
Data Cleaning Pipeline 1: Exploratory Data Analysis

Introduction	
Learning objectives	
Packages	
Dataset	
Visualizing missing data with <code>visdat::vis_dat()</code>	
Generating summary statistics with <code>skimr::skim()</code>	
Visualizing summary statistics with functions from <code>inspectdf</code>	
Exploring categorical variable levels with <code>gtsummary::tbl_summary()</code>	
Creating data reports with <code>DataExplorer::create_report()</code>	
Wrapping up	
WRAP UP!	
Answer Key	

Introduction



Data cleaning is the process of transforming raw, “messy” data into reliable data that can be properly analyzed. This entails identifying **inaccurate**, **incomplete**, or **improbable** data points and resolving data inconsistencies or errors, as well as renaming variable names to make them more clear and simple to manipulate.

Data cleaning tasks can often be tedious and time-consuming. A common joke among data analysts goes “80% of data science is cleaning the data and the remaining 20% is complaining about cleaning the data.” But data cleaning is an essential step of the data analysis process. A little bit of cleaning at the start of the data analysis process will go a long way to improving the quality of your analyses and the ease with which these analyses can be done. And a range of packages and functions in R can significantly simplify the data cleaning process.

In this lesson, we will begin to look at a typical data cleaning pipeline in R. The cleaning steps covered here are unlikely to be exactly what is needed for your own datasets, but they will certainly be a good starting point.

Let’s get started!

Learning objectives

- You can list typical operations involved in data cleaning process
 - You can diagnose dataset issues that warrant data cleaning through functions such as:
 - `visdat::vis_dat()`
 - `skimr::skim()`
 - `inspectdf::inspect_cat()`
 - `inspectdf::inspect_num()`
 - `gtsummary::tbl_summary()`
 - `DataExplorer::create_report()`
-

Packages

The packages loaded below will be required for this lesson:

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

Dataset

The primary dataset we will use in this lesson is from a study conducted in three healthcare centres in Zambesia, Mozambique. The study investigated individual factors associated with time to non-adherence to HIV care and treatment services. For the sake of this lesson, we will only be looking at a modified subset of the full dataset.

The full dataset can be obtained from [Zenodo](#), and the paper can be viewed [here](#).

Let's take a look at this dataset:

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

```
non_adherence
```

```
## # A tibble: 5 × 15
##   patient_id District `Health unit` Sex Age_35
##   <dbl>      <dbl>      <dbl> <chr> <chr>
## 1     10037         1          1 Male over 35
## 2     10537         1          1 F   over 35
## 3       5489         2          3 F   Under 35
## 4       5523         2          3 Male Under 35
## 5       4942         2          3 F   over 35
## # i 10 more variables: `Age at ART initiation` <dbl>,
## #   Education <chr>, Occupation <chr>, ...
```

The first step of data cleaning will be to explore this dataset in order to identify potential issues that warrant cleaning. This preliminary step is sometimes called “exploratory data analysis” or EDA.

Let's take a look at a few simple EDA commands in R that will help you identify possible data errors and inconsistencies.

Visualizing missing data with `visdat::vis_dat()`

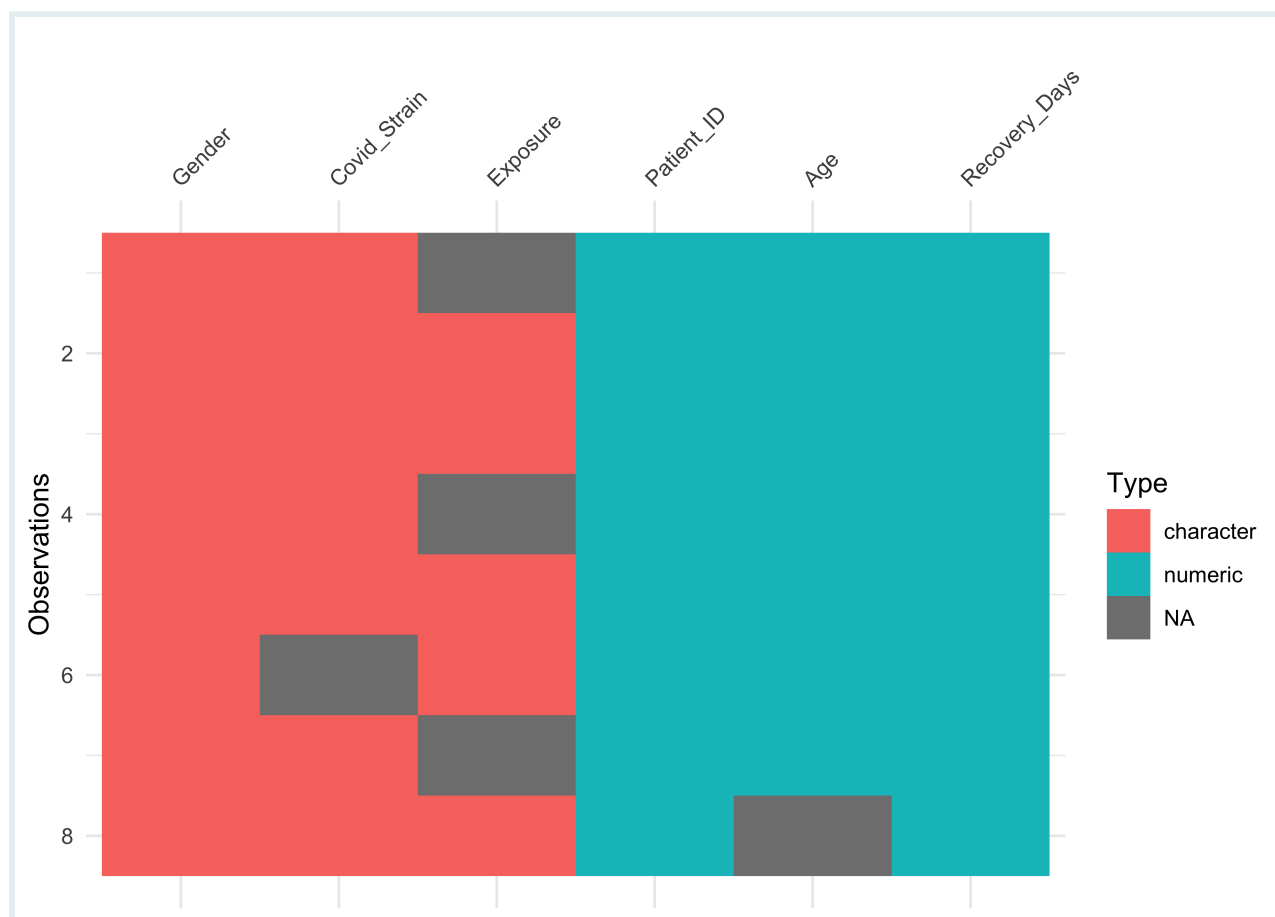
The `vis_dat()` function from the `visdat` package is a wonderful way to quickly visualize data types and missing values in a dataset. It creates a plot that shows a “zoomed out” spreadsheet view of your data frame: each row in the dataframe is represented by a single line on the plot.

Let's try it out with small mock dataset first to get an idea of how it works. Copy the following code to create a dataframe of 8 patients and their COVID-19 diagnosis and recovery information. As you can see below, some patients have missing information, represented as NA.

```
covid_pat <- tribble(
  ~Patient_ID, ~Age, ~Gender, ~Covid_Strain, ~Exposure,
  ~Recovery_Days,
  1, 25, "Male", "Alpha", NA, 10,
  2, 32, "Female", "Delta", "Hospital", 15,
  3, 45, "Male", "Beta", "Travel", 7,
  4, 19, "Female", "Omicron", NA, 21,
  5, 38, "Male", "Alpha", "Unknown", 14,
  6, 55, "Female", NA, "Community", 19,
  7, 28, "Female", "Omicron", NA, 8,
  8, NA, "Female", "Omicron", "Travel", 26
)
covid_pat
```

Now, let's use the function `vis_dat()` on our dataframe to get visual representation of the data types and missing values.

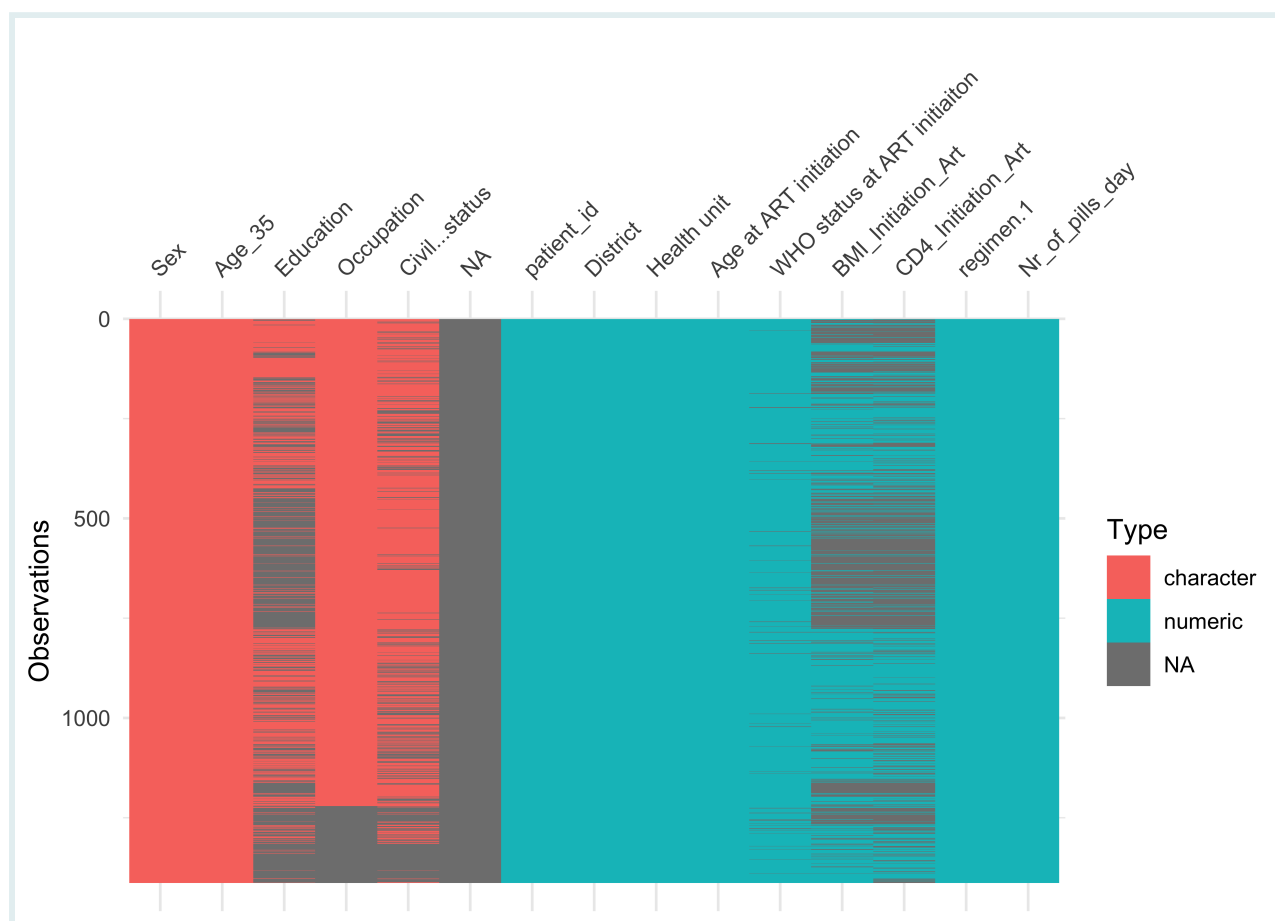
```
vis_dat(covid_pat)
```



That looks great! Each row in our dataframe is represented by a single line in the plot and different colours are used to illustrate the character (pink) and numeric (blue) variables, as well as missing values (grey). From this plot, we can tell that quite a few patients in our dataset are missing data for the Exposure variable.

Now let's turn back to our real-world dataset which is much larger and messier. Large real-world datasets may exhibit intricate patterns in the data that are difficult to spot without visualization, so using functions such as `vis_dat()` can be particularly useful. Let's try it out now!

```
vis_dat(non_adherence)
```



Great! From this view we can already see some problems:

- there seems to be a completely empty column in the data (the column NA which is fully gray)
- several variables have a lot of missing values (such as Education, BMI _Initiation _Art, and CD4 _Initiation _ART)
- the names of some variables are unclear/unclean (e.g., Age at ART initiation and WHO status at ART initiaiton have whitespaces in their names and Civil...status and regimen.1 have special characters, .)

In the next lesson, we will try to remedy these and other issues as best as possible during the data cleaning process. But for now, the goal is that we understand the functions used to identify them. So now that we have a solid grasp on how to visualize missing data with `vis_dat()`, let's take a look at another package and function that can help us generate summary statistics of our variables!

The following dataset was adapted from a study that investigated missed opportunities for HIV testing among patients newly presenting for HIV care at a Swiss university hospital. The full dataset can be found [here](#).

```
missed_ops <- read_csv(here("data/HIV_missed_ops.csv"))
```

Use the `vis_dat()` function to get a visual representation of the data. What potential issues can you spot based on the output?

Generating summary statistics with `skimr::skim()`

The `skim()` function from the `skimr` package provides a console-based overview of the data frame and a summary of every column (by class/type). Let's try it on our `covid_pat` mock data first to get an idea of how it works. First, as a reminder, here is our `covid_pat` dataframe:

```
covid_pat
```

Great, now let's run the `skim()` function!

```
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
Gender	0	1.00	4	6	0	2	0
Covid_Strain	1	0.88	4	7	0	4	0
Exposure	3	0.62	6	9	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	
Recovery_Days	0	1.00	15.00	6.68	7	9.50	14.5	19.50	26	

Amazing! As we can see, this function provides:

- An overview of the rows and columns of the dataframe
- The data type for each variable
- `n_missing`, the number of missing values for each variable
- `complete_rate`, the completeness rate for each variable
- A set of summary statistics: the mean, standard deviation and quartiles for numerical variables; and the frequency and proportions for categorical variables
- Spark histograms for the numerical variables

Now we can tell why this is so valuable with large dataframes! Let's return to our non-adherence dataset and run the `skim()` function on it.

```
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
Sex	0	1.00	1	4	0	2	0
Age_35	0	1.00	7	8	0	2	0
Education	776	0.45	4	10	0	5	0
Occupation	193	0.86	4	22	0	50	0
Civil...status	412	0.71	6	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	p100
patient_id	0	1.00	7364.75	3312.95	147.0	4705.00	7355.0	10098.0	24377.0
District	0	1.00	1.35	0.48	1.0	1.00	1.0	2.0	2.0
Health unit	0	1.00	1.87	0.90	1.0	1.00	2.0	3.0	3.0
Age at ART initiation	0	1.00	32.07	9.51	15.1	25.20	30.4	36.8	75.3
WHO status at ART initiation	45	0.97	1.80	0.98	1.0	1.00	1.0	3.0	4.0
BMI_Initiation_Art	588	0.58	20.37	3.54	9.0	18.12	20.1	22.2	52.0
CD4_Initiation_Art	674	0.52	398.19	243.64	3.0	233.00	343.0	538.0	1401.0
regimen.1	0	1.00	4.64	1.54	1.0	3.00	6.0	6.0	7.0
Nr_of_pills_day	0	1.00	1.51	0.59	1.0	1.00	1.0	2.0	3.0

From this output, we can identify some potential issues with our data:

- we can confirm that the column NA, which you saw from the `vis_dat()` output, is indeed completely empty: it has a `complete_rate` of 0
- the distribution for Age at ART initiation is skewed

Use `skim()` to obtain a detailed overview of the `missed_ops` dataset.

Great! Now that we know how to generate a short report of summary statistics for our variables. In the next section, we'll learn about a couple useful functions from the `inspectdf` package that allows us to visualize different summary stats.

Visualizing summary statistics with functions from `inspectdf`

While the `skimr::skim()` function gives you variable summaries in the console, you may sometimes want variables summaries in richer graphical form instead. For this, the functions `inspectdf::inspect_cat()` and `inspectdf::inspect_num()` can be used.

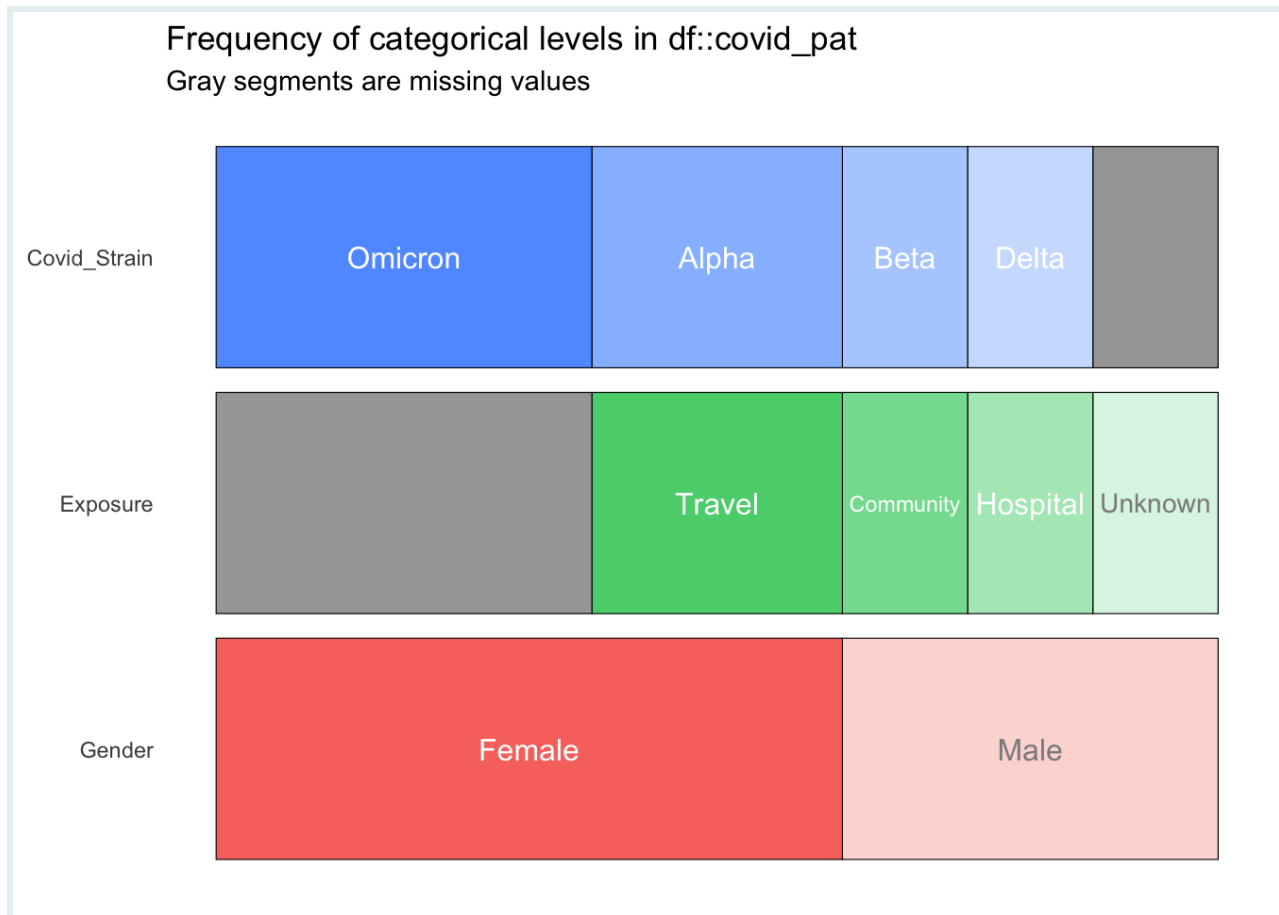
If you run `inspect_cat()` on a data object, you get a tabular summary of the categorical variables in the dataset (the important information is hidden in the `levels` column). Let's try it out on the `covid_pat` dataset first. As a reminder, here is our dataset:

```
covid_pat
```

```
inspect_cat(covid_pat)
```

The magic happens when you run `show_plot()` on the result from `inspect_cat()`:

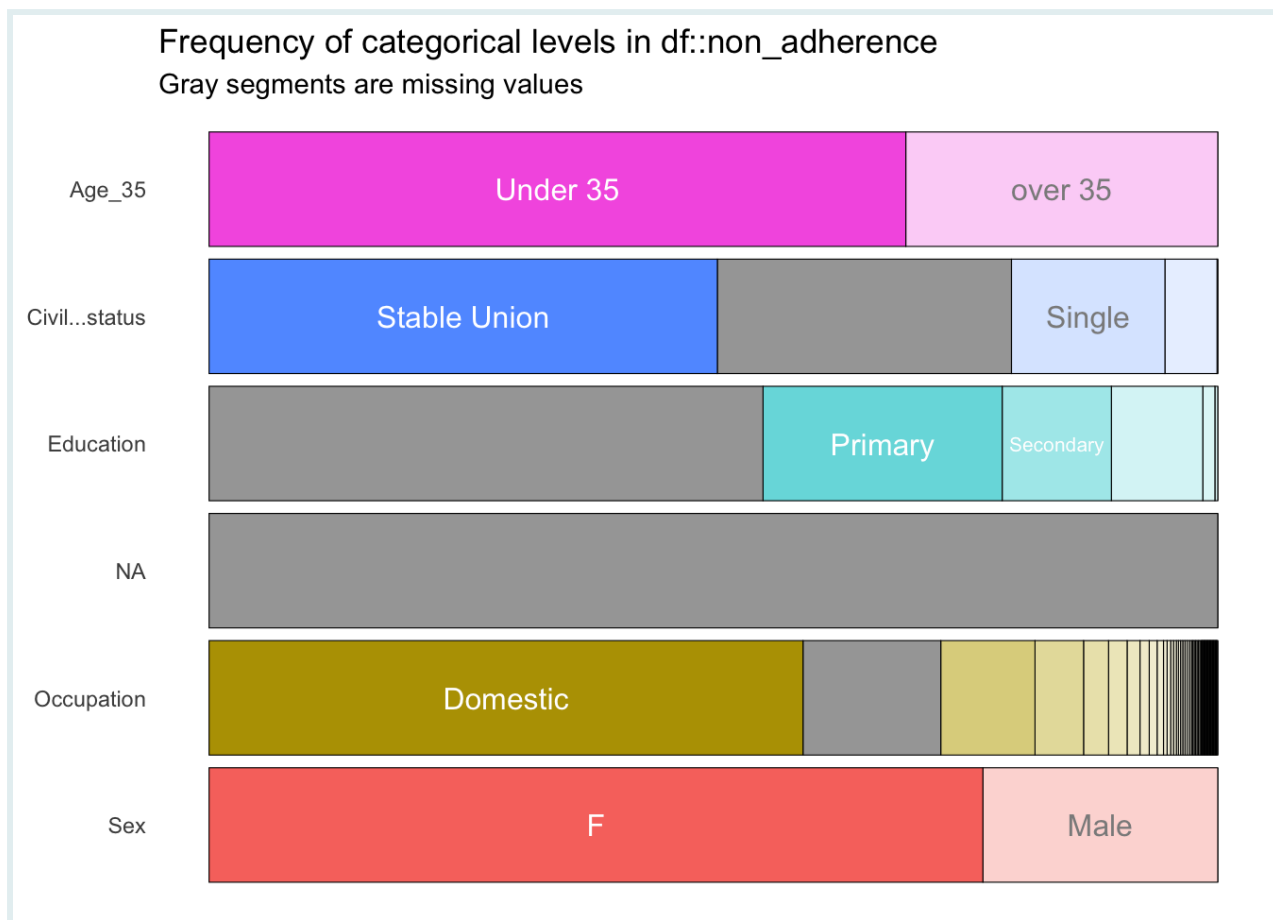
```
inspect_cat(covid_pat) %>%  
  show_plot()
```



That looks great! You get a nice summary figure showing the distribution of categorical variables! Variable levels are also nicely labelled (if there is sufficient space to show a label).

Now, let's try it out on our `non_adherence` dataset:

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



From here you can observe some issues with a few categorical variables:

- The variable level Under 35 is capitalized, whereas over 35 is not. It could be worth standardizing this.
- The variable sex has the levels F and Male. This too could be worth standardizing.
- As we have previously seen, NA is completely empty

Complete the following code to obtain a visual summary of the categorical variables in the missed_op dataset.

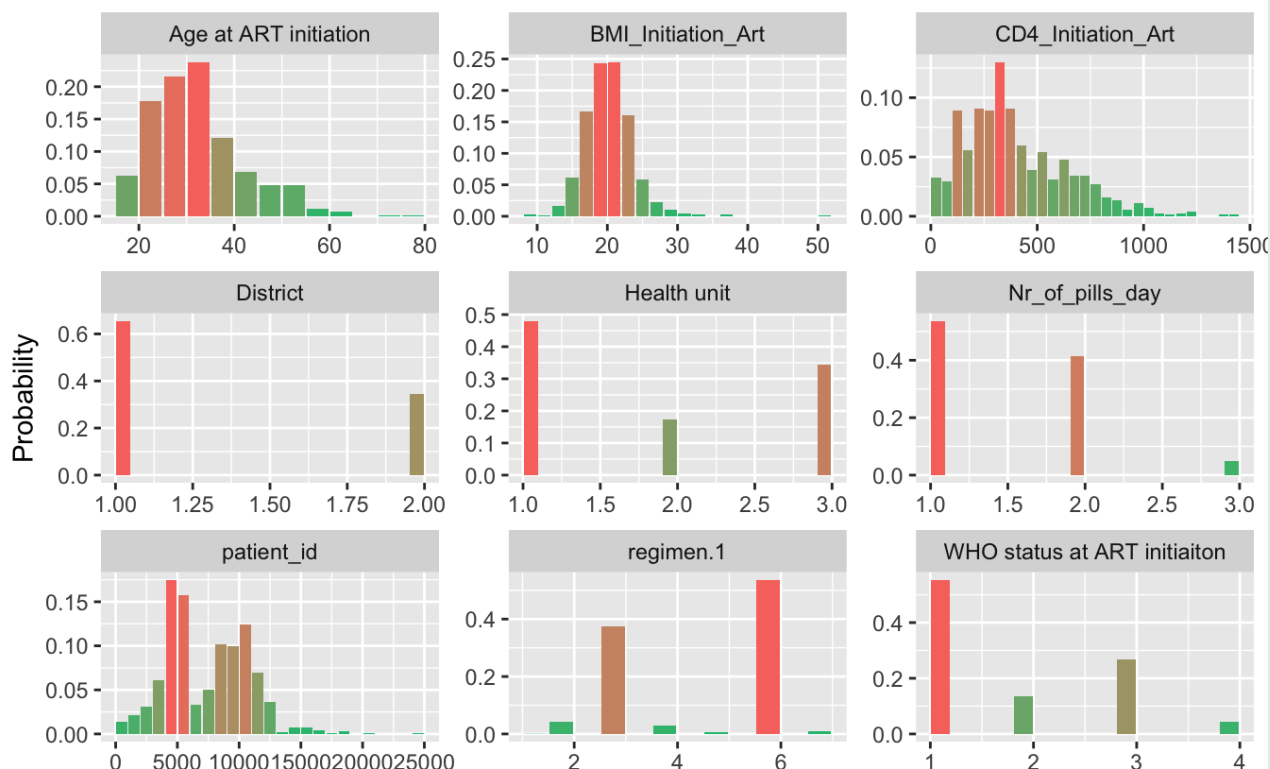
```
inspect___() %>%
```

How many potential data issues can you spot?

Similarly, you can obtain a summary plot for the numerical variables in the dataset with `inspect_num()`. Let's run this on our non_adherence dataset:

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

Histograms of numeric columns in df::non_adherence



From this output, you can notice that many variables which should be factor variables are coded as numeric. In fact, the only true numeric variables are `Age at ART initiation`, `BMI_Initiation_Art`, `CD4_Initiation_Art`, and `Nr_of_pills_day`. We will fix these issues in the next lesson when we move on to data cleaning. For now though, let's take a look at our another function that's particularly helpful for categorical variables!

Use `inspect_num` to create a histograms of your numeric variables in the `missed_op` dataset. Are the numeric variable types correct?

Exploring categorical variable levels with `gtsummary::tbl_summary()`

While the `inspect_cat()` function is useful for a graphical overview of categorical variables, it doesn't provide frequency counts or percentages for the various levels. For this, the `tbl_summary()` from the `gtsummary` package is particularly helpful! The output is particularly long so we'll look at the tibble form and show a photo of the important part for our dataset. You can explore the whole output as you code along at home.

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

```
## # A tibble: 104 × 2  
##   `**Characteristic**` `**N = 1,413**`  
##   <chr>                <chr>  
## 1 patient_id          7,355 (4,705, 10,098)  
## 2 District            <NA>  
## 3 1                    925 (65%)  
## 4 2                    488 (35%)  
## 5 Health unit         <NA>  
## 6 1                    678 (48%)  
## 7 2                    247 (17%)  
## 8 3                    488 (35%)  
## 9 Sex                <NA>  
## 10 F                 1,084 (77%)  
## # i 94 more rows
```

That looks great! As we can see it outputs a summary of the frequency and percentages for categorical variables and the median and IQR for numeric variables.

Below is a photo of part of the output where we can see additional issues with our data that wasn't clear using the `inspect_cat()` function. Some values from our `Occupation` variable are capitalized, whereas others are all in lower case.

Office worker	1 (<0.1%)
Police	3 (0.2%)
professor	11 (0.9%)
Professor	35 (2.9%)
Receptionist	1 (<0.1%)
Retired	2 (0.2%)
rock cutter	9 (0.7%)

This means that R doesn't recognize them as being the same value which would be problematic during analysis. We'll fix these errors in the next lesson, for now though let's move on to our last function for exploratory data analysis!

WATCH OUT



The `tbl_summary()` function is not appropriate for checking variable names. As you may have noticed, all whitespaces were replaced by a period. For example, `Age at ART initiation` becomes `Age.at.ART.initiation`.

Use `tbl_summary()` to output a summary of your `missed_ops` dataset. Can you identify any additional data issues?

Creating data reports with `DataExplorer::create_report()`

Finally, the `create_report()` function from the `DataExplorer` package creates a full data profile of a provided data frame: an HTML file with basic statistics and distribution visualizations.

Let's run this function on the `non_adherence` dataframe and observe its output. Note that it may take a while to run. If it takes too long, you can run it on a dataset sample, rather than the entire dataset.

```
create_report(non_adherence)
```

As you can see, the report is quite extensive. We won't go over all of the outputs from this data report because many of the graphical outputs are the same as those we have seen with previous functions! However, some new outputs include:

- QQ plot to assess the normality of numeric variables
- Correlation analysis (when there are sufficient complete rows)
- Principal component analysis (when there are sufficient complete rows)

Feel free to explore the [package documentation](#) on your own time.

Create a data report for your `missed_ops` data using the `create_report()` function!

Wrapping up

By familiarizing ourselves with the data, we have been able to identify some potential problems that may need to be addressed before the data are used in an analysis.

And as you have seen, the actual code needed to do this data exploration is very little; other R developers have done the difficult work for us by building amazing packages to quickly scan datasets and identify issues.

From the next lesson, we will begin to take on these identified issues one by one, starting with the problem of inconsistent, messy variable names.

See you in the next lesson!

Contributors

The following team members contributed to this lesson:



LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network

A firm believer in science for good, striving to ally programming, health and education



KENE DAVID NWOSU

Data analyst, the GRAPH Network

Passionate about world improvement

References

Some material in this lesson was adapted from the following sources:

- 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/>.

[XXX DEMO HERE XXX]

REMINDER



[XXX SIDE NOTES HERE XXXX]

RECAP



[XXX RECAP HERE XXX]

PRACTICE



(in RMD)

[XXX PRACTICE QUESTIONS HERE XXX] - NOTE: Make sure to add the answers to all practice at the end of the lesson! We are no longer using autograders for the GF project.

[XXX PRACTICE QUESTIONS HERE XXX] - NOTE: Make sure to add the answers to all practice at the end of the lesson! We are no longer using autograders for the GF project.

CHALLENGE



[XXX CHALLENGE PRACTICE QUESTIONS HERE XXX] - NOTE: Make sure to add the answers to all practice at the end of the lesson! We are no longer using autograders for the GF project.

PRO TIP



[XXX PRO-TIPS HERE XXX]

VOCAB



[XXX RELEVANT VOCUBULARY WORDS HERE XXX]

SIDE NOTE



[XXX SIDE NOTES HERE XXXX]

WATCH OUT



[XXX TIPS HERE XXX]

KEY POINT



[XXX KEY POINTS HERE XXX]

Error?



[XXX ERRORS HERE XXX]

RSTUDIO CLOUD



[XXX RSTUDIO CLOUD TIPS HERE XXX]

WRAP UP!

[XXX NICE WRAP UP MESSAGE OR SUMMARY IF NEEDED HERE XXX]

Answer Key

Contributors

The following team members contributed to this lesson:



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

(make sure to update the contributor list accordingly!)