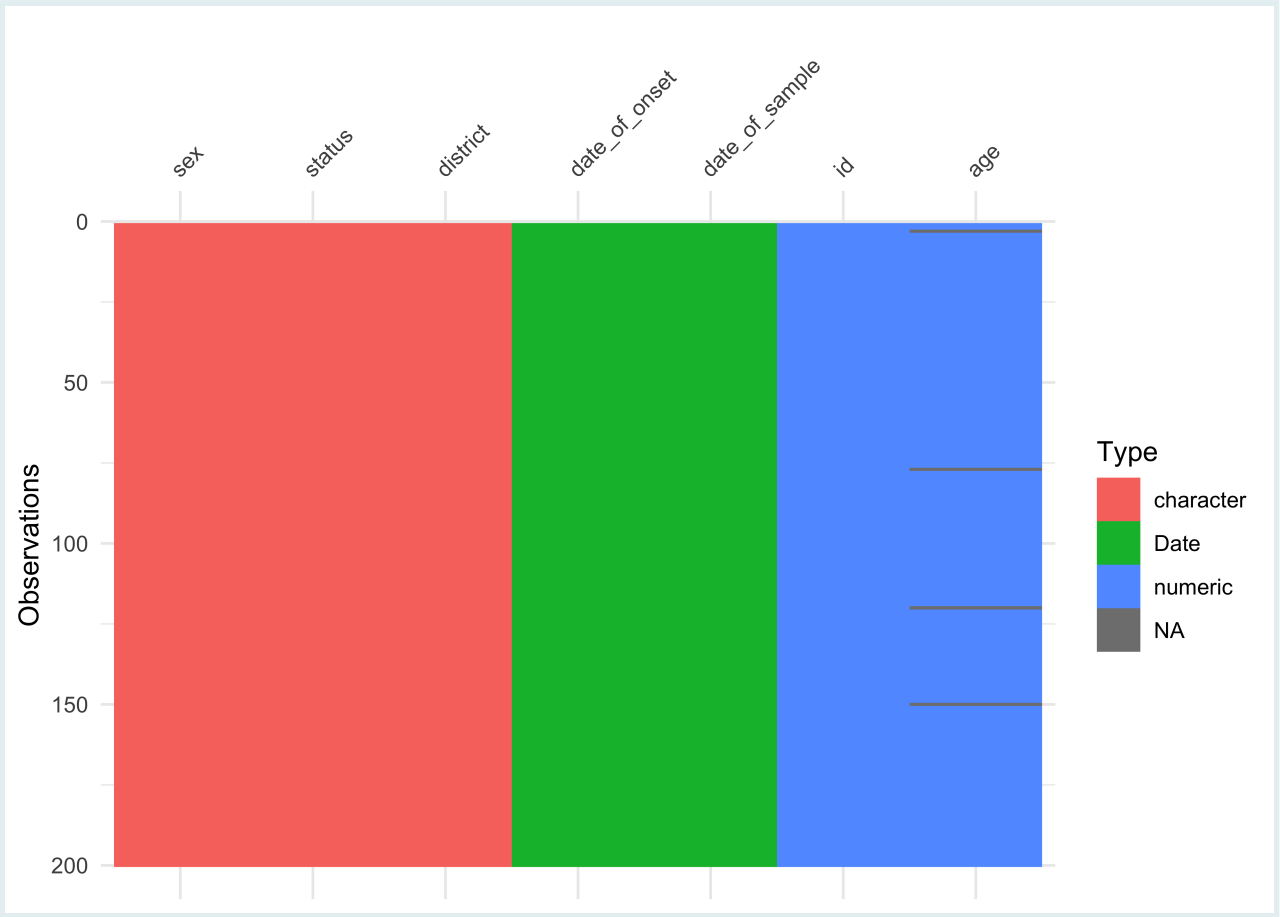
Comme vous pouvez le voir, pour les colonnes numériques de votre jeu de données, `summary()` vous donne la valeur minimale, la valeur maximale, la moyenne, la médiane et les 1er et 3e quartiles.

Pour les colonnes de caractères, il vous donne juste la longueur de la colonne (le nombre de lignes), la "classe" et le "mode". Nous discuterons de ce que "classe" et "mode" signifient plus tard.

`vis_dat()`

La fonction `vis_dat()` du package {visdat} est un excellent moyen de visualiser rapidement les types de données et les valeurs manquantes dans un jeu de données. Essayez ceci maintenant :

```
vis_dat(ebola_sierra_leone)
```



À partir de cette figure, vous pouvez voir rapidement les types de données caractère, date et numérique, et vous pouvez noter que l'âge est manquant dans certains cas.

`inspect_cat()` et `inspect_num()`

Ensuite, `inspect_cat()` et `inspect_num()` du package {inspectdf} vous donnent des résumés visuels de la distribution des variables dans le jeu de données.

Si vous exécutez `inspect_cat()` sur l'objet de données, vous obtenez un résumé tabulaire des variables nominales dans le jeu de données , avec quelques informations cachées dans la colonne `levels` (vous apprendrez plus tard comment extraire ces informations).

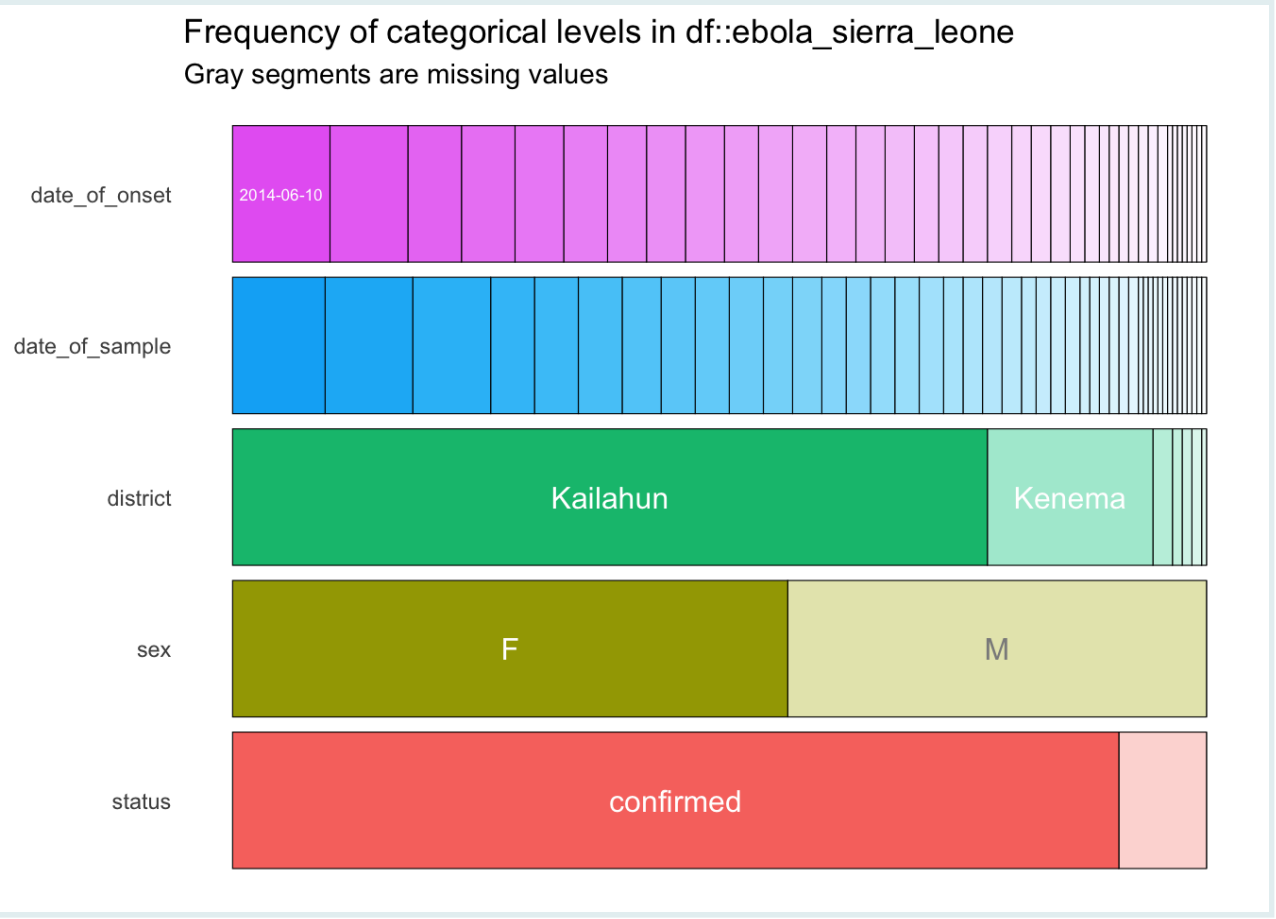
```
inspect_cat(ebola_sierra_leone)
```

```
## # A tibble: 5 × 5
##   col_name         cnt common     common_pcnt levels
##   <chr>          <int> <chr>            <dbl> <named list>
## 1 date_of_onset     39 2014-06-10        10   <tibble>
## 2 date_of_sample    45 2014-06-15         9.5 <tibble>
## 3 district           7 Kailahun          77.5 <tibble>
```

```
## 4 sex                2 F                57    <tibble>
## 5 status             2 confirmed        91    <tibble>
```

Mais la magie opère lorsque vous exécutez `show_plot()` sur le résultat de `inspect_cat()` :

```
# stocker la sortie de `inspect_cat()` dans `resume_cat`
resume_cat <- inspect_cat(ebola_sierra_leone)

# appelle la fonction `show_plot()` sur ce résumé.
show_plot(resume_cat)
```



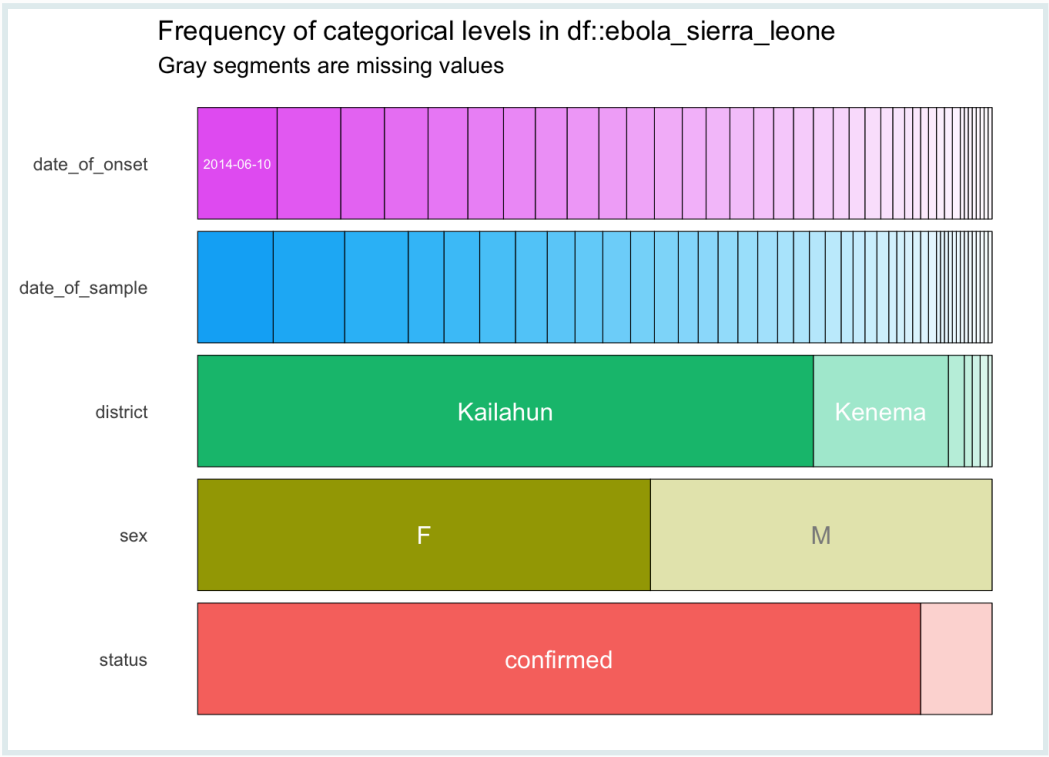
Vous obtenez une figure magnifique montrant la distribution de toutes les variables nominales et de date !

SIDE NOTE

Vous pouvez également exécuter :

```
show_plot(inspect_cat(ebola_sierra_leone))
```

SIDE NOTE



À partir de ce graphique, vous pouvez rapidement dire que la plupart des cas se trouvent à Kailahun et qu'il y a plus de cas chez les femmes que chez les hommes ("F" signifie "femelle").

Un problème est que dans ce graphique, les plus petites catégories ne sont pas étiquetées. Ainsi, par exemple, nous ne savons pas quelle valeur est représentée par la section blanche pour "statut" en bas à droite. Pour voir les étiquettes sur ces catégories plus petites, vous pouvez transformer cela en un graphique interactif avec la fonction `ggplotly()` du package {plotly}.

```
resume_cat_graph <- show_plot(resume_cat)
ggplotly(resume_cat_graph)
```

Merveilleux! Vous pouvez maintenant survoler chacune des barres pour voir la proportion de chaque section de la barre. Par exemple, vous pouvez maintenant dire que 9 % (0,090) des cas ont un statut suspect :

REMINDER

La flèche d'affectation, <–, peut être écrite avec le raccourci RStudio **alt** + **–** (**alt** ET **moins**) sous Windows ou **option** + **–** (**option** ET **moins**) sur macOS.

Vous pouvez obtenir un graphique similaire pour les variables numériques (continues) dans le jeu de données avec `inspect_num()`. Ici, nous montrons les trois étapes en une seule fois.

```
resume_num <- inspect_num(ebola_sierra_leone)
resume_num_graph <- show_plot(resume_num)
ggplotly(resume_num_graph)
```

Cela vous donne un aperçu des colonnes numériques, `age` et `id`. (Bien sûr, la distribution de la variable `id` n'est pas significative.)

Vous pouvez constater que les personnes âgées de 35 à 40 ans (point médian 37,5) constituent le groupe d'âge le plus important, représentant 13,8 % (0,1377…) des cas dans le jeu de données.

Analyse d'une seule variable numérique

Maintenant que vous avez une idée de ce à quoi ressemble le jeu de données, vous pouvez isoler et analyser une seule variable à la fois – c'est ce qu'on appelle *l'analyse univariée*.

Allez-y et créez une nouvelle section dans votre script pour cette analyse univariée.

```
# Analyse univariée, variables numériques ----
```

Commençons par analyser la variable numérique "age".

Extraire un vecteur colonne avec $

Pour extraire une seule variable/colonne d'un jeu de données, utilisez le signe du dollar, l'opérateur $:

```
ebola_sierra_leone$age # extrait la colonne 'age' dans le jeu de données
```

```
##   [1]  6.0 46.0   NA 25.0  8.0 49.0 13.0 50.0 35.0 38.0 60.0 18.0
##  [13] 10.0 14.0 50.0 35.0 43.0 17.0  3.0 60.0 38.0 41.0 49.0 12.0
##  [25] 74.0 21.0 27.0 41.0 42.0 60.0 30.0 50.0 50.0 22.0 40.0 35.0
##  [37] 19.0  3.0 34.0 21.0 73.0 65.0 30.0 70.0 12.0 15.0 42.0 60.0
##  [49] 14.0 40.0 33.0 43.0 45.0 14.0 14.0 40.0 35.0 30.0 17.0 39.0
##  [61] 20.0  8.0 40.0 42.0 53.0 18.0 40.0 20.0 45.0 40.0 60.0 44.0
```

```
##  [73] 33.0 23.0 45.0  7.0   NA 35.0 36.0 42.0 35.0 25.0 30.0 30.0
##  [85] 28.0 14.0 20.0 60.0 67.0 35.0 50.0  4.0 28.0 38.0 30.0 26.0
##  [97] 37.0 30.0  3.0 56.0 32.0 35.0 54.0 42.0 48.0 11.0  1.8 63.0
## [109] 55.0 20.0 62.0 62.0 42.0 65.0 29.0 20.0 33.0 30.0 35.0   NA
## [121] 50.0 16.0  3.0 22.0  7.0 50.0 17.0 40.0 21.0  9.0 27.0 52.0
## [133] 50.0 25.0 10.0 30.0 32.0 38.0 30.0 50.0 26.0 35.0  3.0 50.0
## [145] 60.0 40.0 34.0  4.0 42.0   NA 54.0 18.0 45.0 30.0 35.0 35.0
## [157] 16.0 26.0 23.0 45.0 45.0 45.0 38.0 45.0 35.0 30.0 60.0  5.0
## [169] 18.0  2.0 70.0 35.0  3.0 30.0 80.0 62.0 20.0 45.0 18.0 28.0
## [181] 48.0 38.0 39.0 26.0 60.0 35.0 20.0 50.0 11.0 36.0 29.0 57.0
## [193] 35.0 26.0  6.0 45.0 27.0  6.0 40.0 29.0
```

Cette liste de valeurs est appelée un *vecteur* dans R. Un vecteur est une sorte de structure de données dont les éléments dont d'un seul *type*. Dans ce cas, le type est "numérique". Nous vous présenterons formellement les vecteurs et autres structures de données dans un prochain chapitre. Dans cette leçon, vous pouvez considérer que "vecteur" et "variable" sont synonymes.

Opérations de base sur une variable numérique

Pour obtenir la moyenne de ces âges, vous pouvez exécuter :

```
mean(ebola_sierra_leone$age)
```

```
## [1] NA
```

Mais il semble que nous ayons un problème. R dit que la moyenne est 'NA', ce qui signifie "non applicable" ou "non disponible". C'est parce qu'il y a des valeurs manquantes dans le vecteur des âges. (Avez-vous remarqué cela lorsque vous avez imprimé le vecteur ?) Par défaut, R ne peut pas trouver la moyenne s'il y a des valeurs manquantes. Pour ignorer ces valeurs, utilisez l'argument `na.rm` (qui signifie "Supprimer NA") en lui donnant la valeur T ou TRUE :

```
mean(ebola_sierra_leone$age, na.rm = T)
```

```
## [1] 33.84592
```

Super! Cette nécessité de supprimer les "NA" avant de calculer une statistique s'applique à de nombreuses fonctions. La fonction `median()`, par exemple, renverra également NA par défaut si elle est appelée sur un vecteur avec des NA :

```
median(ebola_sierra_leone$age) # ne fonctionne pas
```

```
## [1] NA
```

```
median(ebola_sierra_leone$age, na.rm = T) # fonctionne
```

```
## [1] 35
```

`mean` et `median` ne sont que deux des nombreuses fonctions R qui peuvent être utilisées pour inspecter une variable numérique. Regardons quelques autres.

Mais d'abord, nous pouvons assigner le vecteur d'âge à un nouvel objet, de sorte que vous n'ayez pas à saisir `ebola_sierra_leone$age` à chaque fois.

```
age_vec <- ebola_sierra_leone$age # assigne le vecteur à l'objet "age_vec"
```

Exécutez maintenant ces fonctions sur `age_vec` et observez leurs résultats :

```
sd(age_vec, na.rm = T) # écart-type
```

```
## [1] 17.26864
```

```
max(age_vec, na.rm = T) # âge maximum
```

```
## [1] 80
```

```
min(age_vec, na.rm = T) # âge minimum
```

```
## [1] 1.8
```

```
summary(age_vec) # min, max, moyenne, quartiles et NAs
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.80   20.00   35.00   33.85   45.00   80.00       4
```

```
length(age_vec) # nombre d'éléments dans le vecteur
```

```
## [1] 200
```

```r
sum(age_vec, na.rm = T) # somme de tous les éléments du vecteur
```

```
## [1] 6633.8
```

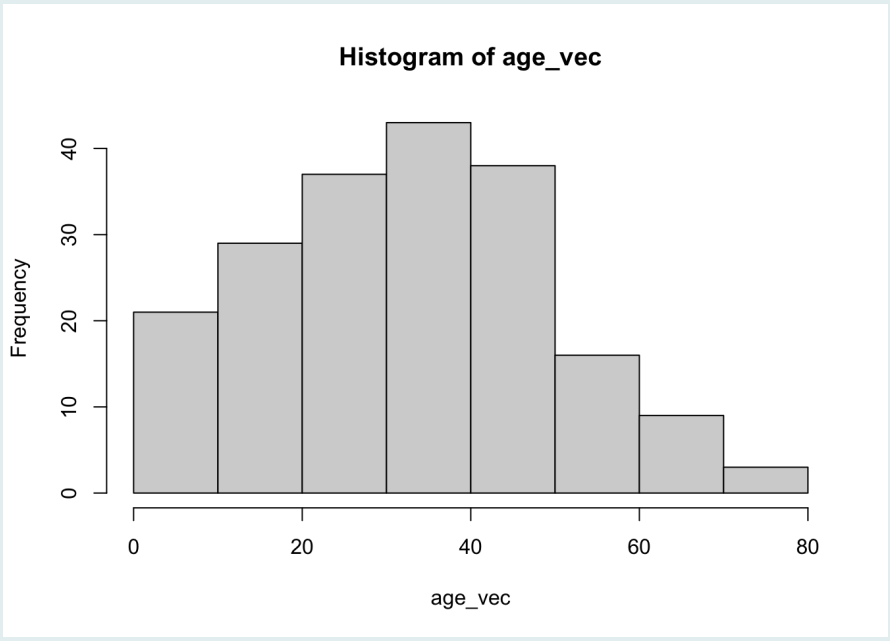
Ne vous sentez pas intimidé par la longue liste de fonctions ! Vous ne devriez pas avoir à les mémoriser ; vous devriez plutôt vous sentir libre de chercher sur Google la fonction correspondant à l'opération que vous voulez effectuer. Vous pouvez rechercher quelque chose comme "quelle est la fonction pour l'écart type dans R". L'un des premiers résultats devrait vous conduire à ce dont vous avez besoin.

Visualisation d'une variable numérique

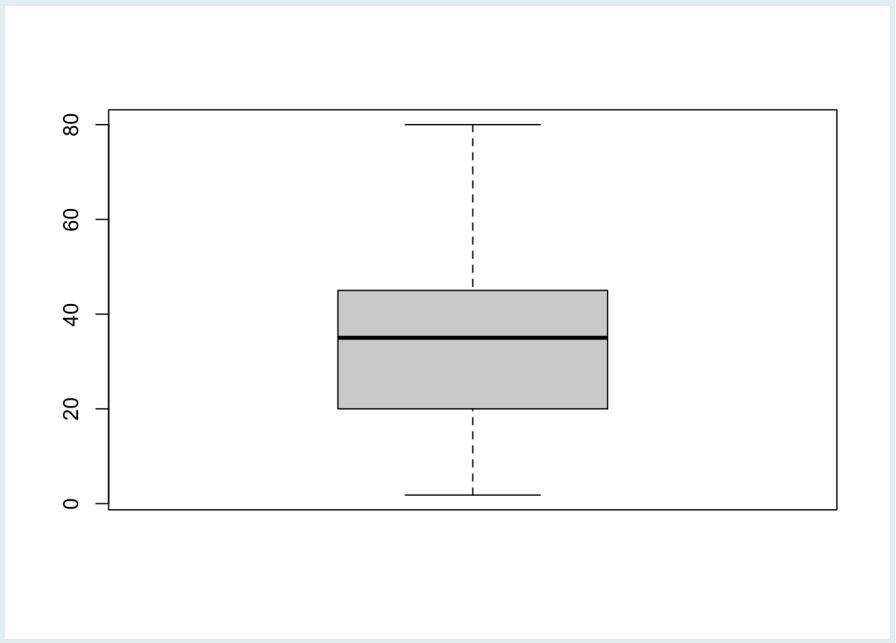
Créons maintenant un graphique pour visualiser la variable 'age'. Les deux graphiques les plus courants pour inspecter la distribution des variables numériques sont les histogrammes (comme la sortie de la fonction `inspect_num()` que vous avez vue précédemment) et boxplots.

R a des fonctions intégrées pour ceux-ci :

```r
hist(age_vec)
```




```r
boxplot(age_vec)
```



Agréable et facile !

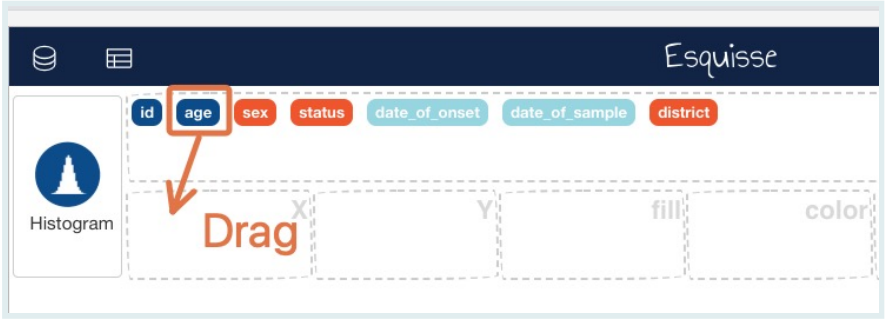
Les fonctions graphiques telles que boxplot() et hist() font partie du package graphique de base de R. Ces fonctions sont simples et rapides à utiliser, mais elles n'offrent pas beaucoup de flexibilité , et il est difficile de faire de beaux graphiques avec elles. C'est pourquoi la plupart des membres de la communauté R utilisent un package d'extension, {ggplot2}, pour la visualisation de leures données.

Dans ce cours, nous utiliserons indirectement ggplot ; en utilisant le package {esquisse}, qui fournit une interface conviviale pour créer des graphiques ggplot2.

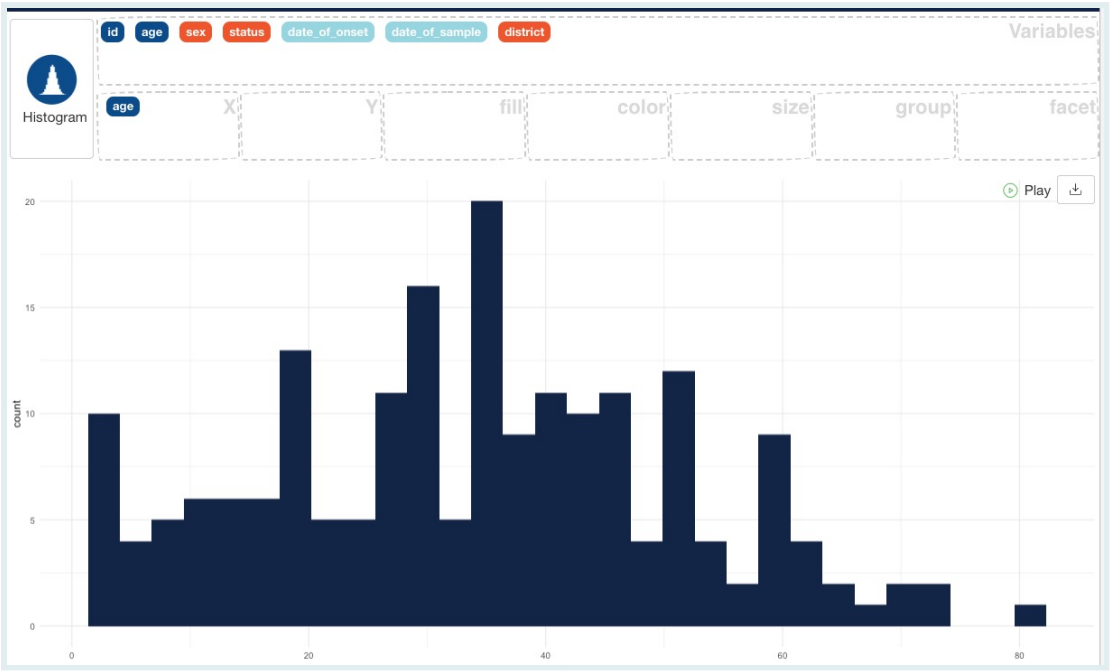
La fonction principale du package {esquisse} est `esquisser()`, et cette fonction prend un seul argument – le jeu de données que vous souhaitez visualiser. Nous pouvons donc exécuter :

```
esquisser(ebola_sierra_leone)
```

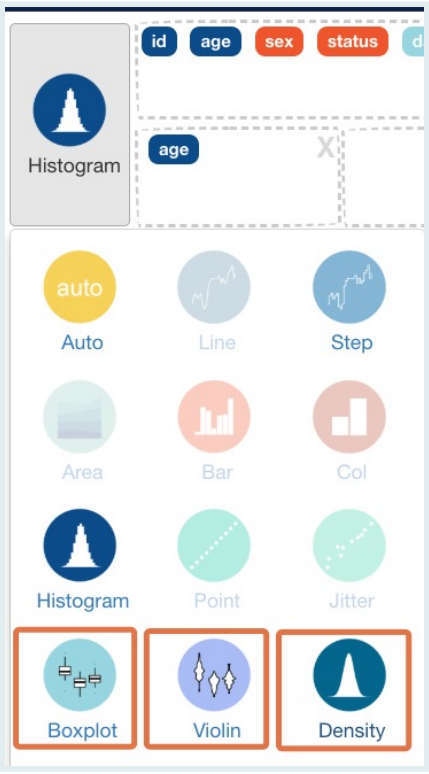
Cela devrait apporter une interface graphique que vous pouvez utiliser pour visualiser différentes variables. Pour visualiser la variable d'âge, faites simplement glisser "age" de la liste des variables vers la zone de l'axe des x :



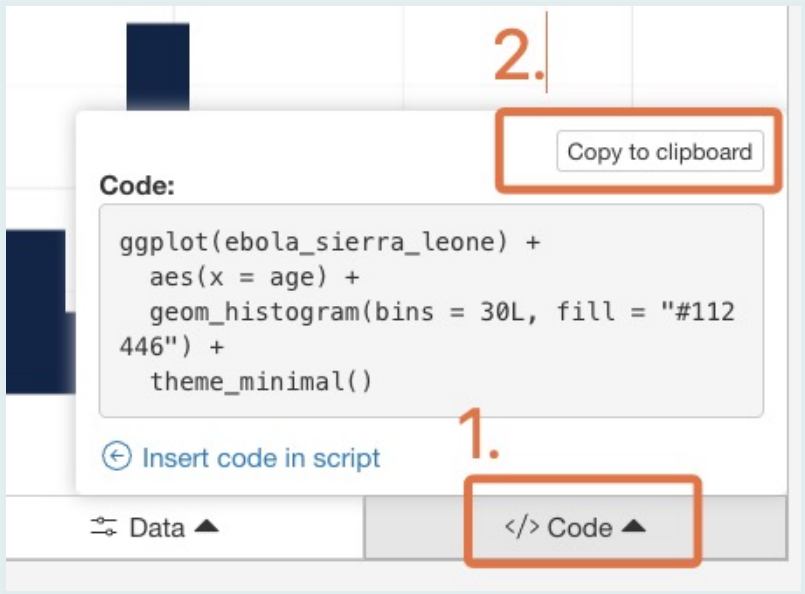
Lorsque age est dans la case de l'axe des x, vous devriez automatiquement obtenir un histogramme des âges :



Vous pouvez modifier le type de graphique en cliquant sur le bouton "Histogram" et en sélectionnant l'un des autres types de graphique valides. Essayez lediagramme en boîte, le diagramme en violon et le diagramme de densité et observez les résultats.



Lorsque vous avez terminé de créer un graphique avec {esquisse}, vous devez copier le code qui a été créé en cliquant sur le bouton “Code” en bas à droite puis “Copy to clipboard” :



Maintenant, collez ce code dans votre script et assurez-vous que vous pouvez l’exécuter à partir de là. Le code devrait ressembler à ceci :

```
ggplot(ebola_sierra_leone) +
  aes(x = age) +
  geom_histogram(bins = 30L, fill = "#112446") +
  theme_minimal()
```

En copiant le code généré dans votre script, vous vous assurez que la visualisation de données que vous avez créée est entièrement reproductible.

PRO TIP

{esquisse} ne peut créer que des graphiques assez simples, donc lorsque vous souhaitez créer des graphiques hautement personnalisés ou complexes, vous devrez apprendre à écrire le code {ggplot} manuellement. Ce sera l’objet d’un cours ultérieur.

Vous devriez également tester les autres onglets de la barre d’outils inférieure pour voir ce qu’ils font : Labels & Title, Plot options, Appearance and Data.

CHALLENGE

graphiques bivariés et multivariés faciles

Dans cette leçon, nous nous concentrons sur l’analyse univariée : explorer et visualiser une variable à la fois. Mais avec esquisse; il est * si * facile de

créer un graphique bivarié ou multivarié, vous pouvez donc déjà vous familiariser avec cela.

Essayez les graphiques suivants :

- Faites glisser `age` dans la case X et `sex` dans la case Y.
- Faites glisser `age` dans la case X, `sex` dans la case Y et `sex` dans la case de remplissage.
- Faites glisser `age` dans la case X et `district` dans la case Y.

Analyse d'une seule variable nominale

Examinons ensuite une variable nominale, les districts des cas signalés :

```
# Analyse univariée, variables nominales ----
ebola_sierra_leone$district
```

```
##   [1] "Kailahun"      "Kailahun"      "Kenema"
##   [4] "Kailahun"      "Kailahun"      "Kailahun"
##   [7] "Kailahun"      "Kailahun"      "Kenema"
##  [10] "Kailahun"      "Kailahun"      "Kailahun"
##  [13] "Kailahun"      "Kailahun"      "Kailahun"
##  [16] "Kailahun"      "Kailahun"      "Kenema"
##  [19] "Kono"          "Kailahun"      "Kailahun"
##  [22] "Kailahun"      "Kenema"        "Kailahun"
##  [25] "Kailahun"      "Kailahun"      "Kailahun"
##  [28] "Kailahun"      "Kenema"        "Kenema"
##  [31] "Kenema"        "Kailahun"      "Kailahun"
##  [34] "Bo"            "Kailahun"      "Kailahun"
##  [37] "Kailahun"      "Kenema"        "Kenema"
##  [40] "Kenema"        "Kailahun"      "Kailahun"
##  [43] "Kailahun"      "Kailahun"      "Kailahun"
##  [46] "Kailahun"      "Western Urban" "Kailahun"
##  [49] "Kailahun"      "Kailahun"      "Kailahun"
##  [52] "Kailahun"      "Kailahun"      "Kailahun"
##  [55] "Kailahun"      "Kailahun"      "Kailahun"
##  [58] "Kailahun"      "Kailahun"      "Kailahun"
##  [61] "Kailahun"      "Kenema"        "Western Urban"
##  [64] "Kambia"        "Kailahun"      "Kailahun"
##  [67] "Kailahun"      "Kailahun"      "Kailahun"
##  [70] "Kailahun"      "Kailahun"      "Kailahun"
##  [73] "Kenema"        "Kailahun"      "Kailahun"
##  [76] "Kenema"        "Kailahun"      "Kailahun"
```

```
##  [79] "Kenema"        "Kailahun"      "Kailahun"
##  [82] "Kailahun"      "Kailahun"      "Kailahun"
##  [85] "Kailahun"      "Kailahun"      "Kailahun"
##  [88] "Kailahun"      "Kailahun"      "Kenema"
##  [91] "Kailahun"      "Kailahun"      "Kailahun"
##  [94] "Kono"          "Port Loko"     "Kenema"
##  [97] "Kailahun"      "Kailahun"      "Kailahun"
## [100] "Kailahun"      "Kenema"        "Kailahun"
## [103] "Kailahun"      "Kenema"        "Kailahun"
## [106] "Kailahun"      "Kailahun"      "Kailahun"
## [109] "Kailahun"      "Kailahun"      "Kenema"
## [112] "Western Urban" "Kailahun"      "Kailahun"
## [115] "Kailahun"      "Kailahun"      "Kailahun"
## [118] "Kailahun"      "Kailahun"      "Kailahun"
## [121] "Kailahun"      "Kailahun"      "Kenema"
## [124] "Kailahun"      "Kailahun"      "Kenema"
## [127] "Kailahun"      "Port Loko"     "Kailahun"
## [130] "Kailahun"      "Kailahun"      "Kailahun"
## [133] "Kailahun"      "Kailahun"      "Kailahun"
## [136] "Kailahun"      "Kailahun"      "Kailahun"
## [139] "Kailahun"      "Kailahun"      "Kailahun"
## [142] "Kailahun"      "Kailahun"      "Kenema"
## [145] "Kenema"        "Kailahun"      "Kenema"
## [148] "Kailahun"      "Kailahun"      "Kailahun"
## [151] "Kailahun"      "Kailahun"      "Kenema"
## [154] "Kailahun"      "Kailahun"      "Kenema"
## [157] "Kailahun"      "Kenema"        "Kailahun"
## [160] "Kailahun"      "Kenema"        "Kailahun"
## [163] "Kailahun"      "Kailahun"      "Kailahun"
## [166] "Bo"            "Kailahun"      "Kailahun"
## [169] "Kailahun"      "Kailahun"      "Kailahun"
## [172] "Kailahun"      "Kenema"        "Kailahun"
## [175] "Kailahun"      "Kenema"        "Kailahun"
## [178] "Kailahun"      "Kailahun"      "Kailahun"
## [181] "Kailahun"      "Kailahun"      "Kailahun"
## [184] "Western Urban" "Kailahun"      "Kailahun"
## [187] "Kenema"        "Kailahun"      "Kailahun"
## [190] "Kailahun"      "Kailahun"      "Kailahun"
## [193] "Kailahun"      "Kenema"        "Kenema"
## [196] "Kailahun"      "Kailahun"      "Kailahun"
## [199] "Kailahun"      "Kenema"
```

Désolé d'avoir imprimé ce très long vecteur !

Tableaux de fréquence

Vous pouvez utiliser la fonction `table()` pour créer un tableau de fréquence d'une variable nominale :

```
table(ebola_sierra_leone$district)
```

```
##
##           Bo      Kailahun        Kambia        Kenema
```

```
##             2           155             1            34
##          Kono     Port Loko Western Urban
##             2             2             4
```

Vous pouvez voir que la plupart des cas se trouvent à Kailahun et à Kenema.

`table()` est une fonction "de base" utile. Mais il existe une meilleure fonction pour créer des tableaux de fréquence, appelée `tabyl()`, à partir du package {janitor}.

Pour l'utiliser, vous devez fournir le nom de votre base de données en premier argument, puis le nom de la variable à tabuler :

```
tabyl(ebola_sierra_leone, district)
```

```
##       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
```

Comme vous pouvez le voir, `tabyl()` vous donne à la fois les nombres et les pourcentages de chaque valeur. Il possède également d'autres fonctionnalités intéressantes que vous verrez plus tard.

Vous pouvez aussi facilement faire des tableaux croisés avec `tabyl()`. Il suffit d'ajouter des variables supplémentaires séparées par une virgule. Par exemple, pour créer un tableau croisé par quartier et par sexe, exécutez :

```
tabyl(ebola_sierra_leone, district, sex)
```

```
##       district  F  M
##             Bo  0  2
##       Kailahun 91 64
##         Kambia  0  1
##         Kenema 20 14
##           Kono  0  2
##      Port Loko  1  1
##  Western Urban  2  2
```

Le résultat nous montre qu'il y avait 0 femmes dans le district de Bo, 2 hommes dans le district de Bo, 91 femmes dans le district de Kailahun, etc.

Visualisation d'une variable nominale

Maintenant, essayons de visualiser la variable `district`. Comme précédemment, la meilleure façon de le faire est d'utiliser la fonction `esquisser()` de {esquisse}. Exécutez à nouveau ce code :

```
esquisser(ebola_sierra_leone)
```

Faites ensuite glisser la variable `district` vers la zone de l'axe X :

Vous devriez obtenir un graphique à barres indiquant le nombre d'individus dans les districts. Copiez le code généré et collez-le dans votre script.

Réponses aux questions sur l'épidémie

Avec les fonctions que vous venez d'apprendre, vous disposez des outils nécessaires pour répondre aux questions sur l'épidémie d'Ebola qui ont été énumérées en haut. Essayez voir! Essayez de répondre à ces questions par vous-même, puis regardez les solutions ci-dessous.

- **Quand le premier cas a-t-il été signalé ? (Indice : regardez la date de l'échantillon)**
- **À fin du mois de juin 2014, quelle tranche d'âge de 10 ans avait eu le plus de cas ?**
- **Quel était l'âge médian des personnes touchées ?**
- **Y avait-il eu plus de cas chez les hommes ou chez les femmes ?**
- **Quel est le district qui a enregistré le plus grand nombre de cas signalés ?**
- **À la fin du mois de juin 2014, l'épidémie était-elle en croissance ou en regression ?**

Solutions

- **Quand le premier cas a-t-il été signalé ?**

```
min(ebola_sierra_leone$date_of_sample)
```

```
## [1] "2014-05-23"
```

Nous n'avons pas la date du rapport, mais la première "date_of_sample" (date à laquelle l'échantillon de test Ebola a été prélevé sur le patient) est le 23 mai. Nous pouvons l'utiliser comme approximation de la date du premier rapport.

- **Quel était l'âge médian des cas ?**

```
median(ebola_sierra_leone$age, na.rm = T)
```

```
## [1] 35
```

L'âge médian des cas était de 35 ans.

- **Y a-t-il plus de cas chez les hommes ou chez les femmes ?**

```
tabyl(ebola_sierra_leone$sex)
```

```
##  ebola_sierra_leone$sex   n percent
##                       F 114    0.57
##                       M  86    0.43
```

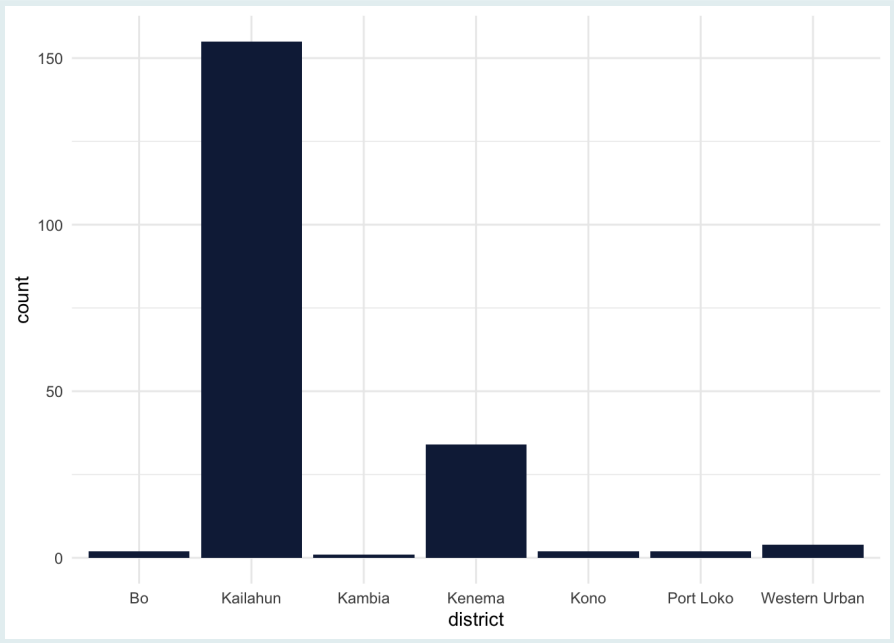
Comme on le voit dans le tableau, il y avait plus de cas chez les femmes. Plus précisément, 57 % des cas concernent des femmes.

- **Quel district a enregistré le plus de cas signalés ?**

```
tabyl(ebola_sierra_leone$district)
```

```
##  ebola_sierra_leone$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
```

```
# On peut aussi tracer le graphique suivant (généré avec esquisse)
ggplot(ebola_sierra_leone) +
  aes(x = district) +
  geom_bar(fill = "#112446") +
  theme_minimal()
```

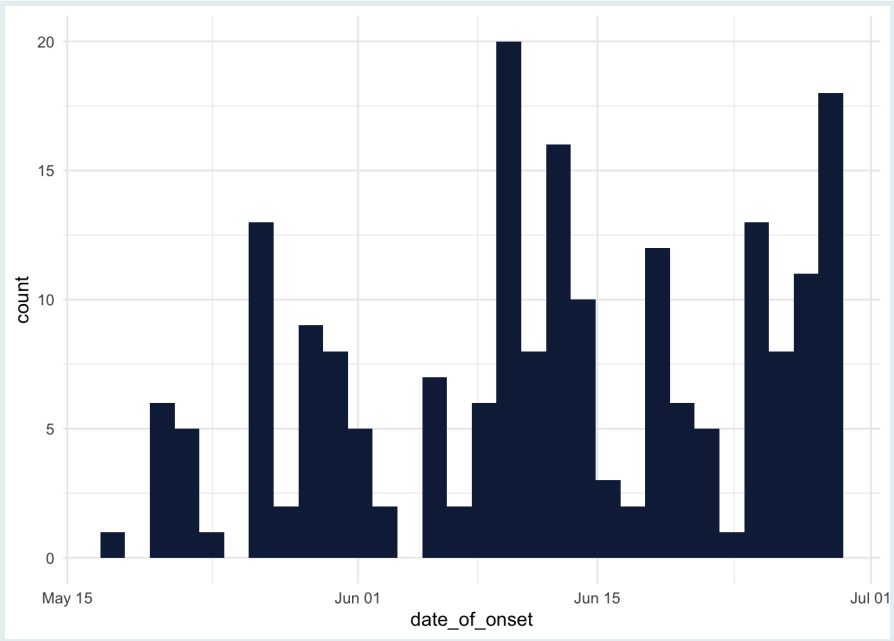


Comme on l'a vu, le district de Kailahun comptait la majorité des cas.

- **À la fin du mois de juin 2014, l'épidémie était-elle en croissance ou en recul ?**

Pour cela, nous pouvons utiliser esquisse pour générer un diagramme à barres qui montre le nombre de cas chaque jour. Faites simplement glisser la variable `date_of_onset` sur l'axe des x. Le code de sortie d'esquisse doit ressembler à ce qui suit :

```
ggplot(ebola_sierra_leone) +
  aes(x = date_of_onset) +
  geom_histogram(bins = 30L, fill = "#112446") +
  theme_minimal()
```



Super! Mais on peut se demander si l'épidémie augmentait ou reculait fin juin 2014 ; une tendance précise n'est pas vraiment claire!

Vous n'en avez pas eu assez ?

Si vous souhaitez mettre en pratique certaines des méthodes et fonctions que vous avez apprises sur un jeu de données similaire, essayez de télécharger les données stockées sur cette page : https://bit.ly/view-yaounde-covid-data

Ce jeu de données se présente sous la forme d'une feuille de calcul Excel, donc lorsque vous importez le jeu de données avec RStudio, vous devez utiliser l'option "From Excel" (File > Import Dataset > From Excel).

Ce jeu de données contient les résultats d'une enquête sérologique COVID-19 menée à Yaoundé, au Cameroun, à la fin de l'année 2020. L'enquête a permis d'estimer le nombre de personnes infectées par le COVID-19 dans la région, en testant les anticorps IgG et IgM. le jeu de données complet peut être obtenu ici : go.nature.com/3R866wx

Conclusion

Toutes nos félicitations! Vous avez maintenant fait vos premiers pas dans l'analyse des données avec R : vous avez importé un jeu de données, exploré sa structure, effectué une analyse et une visualisation univariées de base sur ses variables numériques et nominales, et vous avez pu répondre à des questions importantes sur l'épidémie en vous basant sur cette base base.

Bien sûr, ce n'était qu'un *aperçu* du processus d'analyse des données – beaucoup a été laissé de côté. Avec un peu de chance, cependant, cet aperçu vous a un peu enthousiasmé par ce que vous pouvez faire avec R. Et j'espère que vous pourrez déjà commencer à appliquer certaines d'entre elles à vos propres jeux de données. Le voyage ne fait que commencer ! À bientôt.

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 :

- Barnier, Julien. “Introduction à R Et Au Tidyverse.” Partie 13 Diffuser et publier avec rmarkdown, 24 mai 2022. https://juba.github.io/tidyverse/13-rmarkdown.html.
- Yihui Xie, J. J. Allaire et Garrett Grolemund. “R Markdown : le guide définitif.” Accueil, 11 avril 2022. https://bookdown.org/yihui/rmarkdown/.