
Nettoyage de données: analyse exploratoire

Introduction	
Objectifs d'apprentissage	
Packages	
Jeu de données	
Visualisation des données manquantes avec <code>visdat::vis_dat()</code>	
Génération de statistiques sommaires avec <code>skimr::skim()</code>	
Visualisation des statistiques sommaires avec le package <code>inspectdf</code>	
Explorer les variables catégorielles avec <code>gtsummary::tbl_summary()</code>	
Création de rapports de données avec <code>DataExplorer::create_report()</code>	
En Résumé	
Answer Key	

Introduction

Le nettoyage des données est le processus qui consiste à transformer des données brutes et “désordonnées” en des données fiables pouvant être analysées correctement. Cela implique d'identifier les points de données **inexacts**, **incomplets** ou **improbables** et de résoudre les incohérences ou les erreurs dans les données, ainsi que de renommer les noms de variables pour les rendre plus clairs et simples à manipuler.

Les tâches de nettoyage de données peuvent souvent être fastidieuses. Une blague courante chez les analystes de données dit : “80% de la science des données consiste à nettoyer les données et les 20% restants à se plaindre du nettoyage des données”. Mais le nettoyage des données est une étape essentielle du processus d'analyse des données. Un peu de nettoyage avant de commencer votre analyse contribuera à améliorer la qualité de vos analyses et la facilité avec laquelle ces analyses peuvent être effectuées. Et une gamme de packages et de fonctions dans R peuvent énormément simplifier le processus de nettoyage des données.

Dans cette leçon, nous allons commencer à examiner un processus de nettoyage de données typique dans R. Les étapes de nettoyage couvertes ici ne correspondent probablement pas exactement à ce dont vous aurez besoin pour vos propres données, mais elles constituent certainement un bon point de départ.

Commençons !

Objectifs d'apprentissage

- Vous pouvez énumérer les opérations typiques impliquées dans le processus de nettoyage des données

- Vous pouvez diagnostiquer des problèmes dans vos jeux de données à l'aide des fonctions suivantes :

- `visdat::vis_dat()`
- `skimr::skim()`
- `inspectdf::inspect_cat()`
- `inspectdf::inspect_num()`
- `gtsummary::tbl_summary()`
- `DataExplorer::create_report()`

Packages

Les packages chargés ci-dessous seront nécessaires pour cette leçon :

```
if(!require("pacman")) install.packages("pacman")
pacman::p_load(visdat,
               skimr,
               inspectdf,
               gtsummary,
               DataExplorer,
               tidyverse)
```

Jeu de données

Le jeu de données principal que nous utiliserons dans cette leçon provient d'une étude menée dans trois centres de santé de Zambezia, au Mozambique. L'étude a examiné les facteurs individuels associés au temps jusqu'à la non-adhérance aux services de soins et de traitement du VIH. Pour les besoins de cette leçon, nous ne regarderons qu'un sous-ensemble modifié de l'ensemble de données complet.

L'ensemble de données complet peut être trouvé [ici](#), et l'article peut être consulté [ici](#).

Jetons un coup d'œil à ce jeu de données :

```
non_adherence <- read_csv(here("data/non_adherence_VF.csv"))
```

```
non_adherence
```

```
## # A tibble: 5 × 15
##   id_patient District `Unite de sante` Sexe Age_35
##   <dbl>      <dbl>          <dbl> <chr> <chr>
## 1     10037         1             1 Homme plus de 35 ans
## 2     10537         1             1 F    plus de 35 ans
## 3      5489         2             3 F    Moins de 35 ans
## 4      5523         2             3 Homme Moins de 35 ans
## 5      4942         2             3 F    plus de 35 ans
## # i 10 more variables: `Age a l'initiation du ARV` <dbl>,
## #   Education <chr>, Occupation <chr>, ...
```

La première étape du nettoyage des données consistera à explorer cet ensemble de données afin d'identifier les problèmes potentiels qui nécessitent un nettoyage. Cette étape préliminaire est parfois appelée "analyse exploratoire des données" ou AED.

Regardons quelques fonctions simples dans R qui vous aideront à identifier les erreurs et incohérences de données possibles.

Visualisation des données manquantes avec `visdat::vis_dat()`

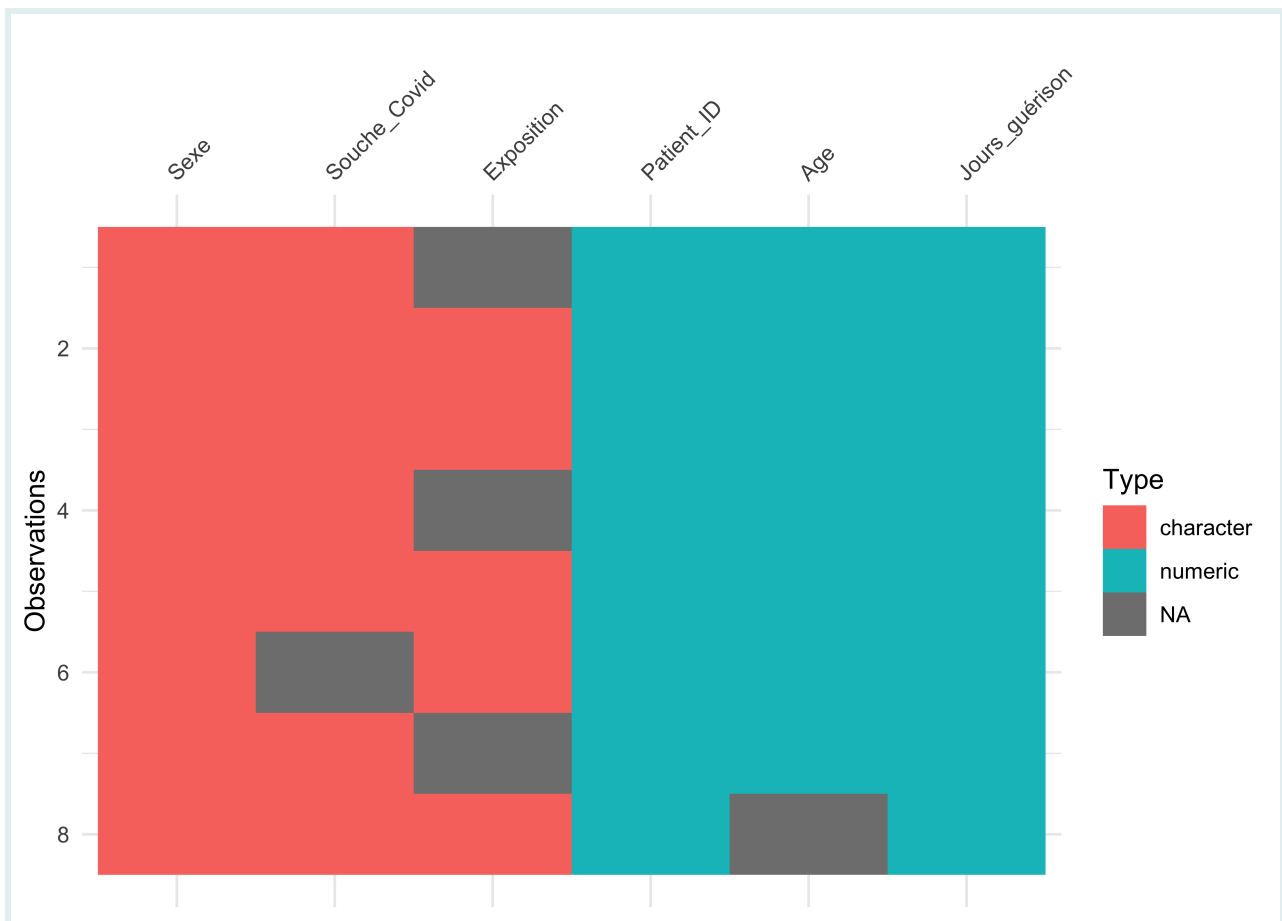
La fonction `vis_dat()` du package `visdat` est un excellent moyen de visualiser rapidement les types de données et les valeurs manquantes dans un jeu de données. Elle crée un graphique qui montre une vue "zoomée" de votre dataframe : chaque ligne du dataframe est représentée par une seule ligne sur le graphique.

Essayons d'abord avec un petit jeu de données fictif pour comprendre son fonctionnement. Copiez le code suivant pour créer un dataframe de 8 patients et leurs informations de diagnostic et de guérison du COVID-19. Comme vous pouvez le voir ci-dessous, certaines informations sont manquantes pour certains patients, représentées par NA.

```
covid_pat <- tribble(
  ~Patient_ID, ~Age, ~Sexe, ~Souche_Covid, ~Exposition, ~Jours_guérison,
  1, 25, "Homme", "Alpha", NA, 10,
  2, 32, "Femme", "Delta", "Hôpital", 15,
  3, 45, "Homme", "Beta", "Voyage", 7,
  4, 19, "Femme", "Omicron", NA, 21,
  5, 38, "Homme", "Alpha", "Inconnu", 14,
  6, 55, "Femme", NA, "Communauté", 19,
  7, 28, "Femme", "Omicron", NA, 8,
  8, NA, "Femme", "Omicron", "Voyage", 26
)
covid_pat
```

Maintenant, utilisons la fonction `vis_dat()` sur notre dataframe pour obtenir une représentation visuelle des types de données et des valeurs manquantes.

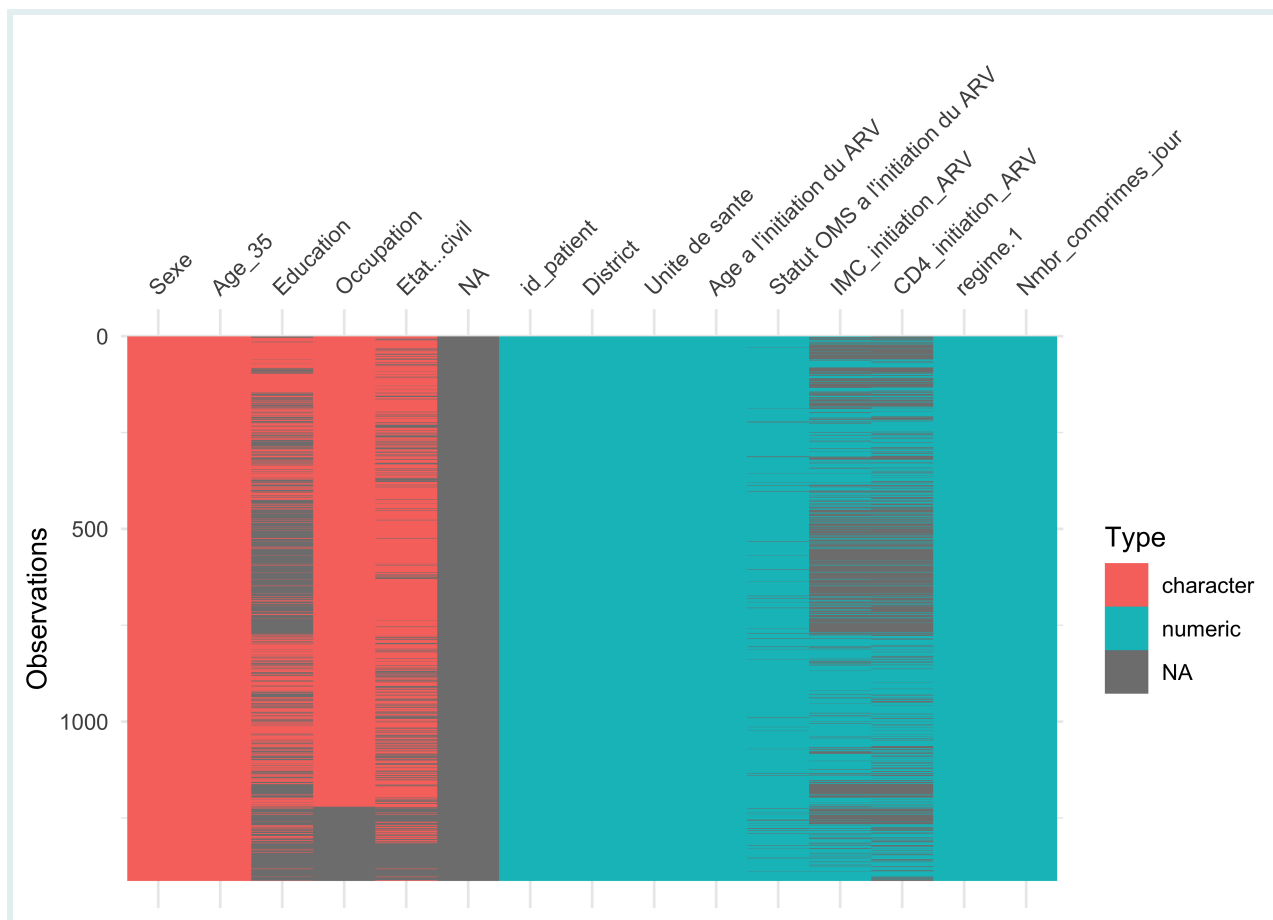
```
vis_dat(covid_pat)
```



C'est parfait ! Chaque ligne de notre dataframe est représentée par une seule ligne dans le graphique et différentes couleurs sont utilisées pour illustrer les variables caractères (rose) et numériques (bleu), ainsi que les valeurs manquantes (gris). À partir de ce graphique, nous pouvons voir que plusieurs patients de notre jeu de données ont des données manquantes pour la variable `Exposition`.

Regardons maintenant notre jeu de données du monde réel, qui est beaucoup plus grand et désordonné. Les grands jeux de données réels peuvent présenter des structures complexes qui sont difficiles à repérer sans visualisation, donc des fonctions comme `vis_dat()` peuvent être particulièrement utiles. Essayons-le maintenant !

```
vis_dat(non_adherence)
```



Génial ! À partir de cette vue, nous pouvons déjà voir certains problèmes :

- Il semble y avoir une colonne complètement vide (la colonne NA qui est entièrement grise)
- Plusieurs variables ont beaucoup de valeurs manquantes (comme Education, IMC_initiation_ARV et CD4_initiation_ARV)
- Les noms de certaines variables sont peu clairs/non nettoyés (par exemple, Age a l'initiation du ARV et Statut OMS a l'initiation du ARV ont des espaces dans leurs noms et Etat...civil et regimen.1 ont des caractères spéciaux .)

Dans la prochaine leçon, nous essayerons de remédier à ces problèmes pendant le processus de nettoyage des données. Mais pour l'instant, l'objectif est que nous comprenions les fonctions utilisées pour les identifier. Maintenant que nous savons visualiser les données manquantes avec `vis_dat()`, jetons un coup d'œil à un autre package et fonction qui peut nous aider à générer des statistiques sommaires de nos variables !

Q : Repérer les problèmes de données avec `vis_dat()`

L'ensemble de données suivant a été adapté d'une étude qui a examiné les opportunités manquées de dépistage du VIH chez les patients se présentant pour la première fois pour

des soins contre le VIH dans un hôpital universitaire suisse. L'ensemble de données complet peut être trouvé [ici](#).

```
opp_manquees <- read_csv(here("data/opportunites_manquees.csv"))
```

```
## Rows: 201 Columns: 16
## — Column specification
## Delimiter: ","
## chr (11): sexe, age, origine, acquis, chronic, aigu, present_tardive,
##   raison...
## dbl (4): cd4, categorie_cd4, nmbr_consult, opp_manquees
## lgl (1): NaN.
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
##   message.
```

Utilisez la fonction `vis_dat()` pour obtenir une représentation visuelle des données. Quels problèmes potentiels pouvez-vous repérer ?

Génération de statistiques sommaires avec `skimr::skim()`

La fonction `skim()` du package `skimr` fournit un résumé de chaque colonne (par classe/type) dans la console. Essayons-la d'abord sur nos données fictives `covid_pat` pour comprendre son fonctionnement. Tout d'abord, pour rappel, voici notre dataframe `covid_pat` :

```
covid_pat
```

Parfait, essayons maintenant la fonction `skim()` !

```
skimr::skim(covid_pat)
```

Table 1: Data summary

Name	covid_pat
Number of rows	8
Number of columns	6
Column type frequency:	
character	3
numeric	3
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Sexe	0	1.00	5	5	0	2	0
Souche_Covid	1	0.88	4	7	0	4	0
Exposition	3	0.62	6	10	0	4	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Patient_ID	0	1.00	4.50	2.45	1	2.75	4.5	6.25	8	
Age	1	0.88	34.57	12.39	19	26.50	32.0	41.50	55	
Jours_guérison	0	1.00	15.00	6.68	7	9.50	14.5	19.50	26	

Incroyable ! Comme nous pouvons le voir, cette fonction fournit :

- Un aperçu des lignes et des colonnes du dataframe
- Le type de données pour chaque variable
- `n_missing`, le nombre de valeurs manquantes pour chaque variable
- `complete_rate`, le taux d'exhaustivité pour chaque variable
- Un ensemble de statistiques sommaires : la moyenne, l'écart-type, et les quartiles pour les variables numériques ; et la fréquence et les proportions pour les variables catégorielles
- Des histogrammes spark pour les variables numériques

Maintenant, nous pouvons voir pourquoi cela est si utile avec de grands dataframes ! Revenons à notre jeu de données `non_adherence` et exécutons la fonction `skim()` dessus.

```
non_adherence
```

```
skimr::skim(non_adherence)
```

Table 2: Data summary

Name	non_adherence
Number of rows	1413
Number of columns	15
Column type frequency:	
character	5
logical	1
numeric	9

Group variables	None
-----------------	------

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Sexe	0	1.00	1	5	0	2	0
Age_35	0	1.00	14	15	0	2	0
Education	776	0.45	4	10	0	5	0
Occupation	193	0.86	4	25	0	49	0
Etat...civil	412	0.71	7	12	0	4	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
NA	1413	0	NaN	:

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p
id_patient	0	1.00	7364.75	3312.95	147.0	4705.00	7355.0	10098.0	243
District	0	1.00	1.35	0.48	1.0	1.00	1.0	2.0	
Unite de sante	0	1.00	1.87	0.90	1.0	1.00	2.0	3.0	
Age a l'initiation du ARV	0	1.00	32.07	9.51	15.1	25.20	30.4	36.8	7
Statut OMS a l'initiation du ARV	45	0.97	1.80	0.98	1.0	1.00	1.0	3.0	
IMC_initiation_ARV	588	0.58	20.37	3.54	9.0	18.12	20.1	22.2	5
CD4_initiation_ARV	674	0.52	398.19	243.64	3.0	233.00	343.0	538.0	140
regime.1	0	1.00	4.64	1.54	1.0	3.00	6.0	6.0	
Nmbr_comprimes_jour	0	1.00	1.51	0.59	1.0	1.00	1.0	2.0	

À partir de cette sortie de données, nous pouvons identifier certains problèmes potentiels :

- Nous pouvons confirmer que la colonne NA est bien complètement vide : elle a un `complete_rate` de 0
- La distribution de Age a l'initiation du ARV est asymétrique

Q : Générer des statistiques sommaires avec `skim()`

Utilisez `skim()` pour obtenir un aperçu détaillé de l'ensemble de données `opp_manquees`.

Super ! Maintenant nous savons comment générer un bref rapport de statistiques sommaires pour nos variables. Dans la prochaine section, nous découvrirons quelques fonctions utiles du package `inspectdf` qui nous permettent de visualiser différentes statistiques sommaires.

Visualisation des statistiques sommaires avec le package `inspectdf`

Bien que la fonction `skimr::skim()` vous donne des résumés de variables dans la console, parfois il est préférable d'avoir un résumé de variables sous une forme graphique plus riche. Pour cela, les fonctions `inspectdf::inspect_cat()` et `inspectdf::inspect_num()` peuvent être utilisées.

Si vous exécutez `inspect_cat()` sur un jeu de données, vous obtenez un résumé des variables catégorielles dans un tableau (les informations importantes sont cachées dans

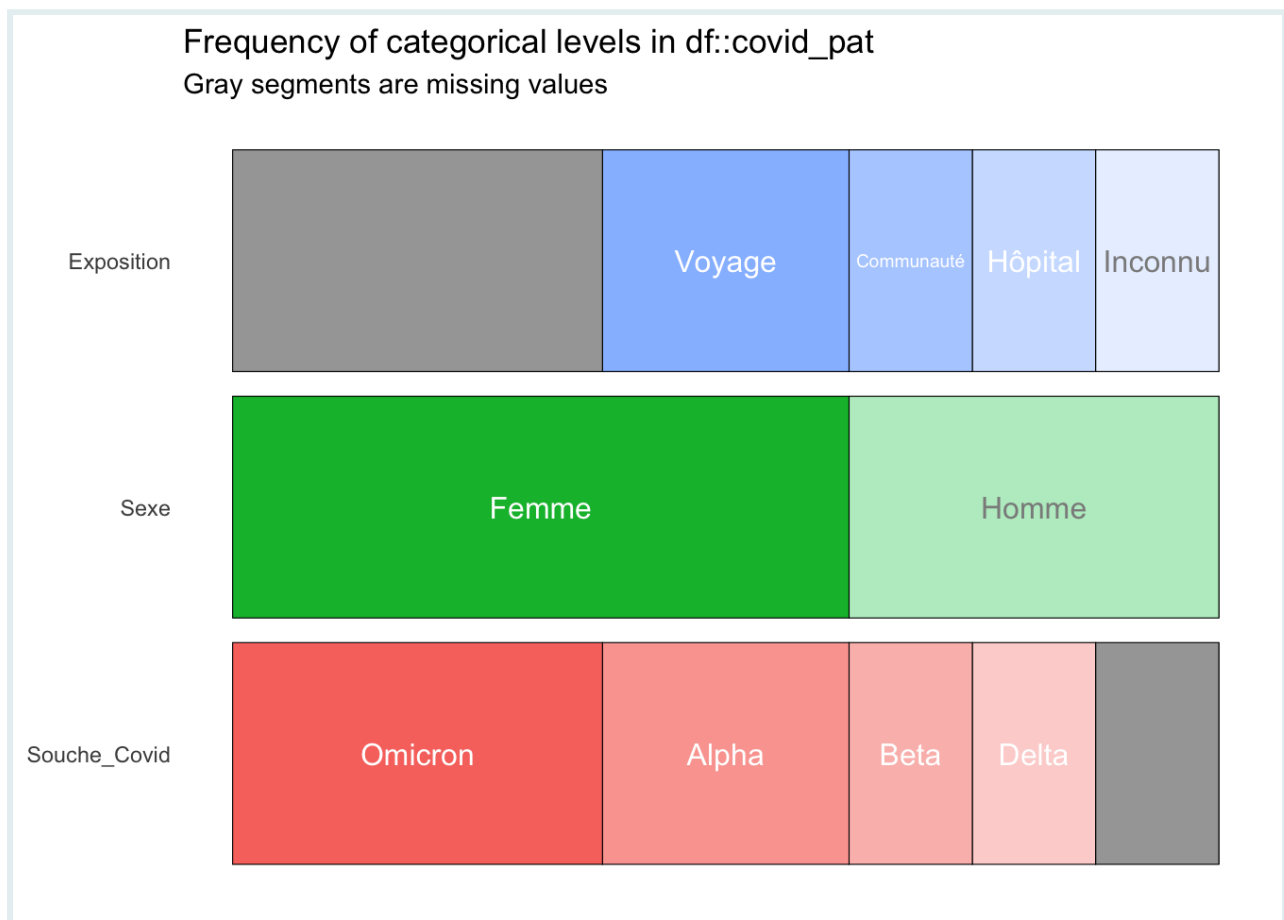
la colonne levels). Essayons-le d'abord sur le jeu de données covid_pat. Pour rappel, voici notre jeu de données :

```
covid_pat
```

```
inspect_cat(covid_pat)
```

La magie se produit lorsque vous utilisiez la fonction `show_plot()` sur le résultat de `inspect_cat()` :

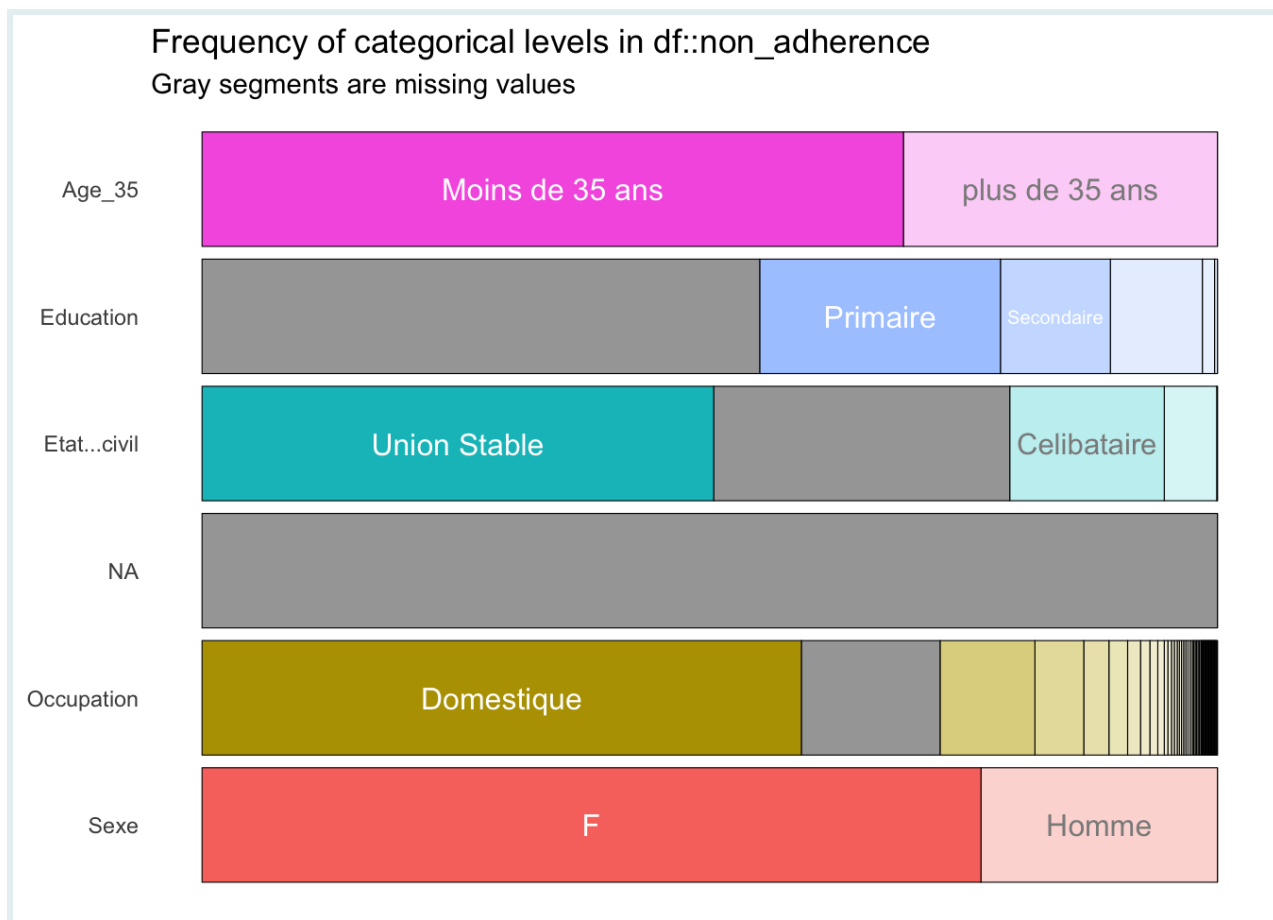
```
inspect_cat(covid_pat) %>%  
  show_plot()
```



C'est parfait ! Vous obtenez une belle figure récapitulative montrant la distribution des variables catégorielles ! Les niveaux de variable sont également étiquetés, s'il y a suffisamment d'espace pour afficher une étiquette.

Maintenant, essayons-le sur notre jeu de données non_adherence :

```
inspect_cat(non_adherence) %>%  
  show_plot()
```



À partir de là, vous pouvez observer certains problèmes avec quelques variables catégorielles :

- Pour le variable Age_35, on a deux niveaux. Le niveau Moins de 35 ans commence par une majuscule, tandis que plus de 35 ans commence par une minuscule. Ça pourrait être un bon idée de standardiser cela.
- La variable sexe a les niveaux F et Homme. Cela pourrait également valoir la peine d'être standardisé.
- Comme nous l'avons vu précédemment, NA est complètement vide.

Q : Repérer les problèmes de données avec `inspect_cat()`

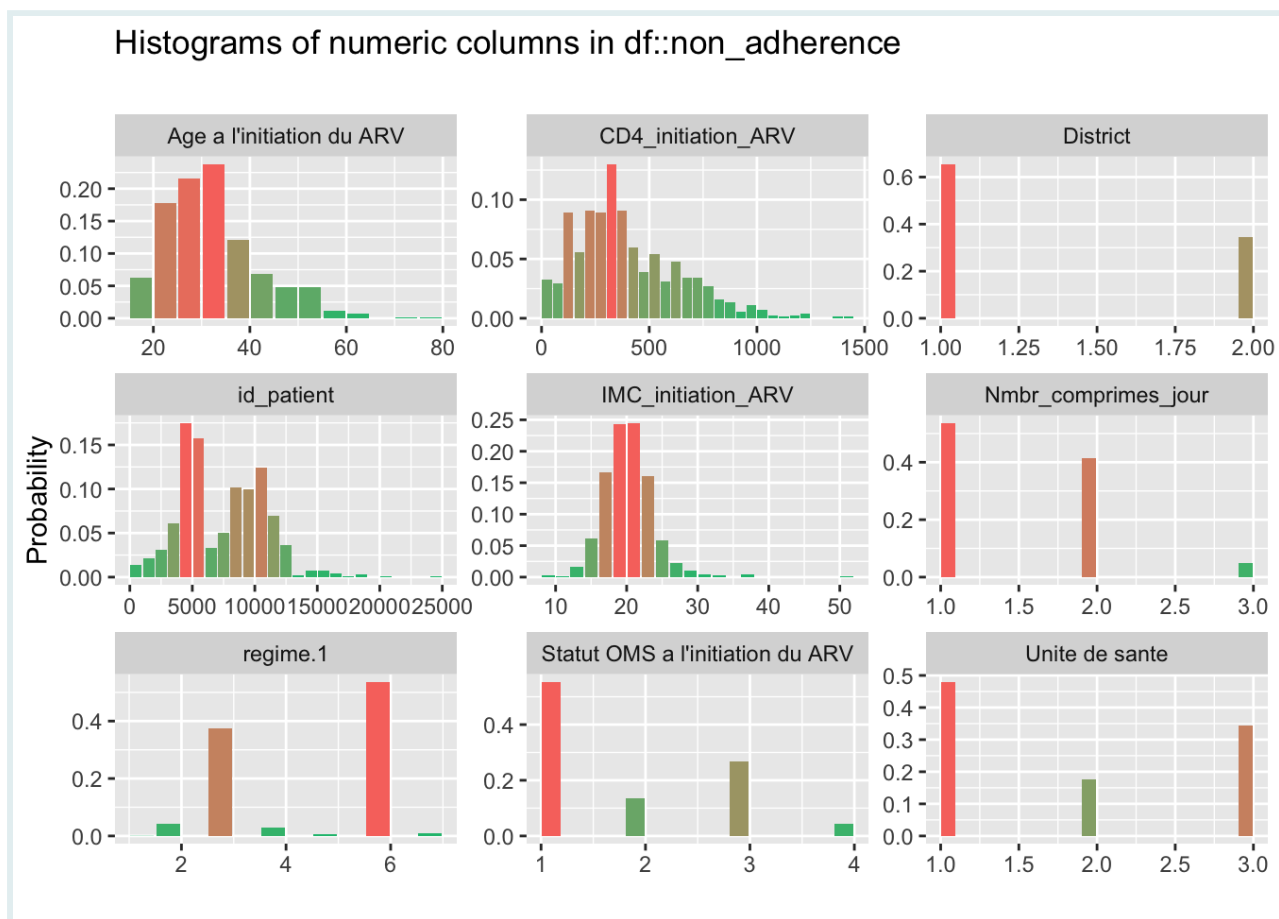
Complétez le code suivant pour obtenir un résumé visuel des variables catégorielles dans le jeu de données opp_manquees.

```
inspect___() %>%
```

Combien de problèmes de données potentiels pouvez-vous repérer ?

De même, vous pouvez obtenir un graphique de résumé pour les variables numériques du jeu de données avec `inspect_num()`. Essayons cela sur notre jeu de données `non_adherence` :

```
inspect_num(non_adherence) %>%  
  show_plot()
```



À partir de ce graphique généré, vous remarqueriez que de nombreuses variables qui devraient être des variables factor sont codées comme numériques. En fait, les seules vraies variables numériques sont `Age a l'initiation du ARV`, `IMC_initiation_ARV`, `CD4_initiation_ARV` et `Nmbr_comprimes_jour`. Nous corrigerons ces problèmes dans la prochaine leçon lorsque nous passerons au nettoyage des données. En attendant, regardons de plus près une autre fonction qui s'avère particulièrement utile pour les variables catégorielles !

Q : Types de variables avec `inspect_num()`

Utilisez `inspect_num` pour créer des histogrammes de vos variables numériques dans le jeu de données `opp_manquees`. Les types de variables numériques sont-ils corrects ?

Explorer les variables catégorielles avec `gtsummary::tbl_summary()`

Bien que la fonction `inspect_cat()` soit utile pour un aperçu graphique des variables catégorielles, elle ne fournit pas d'informations sur les fréquences et les pourcentages pour les différents niveaux. Pour cela, `tbl_summary()` du package `gtsummary` est particulièrement utile ! La sortie de données étant particulièrement longue, nous allons regarder la forme tibble et montrer une photo de la partie importante pour notre jeu de données. Vous pouvez explorer les données complètes en codant chez vous.

Essayons-le sur notre jeu de données `non_adherence` :

```
gtsummary::tbl_summary(non_adherence) %>%  
  as_tibble()
```

```
## # A tibble: 103 × 2  
##   `**Characteristic**` `**N = 1,413**`  
##   <chr>                <chr>  
## 1 id_patient          7,355 (4,705, 10,098)  
## 2 District            <NA>  
## 3 1                    925 (65%)  
## 4 2                    488 (35%)  
## 5 Unite de sante      <NA>  
## 6 1                    678 (48%)  
## 7 2                    247 (17%)  
## 8 3                    488 (35%)  
## 9 Sexe                <NA>  
## 10 F                  1,084 (77%)  
## # i 93 more rows
```

Super ! Comme nous pouvons le voir, cela nous fournit un résumé des fréquences et des pourcentages pour les variables catégorielles et la médiane et l'IQR pour les variables numériques.

Ci-dessous vous pouvez voir une photo d'une partie de la sortie où nous pouvons remarquer des problèmes supplémentaires dans nos données qui n'étaient pas clairs avec la fonction `inspect_cat()`. Certaines valeurs de notre variable `Occupation` sont en majuscules, tandis que d'autres sont entièrement en minuscules.

Police	3 (0.2%)
Pompier	1 (<0.1%)
professeur	11 (0.9%)
Professeur	35 (2.9%)
Proprietaire de magasin	1 (<0.1%)
Receptionniste	1 (<0.1%)
Retraite	2 (0.2%)

Cela signifie que R ne les reconnaît pas comme étant la même valeur, ce qui poserait problème lors de l'analyse. Nous corrigerons ces erreurs dans la prochaine leçon, pour l'instant passons à notre dernière fonction pour l'analyse exploratoire des données !

PRACTICE



(in RMD)

Q : Repérer les problèmes de données avec `tbl_summary()`

Utilisez `tbl_summary()` pour produire un résumé de votre jeu de données `opp_manquees`. Pouvez-vous identifier des problèmes de données supplémentaires ?

Création de rapports de données avec `DataExplorer::create_report()`

Enfin, la fonction `create_report()` du package `DataExplorer` crée un profil complet d'un dataframe : un fichier HTML avec des statistiques de base et des visualisations de distribution.

Utilisons cette fonction sur notre jeu de données `non_adherence`. Notez que cela peut prendre un certain temps. Si cela prend trop de temps, vous pouvez l'exécuter sur un sous-ensemble du jeu de données plutôt que sur l'ensemble du jeu de données.


```
create_report(non_adherence)
```

Comme vous pouvez le voir, le rapport est assez complet. Nous ne passerons pas en revue toutes les sorties de ce rapport de données car de nombreuses graphiques sont les mêmes que celles que nous avons vues avec les fonctions précédentes ! Cependant, certains nouveautés incluent :

- Un graphique QQ pour évaluer la normalité des variables numériques
- Une analyse de corrélation (lorsqu'il y a suffisamment de lignes complètes)
- Une analyse en composantes principales (lorsqu'il y a suffisamment de lignes complètes)

N'hésitez pas à explorer la [documentation du package](#) par vous-même.

Q : Rapport de données avec `create_report()`

Créez un rapport de données pour vos données `opp_manquees` en utilisant la fonction `create_report()` !

En Résumé

En nous familiarisant avec les données, nous avons pu identifier certains problèmes potentiels qu'il faudra résoudre avant d'utiliser les données dans une analyse.

Et comme vous l'avez vu d'autres développeurs R ont fait le travail difficile pour créer d'incroyables packages pour analyser rapidement les jeux de données et identifier les problèmes.

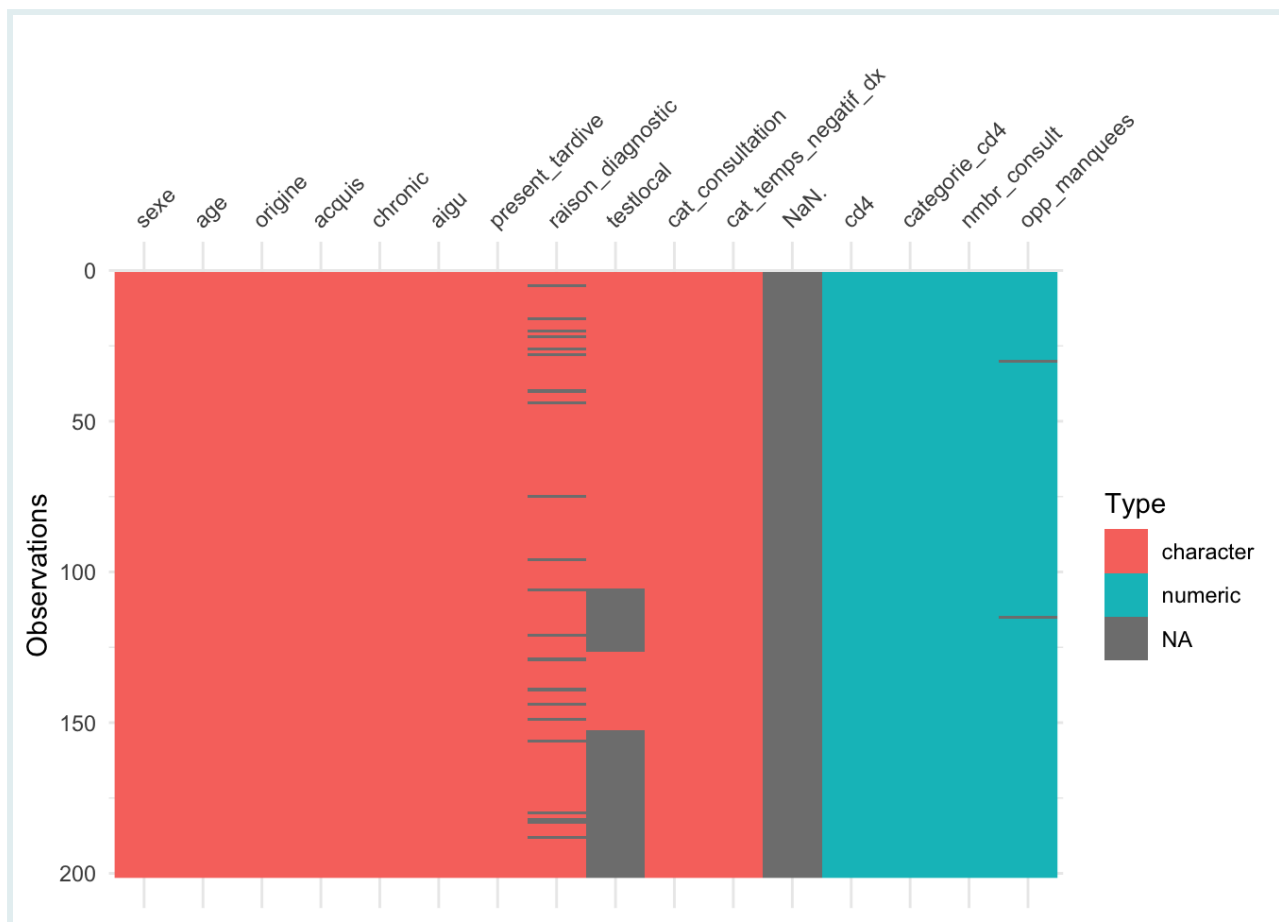
À partir de la prochaine leçon, nous allons aborder ces problèmes identifiés un par un, en commençant par le problème de noms de variables incohérents et désordonnés.

On se retrouve à la prochaine leçon !

Answer Key

Q : Repérer les problèmes de données avec `vis_dat()`

```
vis_dat(opp_manquees)
```



- La colonne NaN est complètement vide

Q : Générer des statistiques sommaires avec `skim()`

```
skim(opp_manquees)
```

Table 3: Data summary

Name	opp_manquees
Number of rows	201
Number of columns	16
Column type frequency:	
character	11
logical	1
numeric	4
Group variables	
	None





Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
sexe	0	1.00	1	5	0	3	0
age	0	1.00	3	5	0	3	0
origine	0	1.00	6	21	0	3	0
acquis	0	1.00	3	19	0	4	0
chronic	0	1.00	3	3	0	2	0
aigu	0	1.00	3	8	0	2	0
present_tardive	0	1.00	3	3	0	2	0
raison_diagnostic	21	0.90	9	50	0	8	0
testlocal	70	0.65	20	26	0	3	0
cat_consultation	0	1.00	28	30	0	3	0
cat_temps_negatif_dx	0	1.00	29	39	0	3	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
NaN.	201	0	NaN :	

Variable type: numeric

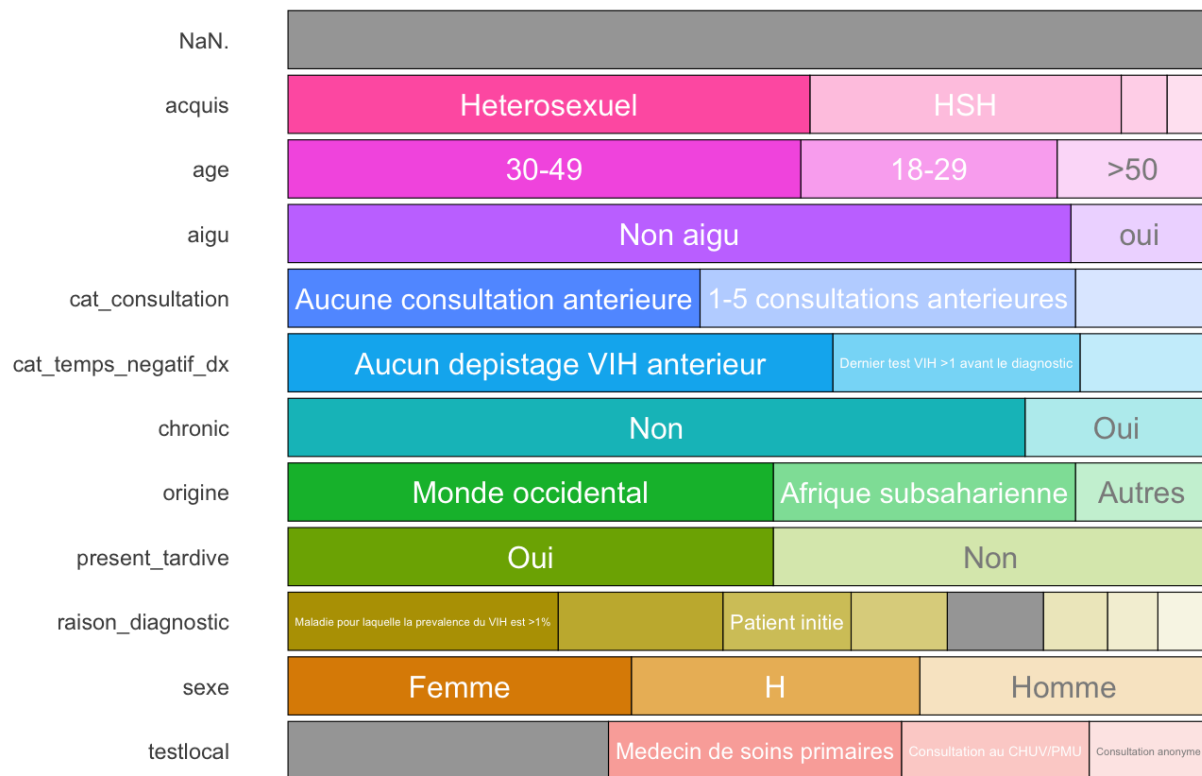
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
cd4	0	1.00	342.13	261.68	2	147	293	452	1261	
categorie_cd4	0	1.00	2.27	1.12	1	1	2	3	4	
nmbr_consult	0	1.00	1.98	2.90	0	0	1	3	21	
opp_manquees	2	0.99	1.83	3.01	0	0	0	3	17	

Q : Repérer les problèmes de données avec `inspect_cat()`

```
inspect_cat(opp_manquees) %>%
  show_plot()
```

Frequency of categorical levels in df::opp_manquees

Gray segments are missing values

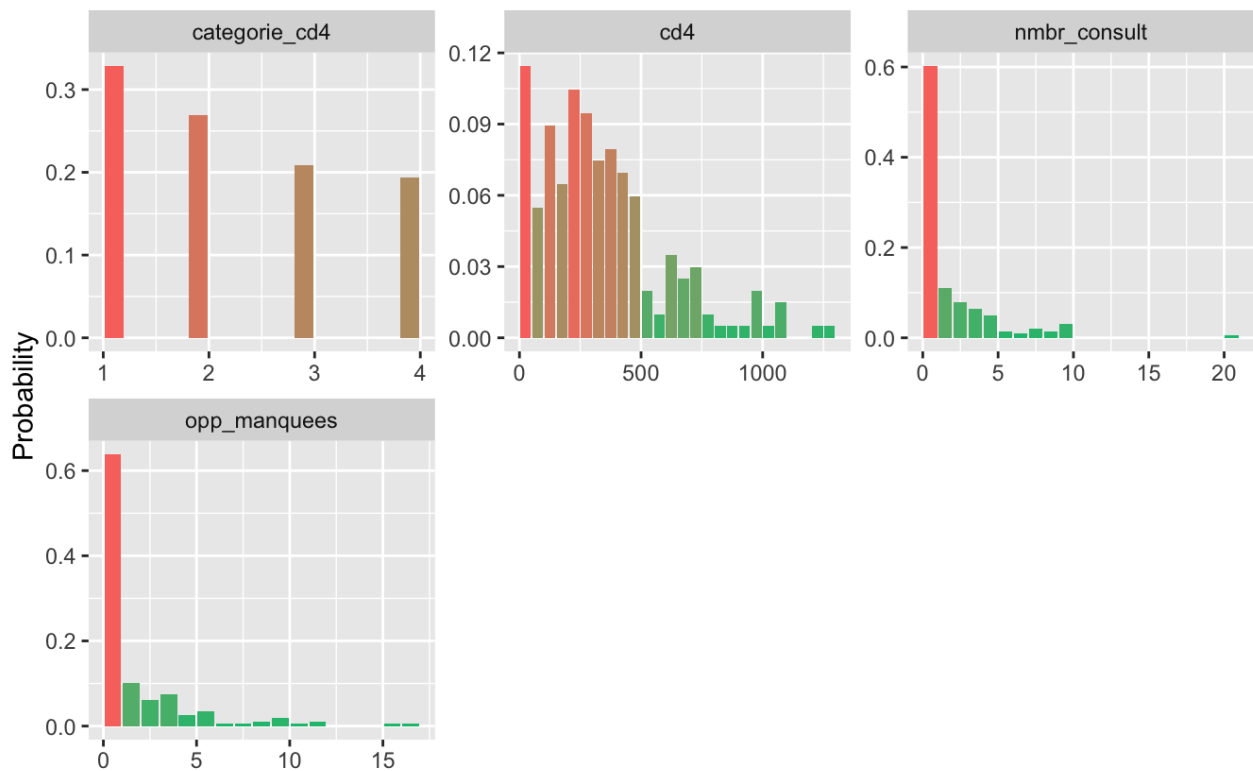


- La variable acute a 2 niveaux : Not acute et yes. Cela devrait être normalisé.
- La variable sex a 3 niveaux : Female, Male et M. Le M devrait être changé en Male.

Q : Types de variables avec inspect_num()

```
inspect_num(opp_manquees) %>%
  show_plot()
```

Histograms of numeric columns in df::opp_manquees



- La variable cd4category devrait être une variable de type facteur.

Q : Repérer les problèmes de données avec `tbl_summary()`

```
tbl_summary(opp_manquees)
```

Characteristic	N = 201 ¹
sexe	
Femme	75 (37%)
H	63 (31%)
Homme	63 (31%)
age	
>50	33 (16%)
18-29	56 (28%)
30-49	112 (56%)
origine	
Afrique subsaharienne	66 (33%)
Autres	29 (14%)
Monde occidental	106 (53%)
acquis	
Autre	10 (5.0%)
Heterosexuel	114 (57%)
HSH	68 (34%)
Usage de drogues IV	9 (4.5%)
chronic	
Non	161 (80%)
Oui	40 (20%)
cd4	293 (147, 452)
categorie_cd4	
1	66 (33%)
2	54 (27%)

Characteristic	N = 201 ¹
4	39 (19%)
aigu	
Non aigu	171 (85%)
oui	30 (15%)
present_tardive	
Non	95 (47%)
Oui	106 (53%)
raison_diagnostic	
Grossesse	14 (7.8%)
Introduction d'un traitement immunosuppresseur	1 (0.6%)
Maladie definissant le sida	21 (12%)
Maladie pour laquelle la prevalence du VIH est >1%	59 (33%)
Patient initie	28 (16%)
Risque epidem.	10 (5.6%)
Risque epidemiologique	11 (6.1%)
Suspicion d'infection aigue par le VIH	36 (20%)
Unknown	21
testlocal	
Consultation anonyme	26 (20%)
Consultation au CHUV/PMU	41 (31%)
Medecin de soins primaires	64 (49%)
Unknown	70
nmbr_consult	1.00 (0.00, 3.00)
cat_consultation	
>5 consultations anterieures	29 (14%)
1-5 consultations anterieures	82 (41%)
Aucune consultation anterieure	90 (45%)
cat_temps_negatif_dx	
Aucun depistage VIH anterieur	119 (59%)
Depistage du VIH l'annee precedente	28 (14%)
Dernier test VIH >1 avant le diagnostic	54 (27%)
opp_manquees	0.00 (0.00, 3.00)
Unknown	2
NaN.	0 (NA%)
Unknown	201
¹ n (%); Median (IQR)	

Q : Rapport de données avec `create_report()`

```
DataExplorer::create_report(opp_manquees)
```

References

Une partie du matériel de cette leçon a été adaptée des sources suivantes :

- Batra, Neale, et al. The Epidemiologist R Handbook. 2021. *Cleaning data and core functions*. <https://epirhandbook.com/en/cleaning-data-and-core-functions.html#cleaning-data-and-core-functions>
- Waring E, Quinn M, McNamara A, Arino de la Rubia E, Zhu H, Ellis S (2022). skimr: Compact and Flexible Summaries of Data. <https://docs.ropensci.org/skimr/> (website), <https://github.com/ropensci/skimr/>.

Contributeurs

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



AMANDA MCKINLEY

R Developer and Instructor, the GRAPH Network



KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about world improvement



LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education