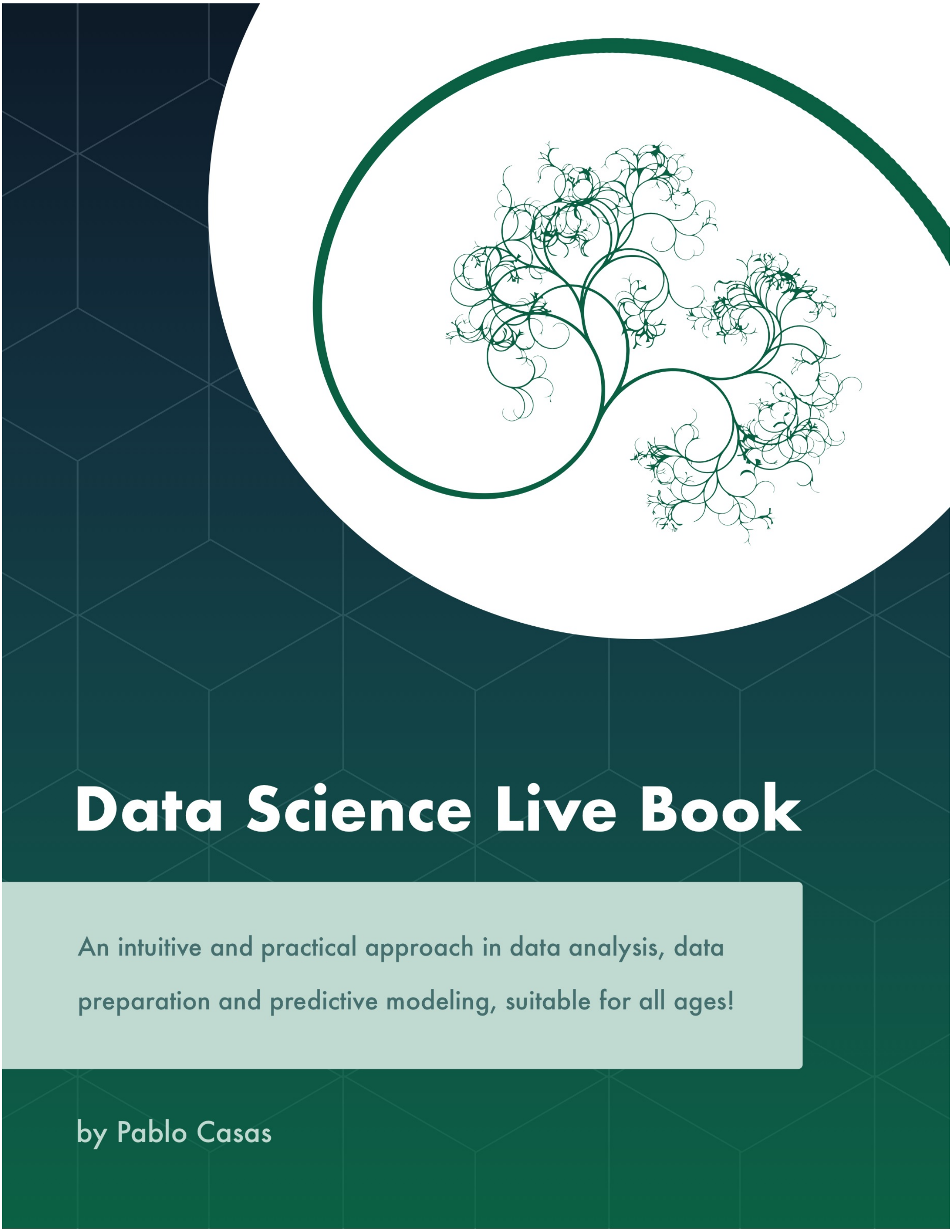
Table of Contents

0Data Science Live Book



0.1A book to learn data science, data analysis and machine learning, suitable for all ages!

0.1.1Last update: 2017-04-21

0.2*What does it cover?*

This live book (`#dsLiveBook`) covers common aspects in predictive modeling:

- A. **Exploratory Data Analysis**
- B. **Data Preparation**
- C. **Selecting Best Variables**
- D. **Scoring Data**
- E. **Assessing Model Performance**

0.3Upcoming updates

More info about methodological aspects in data preparation.

0.4What programming language do I need?

Most of the concepts are independent from the language, the focus is on general concepts. But when technical example is required it is done in R language, using the `funModeling` package which you can install by doing: `install.packages("funModeling")`

0.5Book Focus

- **Stimulate intuition** behind concepts: The explanation of how to interpret results brings a deeper understanding of **what is being done**, boosting the freedom to use that knowledge in other situations regardless of the language.
- Regarding technical aspects.... model creation consumes around **10%** of almost any predictive modeling project; the **Live Book** and `funModeling` will try to cover remaining 90%.

Why a live book ? Hopefully this book barely has an end, it will be updated periodically. And you can contribute! below the github link.

First published at: livebook.datascienceheroes.com

1.1Profiling Data

1.1What is this about?

Quantity of zeros, NA, Inf, unique values; as well as the data type may lead to a good or bad model. Here's an approach to cover the very first step in data modeling.

```
## Loading funModeling !
library(funModeling)
library(dplyr)
data(heart_disease)
```

1.1Checking NA, zeros, data type and unique values

```
my_data_status=df_status(heart_disease)
```

```
##                  variable q_zeros p_zeros q_na p_na q_inf p_inf    type
## 1                     age       0    0.00    0 0.00     0     0 integer
## 2                  gender       0    0.00    0 0.00     0     0  factor
## 3              chest_pain       0    0.00    0 0.00     0     0  factor
## 4  resting_blood_pressure       0    0.00    0 0.00     0     0 integer
## 5       serum_cholestoral       0    0.00    0 0.00     0     0 integer
## 6     fasting_blood_sugar     258   85.15    0 0.00     0     0  factor
## 7         resting_electro     151   49.83    0 0.00     0     0  factor
## 8           max_heart_rate       0    0.00    0 0.00     0     0 integer
## 9             exer_angina     204   67.33    0 0.00     0     0 integer
## 10                oldpeak      99   32.67    0 0.00     0     0 numeric
## 11                  slope       0    0.00    0 0.00     0     0 integer
## 12      num_vessels_flour     176   58.09    4 1.32     0     0 integer
## 13                   thal       0    0.00    2 0.66     0     0  factor
## 14 heart_disease_severity     164   54.13    0 0.00     0     0 integer
## 15           exter_angina     204   67.33    0 0.00     0     0  factor
## 16      has_heart_disease       0    0.00    0 0.00     0     0  factor
##    unique
## 1      41
## 2       2
## 3       4
## 4      50
## 5     152
## 6       2
## 7       3
## 8      91
## 9       2
## 10     40
## 11      3
## 12      4
## 13      3
## 14      5
## 15      2
## 16      2
```

- `q_zeros` : quantity of zeros (`p_zeros` : in percentage)
- `q_inf` : quantity of infinite values (`p_inf` : in percentage)
- `q_na` : quantity of NA (`p_na` : in percentage)
- `type` : factor or numeric
- `unique` : quantity of unique values

1.0.1Why are these metrics important?

- **Zeros**: Variables with **lots of zeros** may be not useful for modeling, and in some cases it may dramatically bias the model.
- **NA**: Several models automatically exclude rows with NA (**random forest**, for

example). As a result, the final model can be biased due to several missing rows because of only one variable. For example, if the data contains only one out of 100 variables with 90% of NAs, the model will be training with only 10% of original rows.
- **Inf**: Infinite values may lead to an unexpected behavior in some functions in R.
- **Type**: Some variables are encoded as numbers, but they are codes or categories, and the models **don't handle them** in the same way.
- **Unique**: Factor/categorical variables with a high number of different values (~30), tend to do overfitting if categories have low cardinality, (**decision trees**, for example).

1.0.2Filtering unwanted cases

The function `df_status` takes a data frame and returns a the status table to quickly remove unwanted cases.

Removing variables with high number of NA/zeros

```
# Removing variables with 60% of zero values
vars_to_remove=filter(my_data_status, p_zeros > 60)  %>% .$variable
vars_to_remove
```

```
## [1] "fasting_blood_sugar" "exer_angina"         "exter_angina"
```

```
## Keeping all columns except vars_to_remove
heart_disease_2=select(heart_disease, -one_of(vars_to_remove))
```

Ordering data by percentage of zeros

```
arrange(my_data_status, -p_zeros) %>% select(variable, q_zeros, p_zeros)
```

```
##                    variable q_zeros p_zeros
## 1      fasting_blood_sugar     258   85.15
## 2              exer_angina     204   67.33
## 3             exter_angina     204   67.33
## 4         num_vessels_flour     176   58.09
## 5   heart_disease_severity     164   54.13
## 6           resting_electro     151   49.83
## 7                  oldpeak      99   32.67
## 8                      age       0    0.00
## 9                   gender       0    0.00
## 10              chest_pain       0    0.00
## 11 resting_blood_pressure       0    0.00
## 12       serum_cholestoral       0    0.00
## 13          max_heart_rate       0    0.00
## 14                   slope       0    0.00
## 15                    thal       0    0.00
## 16       has_heart_disease       0    0.00
```

1.1Profiling categorical variable

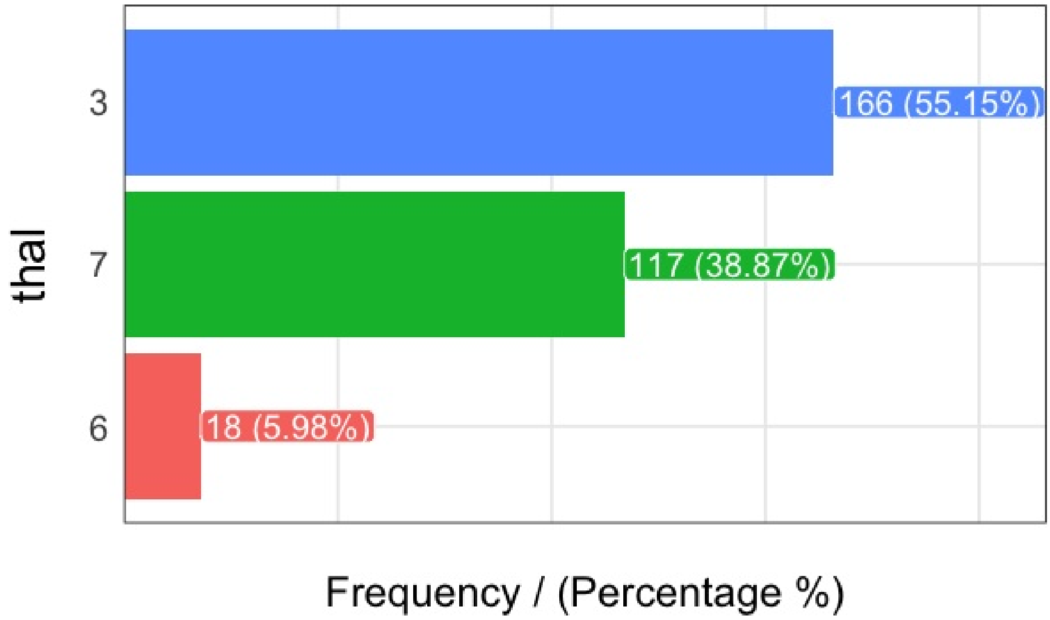
Make sure you have the latest funModeling version (>= 1.3).

Frequency or distribution analysis is made simple by the `freq` function. It retrieves the distribution in a table and a plot (by default) which shows the distribution in absolute and relative numbers.

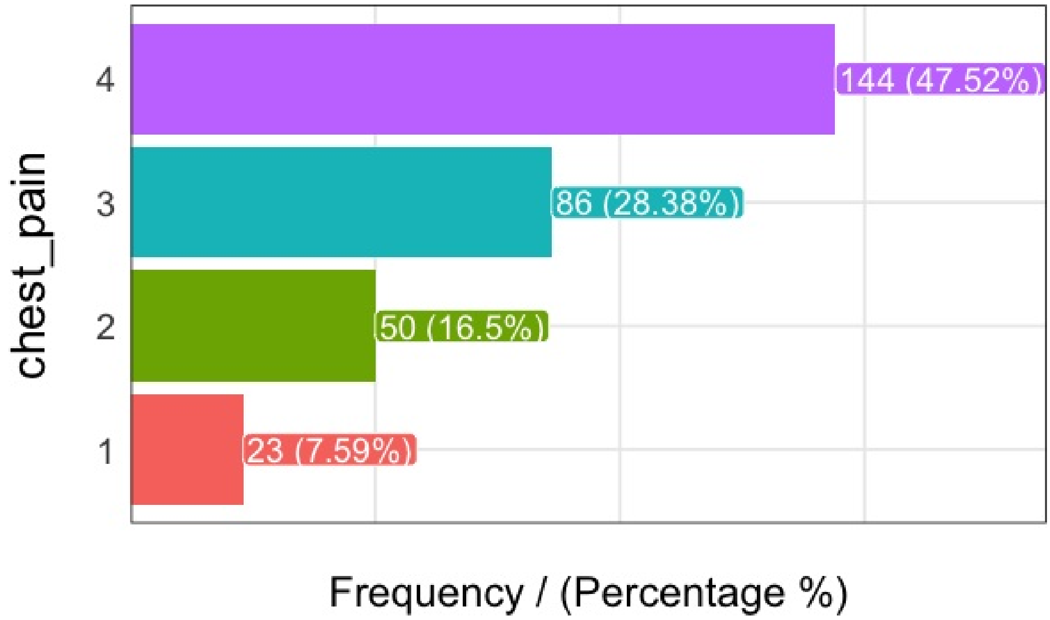
If you want the distribution for two variables:

```
freq(data=heart_disease, str_input = c('thal','chest_pain'))
```

```
## Warning in if (is.na(str_input)) {: the condition has length > 1 and only
## the first element will be used
```




```
##   thal frequency percentage cumulative_perc
## 1    3       166      55.15           55.15
## 2    7       117      38.87           94.02
## 3    6        18       5.98          100.00
```




```
##   chest_pain frequency percentage cumulative_perc
## 1          4       144      47.52           47.52
## 2          3        86      28.38           75.90
## 3          2        50      16.50           92.40
## 4          1        23       7.59          100.00
```

```
## [1] "Variables processed: thal, chest_pain"
```

As well as in the remaining `funModeling` functions, if `str_input` is missing it will run for all factor or character variables present in given data frame:

```
freq(data=heart_disease)
```

Also, as the other plot functions in the package, if there is the need of exporting plots, add the `path_out` parameter (it will create the folder if it's not created yet)

```
freq(data=heart_disease, path_out='my_folder')
```

4High Cardinality Variable in Descriptive Stats

4.1What is this about?

A **high cardinality** variable is one in which it can take *many* different values. For example country.

This chapter will cover cardinality reduction based on Pareto rule, using the `freq` function which gives a quick view about where the most of values are concentrated and variable distribution.

4.2High Cardinality in Descriptive Statistics

The following example contains a survey of 910 cases, with 3 columns: `person`, `country` and `has_flu`, which indicates having such illness in the last month.

```
library(funModeling)
```

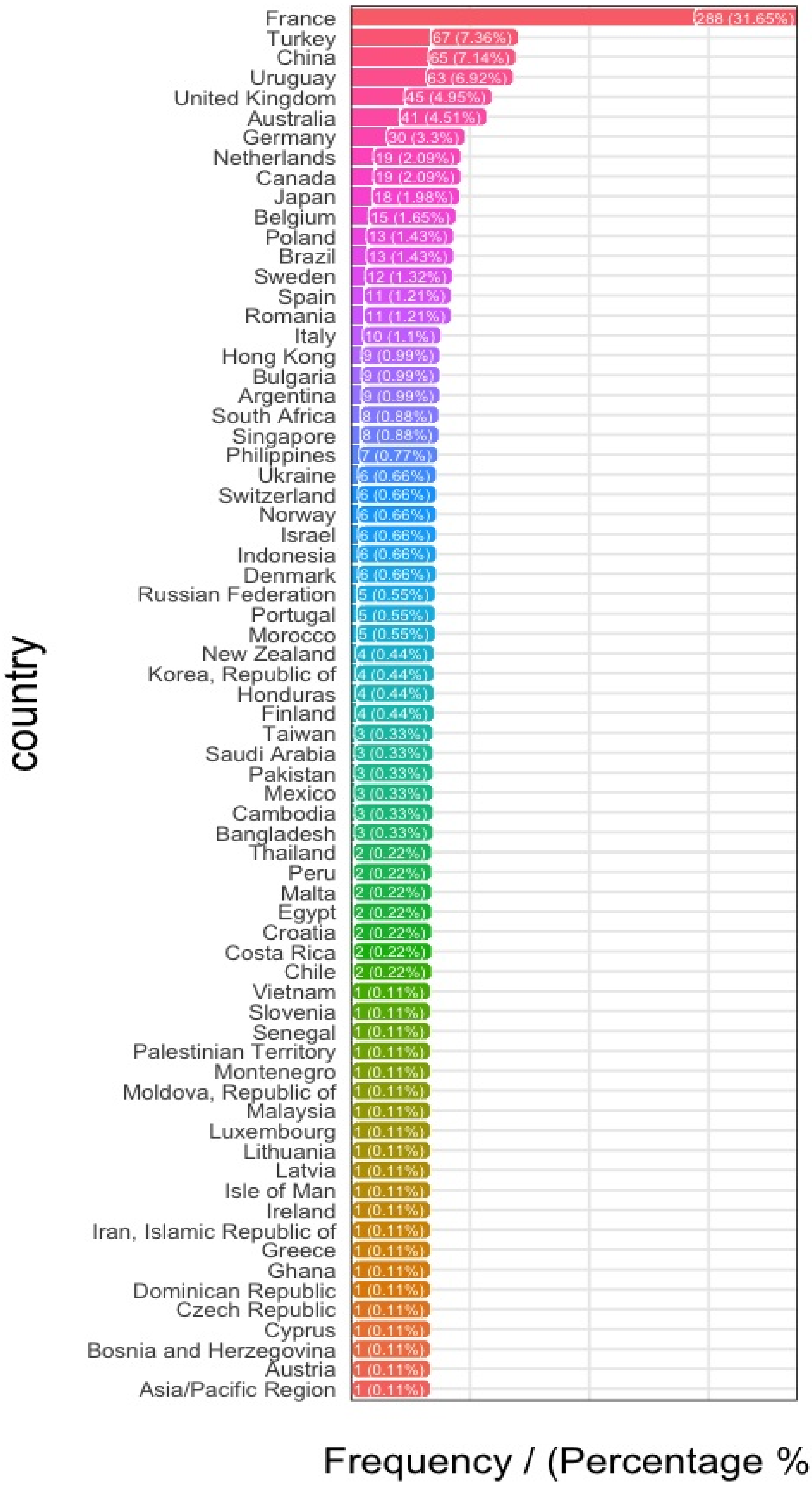
`data_country` data comes inside `funModeling` package (please update to release 1.6).

Quick `data_country` profiling (first 10 rows)

```
# plotting first 10 rows
head(data_country, 10)
```

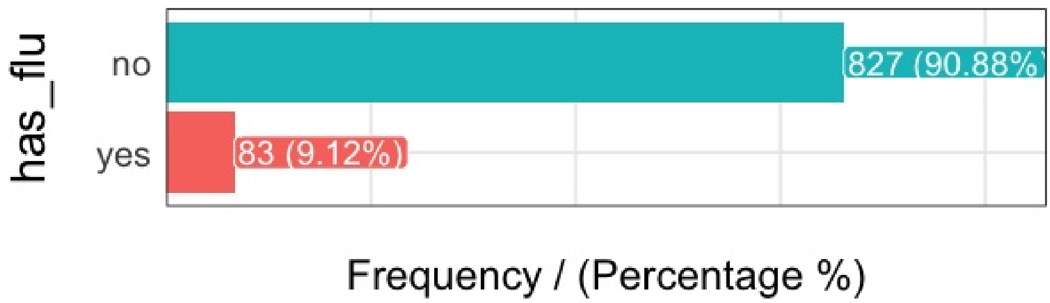
```
##     person     country has_flu
## 478    478      France      no
## 990    990      Brazil      no
## 606    606      France      no
## 575    575 Philippines      no
## 806    806      France      no
## 232    232      France      no
## 422    422      Poland      no
## 347    347     Romania      no
## 858    858     Finland      no
## 704    704      France      no
```

```
# exploring data, displaying only first 10 rows
head(freq(data_country, "country"), 10)
```


country
Frequency / (Percentage %
France 288 (31.65%)
Turkey 67 (7.36%)
China 65 (7.14%)
Uruguay 63 (6.92%)
United Kingdom 45 (4.95%)
Australia 41 (4.51%)
Germany 30 (3.3%)
Netherlands 19 (2.09%)
Canada 19 (2.09%)
Japan 18 (1.98%)
Belgium 15 (1.65%)
Poland 13 (1.43%)
Brazil 13 (1.43%)
Sweden 12 (1.32%)
Spain 11 (1.21%)
Romania 11 (1.21%)
Italy 10 (1.1%)
Hong Kong 9 (0.99%)
Bulgaria 9 (0.99%)
Argentina 9 (0.99%)
South Africa 8 (0.88%)
Singapore 8 (0.88%)
Philippines 7 (0.77%)
Ukraine 6 (0.66%)
Switzerland 6 (0.66%)
Norway 6 (0.66%)
Israel 6 (0.66%)
Indonesia 6 (0.66%)
Denmark 6 (0.66%)
Russian Federation 5 (0.55%)
Portugal 5 (0.55%)
Morocco 5 (0.55%)
New Zealand 4 (0.44%)
Korea, Republic of 4 (0.44%)
Honduras 4 (0.44%)
Finland 4 (0.44%)
Taiwan 3 (0.33%)
Saudi Arabia 3 (0.33%)
Pakistan 3 (0.33%)
Mexico 3 (0.33%)
Cambodia 3 (0.33%)
Bangladesh 3 (0.33%)
Thailand 2 (0.22%)
Peru 2 (0.22%)
Malta 2 (0.22%)
Egypt 2 (0.22%)
Croatia 2 (0.22%)
Costa Rica 2 (0.22%)
Chile 2 (0.22%)
Vietnam 1 (0.11%)
Slovenia 1 (0.11%)
Senegal 1 (0.11%)
Palestinian Territory 1 (0.11%)
Montenegro 1 (0.11%)
Moldova, Republic of 1 (0.11%)
Malaysia 1 (0.11%)
Luxembourg 1 (0.11%)
Lithuania 1 (0.11%)
Latvia 1 (0.11%)
Isle of Man 1 (0.11%)
Ireland 1 (0.11%)
Iran, Islamic Republic of 1 (0.11%)
Greece 1 (0.11%)
Ghana 1 (0.11%)
Dominican Republic 1 (0.11%)
Czech Republic 1 (0.11%)
Cyprus 1 (0.11%)
Bosnia and Herzegovina 1 (0.11%)
Austria 1 (0.11%)
Asia/Pacific Region 1 (0.11%)

```
##           country frequency percentage cumulative_perc
## 1          France       288      31.65           31.65
## 2          Turkey        67       7.36           39.01
## 3           China        65       7.14           46.15
## 4         Uruguay        63       6.92           53.07
## 5  United Kingdom        45       4.95           58.02
## 6       Australia        41       4.51           62.53
## 7         Germany        30       3.30           65.83
## 8          Canada        19       2.09           67.92
## 9     Netherlands        19       2.09           70.01
## 10          Japan        18       1.98           71.99
```

```
# exploring data
freq(data_country, "has_flu")
```




```
##   has_flu frequency percentage cumulative_perc
## 1      no       827      90.88           90.88
## 2     yes        83       9.12          100.00
```

The last table shows there are **70 different countries**, and ~9% of people who had flu - `has_flu="yes"` .

But many of them have almost no participation in the data. This is the *long tail*, so one technique to reduce cardinality is to keep those categories that are present the a high percentahge of data share, for example 70, 80 or 90%, the Pareto principle.

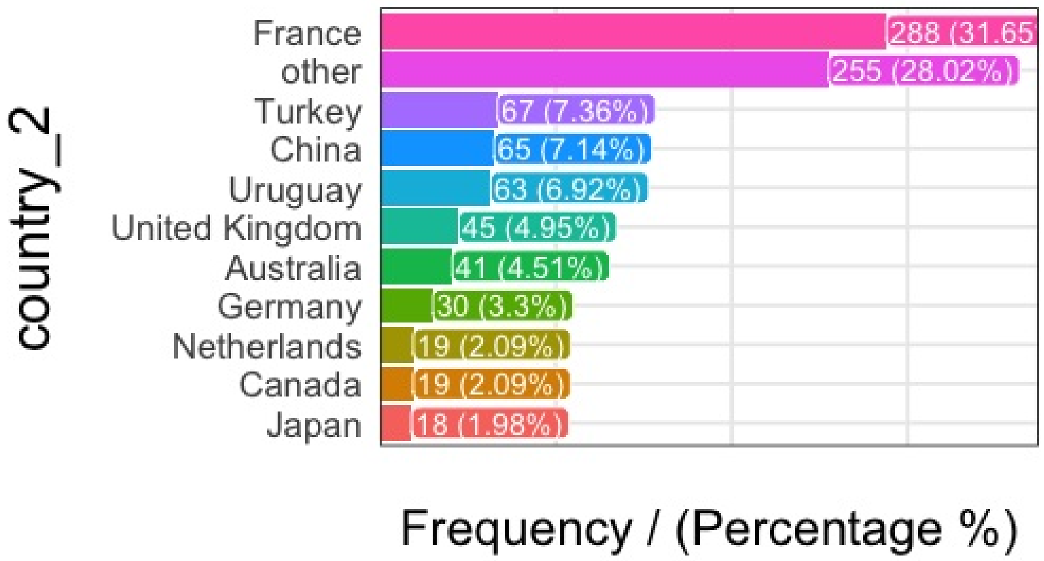
```
# 'freq' function, from 'funModeling' package, retrieves the cumulative_percentage
 that will help to do the cut.
country_freq=freq(data_country, 'country', plot = F)

# Since 'country_freq' is an ordered table by frequency, let's inspect the first 1
0 rows with the most share.
country_freq[1:10,]
```

```
##          country frequency percentage cumulative_perc
## 1         France       288      31.65           31.65
## 2         Turkey        67       7.36           39.01
## 3          China        65       7.14           46.15
## 4        Uruguay        63       6.92           53.07
## 5 United Kingdom        45       4.95           58.02
## 6      Australia        41       4.51           62.53
## 7        Germany        30       3.30           65.83
## 8         Canada        19       2.09           67.92
## 9    Netherlands        19       2.09           70.01
## 10         Japan        18       1.98           71.99
```

So 10 countries represent more the 70% of cases. We can assign the category `other` to the remaining cases and plot:

```
data_country$country_2=ifelse(data_country$country %in% country_freq[1:10,'country'
], data_country$country, 'other')
freq(data_country, 'country_2')
```



```
##          country_2 frequency percentage cumulative_perc
## 1           France       288      31.65           31.65
## 2            other       255      28.02           59.67
## 3           Turkey        67       7.36           67.03
## 4            China        65       7.14           74.17
## 5          Uruguay        63       6.92           81.09
## 6   United Kingdom        45       4.95           86.04
## 7        Australia        41       4.51           90.55
## 8          Germany        30       3.30           93.85
## 9           Canada        19       2.09           95.94
## 10     Netherlands        19       2.09           98.03
## 11           Japan        18       1.98          100.00
```

4.3Final comments

Low representative categories are sometimes errors in data, such as having: `Egypt` , `Eggypt.` , and may give some evidence in bad habbits collecting data and/or possible errors when collecting from the source.

There is no general rule to shrink data, it depends on each case.

Next recommended chapter: **High Cardinality Variable in Predictive Modeling**