TSAR Project Assignment Part 2 (5 points)

Data Wrangling with dplyr and tidyr

gwendolin.wilke@fhnw.ch

Task 1 (1 point)

- Go to your code from Project 1. Save the private data set you generated at the beginning of Project 1 to disk (if you didn't do so already.)
- Now create a new Quarto document for Project 2. Call it "`P2-<your TSAR-ID>-<your last name>.qmd`". (E.g., "P2-12-Burri.qmd").
- In your new Quarto document, load your private data set (the one you just saved to disk from P1) and store it in the variable `myLCdata`.
- Download `dirtyfy.rds` from GitHub and load it. It contains a function that accepts 2 arguments: a data frame and an id number. Apply the function to `myLCdata` and use your personal TSAR ID for the id number argument. Store the resulting data frame in the variable `myLCdata_dirty`.
- Install and load the package `DataExplorer`. The function `create_report()` is now available to you. Apply it to `myLCdata_dirty` to automatically generate an EDA report (EDA = *Explorator Data Analysis*). **Note:** Make sure to **not execute** `create_report()` within your Quarto document. Instead, execute it in the console and only include the code statement you used within Quarto for documentation purposes.
  - *Remark:* `create_report()` *lets you specify a target variable. You don't need this option. (It is useful for classification tasks, e.g., providing correlation plots of all variables with the target. If you want to see what it does, load the whole data set* `LCdata` *and specify* `int_rate` *as target variable.)*
- In the generated EDA report, look at the graph titled `Missing Data Profile`. It shows the percentage of missing values (`NA`s) per attribute. Cut this graph out, save it on disk as a picture and include the picture in your Quarto document for documentation purposes. To practice data wrangling with `dplyr` and `tidyr`, Task 2 will ask you to create similar graphs.

Task 2 (4 points)

The function you applied to your private data set in Task 1 ingested the following special values to the **character attributes** of your private data set (with the goal to "make it dirty"):

- **`NA` (Not Available).** Indicates missing values that R recognizes as missing.

  Remark: In the data cleaning step - which we do not tackle in this lecture - one would need to decide what to do with these values. Options are to remove the corresponding rows, to remove the corresponding columns or to impute (fill up) the missing values.

- **The string `"n/a"`.** We pretend this string was used in the source data set to indicate missing data. Since it is a string, R does not recognize it as missing.

  Remark: In the data cleaning step one would need to look out for such special strings and transform them into `NA`, if applicable.

- **A space (`" "`).** Another special string.

  Remark: Notice that it may or may not indicate missing data!

- **The empty string (`""`).** Another special string.

The following special values where ingested to the **numerical attributes** of your data set:

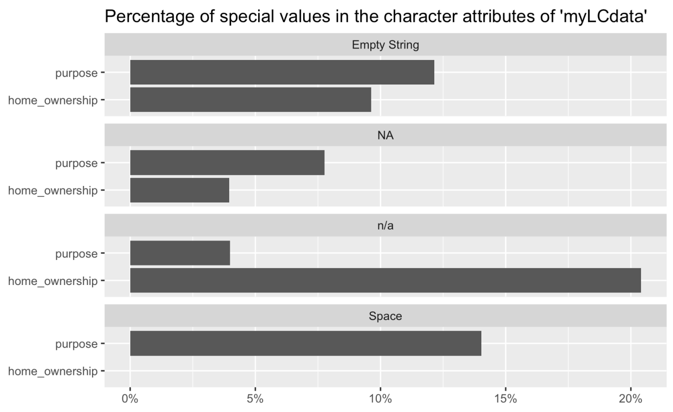
- **`NA`.**
- **`NaN` (Not a Number).** Results from invalid computations such as `0/0`.

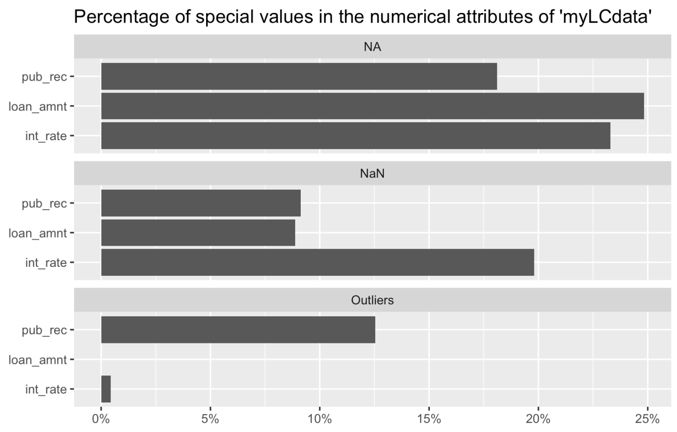
  Remark: In the data cleaning step, one would need to decide what to do with these values. E.g., try to correct them or treat them as missing by transforming them into `NA`.

- **Outliers.** These were actually not ingested, but already present in the data. There is no rigid mathematical definition of what constitutes an outlier. In this task, we use Tukey's fences: We count values as outliers if they lie outside the whiskers of the boxlot.

  Remark: Outliers may be errors, but also can hold important information. A prominent example is cardiac arrhythmia in a heart disease patient's electrocardiogram. In the data cleaning step, one would inspect the outliers in detail to decide what to do with them.

Your task is to visualize the percentage of the above listed special values of `myLCdata_dirty` per attribute. The result should look similar to the plots shown below. To solve the task, use `dplyr`, `tidyr` and `ggplot2`.


Percentage of special values in the character attributes of 'myLCdata'
Empty String
NA
n/a
Space
purpose
home_ownership
0%
5%
10%
15%
20%


Percentage of special values in the numerical attributes of 'myLCdata'
NA
NaN
Outliers
pub_rec
loan_amnt
int_rate
0%
5%
10%
15%
20%
25%

Instructions for Task 2

1. First, count the number of special values per attribute. To do that use `summarise()` from `dplyr`. Here are some hints:

   - To count the number of `NA`s in a column `col`, use `sum(is.na(col))`.

     Remark: Make it clear to yourself what is happening here in terms of vectorization.

   - To count the number of `NAN`s in a column `col`, use `sum(is.nan(col))`.

   - To count the number of *outliers* in a column `col` according to Tukey's rule, use `boxplot.stats(col)`:

     - `boxplot.stats(col)$stats[1]` holds the lower whisker of the boxplot. All values smaller than that are considered outliers.

     - `boxplot.stats(col)$stats[5]` holds the upper whisker of the boxplot. All values greater than that are considered outliers.

       Remark: `boxplot.stats(col)$out` *holds the outliers in* `col`*. You don't need the actual values here, since you only need to count them.*

   - To count *special strings*, use logical tests.

   - The following `dplyr` verbs shorten the task:

     - `across()` is a `dplyr` verb that allows to perform operations across multiple columns. It needs 2 arguments: The *columns* to apply a function to, and the *function* itself.

     - `everything()` is a helper function of `across()`. It specifies that the *function* should be applied to *all columns*.

     - `where(is.numeric)` is a helper function of `across()`. It specifies that the *function* should only be applied to the *numerical columns*.

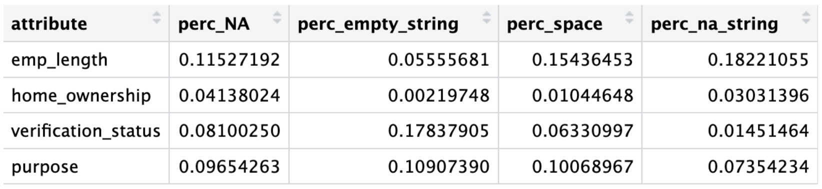
     - `where(is.character)` is a helper function of `across()`. It specifies that the *function* should only be applied to the *character columns* of the data frame.

     - You need to specify the *function* using *formula syntax*, i.e., the tilde operator (`~`). Inside your function specification, use the dot (`.`) to refer to the *current column* being operated on. Example:

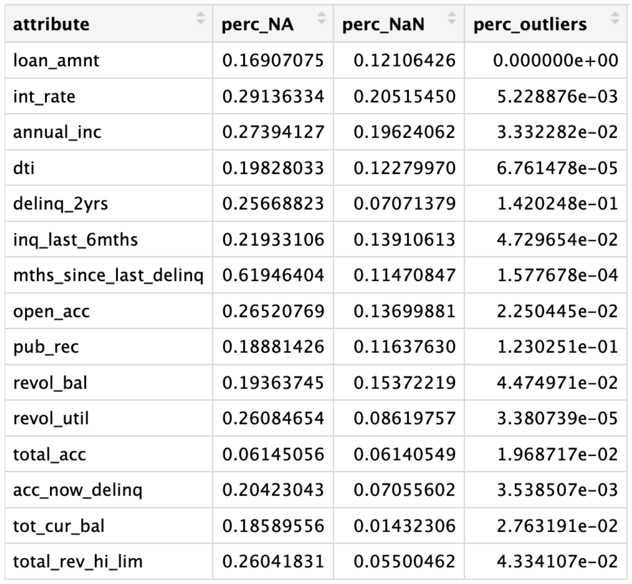
       ```r
       library(dplyr)

       # Transforming the factor variable to char (for demonstration purposes):
       iris <- iris %>%  mutate(Species = as.character(Species))
       ```

```
# Exemplifying across():
iris %>% summarize(across(everything(), ~sum(. == "setosa")))
iris %>% summarize(across(where(is.numeric), ~mean(.)))
iris %>% summarize(across(where(is.character), ~max(nchar(.))))
```

2. Once you successfully gathered the special value counts of all attributes, convert the counts in *relative counts* (by dividing by the number of rows of `myLCdata_dirty`).
3. Next gather the resulting values in *two data frames*: one for the numerical attributes, one for the character attributes. The resulting data frames should look similar to the tables shown below. Notice that `pivot_longer()` from `tidyr` may come in handy here.



attribute	perc_NA	perc_empty_string	perc_space	perc_na_string
emp_length	0.11527192	0.05555681	0.15436453	0.18221055
home_ownership	0.04138024	0.00219748	0.01044648	0.03031396
verification_status	0.08100250	0.17837905	0.06330997	0.01451464
purpose	0.09654263	0.10907390	0.10068967	0.07354234



attribute	perc_NA	perc_NaN	perc_outliers
loan_amnt	0.16907075	0.12106426	0.000000e+00
int_rate	0.29136334	0.20515450	5.228876e-03
annual_inc	0.27394127	0.19624062	3.332282e-02
dti	0.19828033	0.12279970	6.761478e-05
delinq_2yrs	0.25668823	0.07071379	1.420248e-01
inq_last_6mths	0.21933106	0.13910613	4.729654e-02
mths_since_last_delinq	0.61946404	0.11470847	1.577678e-04
open_acc	0.26520769	0.13699881	2.250445e-02
pub_rec	0.18881426	0.11637630	1.230251e-01
revol_bal	0.19363745	0.15372219	4.474971e-02
revol_util	0.26084654	0.08619757	3.380739e-05
total_acc	0.06145056	0.06140549	1.968717e-02
acc_now_delinq	0.20423043	0.07055602	3.538507e-03
tot_cur_bal	0.18589556	0.01432306	2.763191e-02
total_rev_hi_lim	0.26041831	0.05500462	4.334107e-02

4. Now make the two data frames *tidy*, so that you have an easier life applying `ggplot()`.
5. As a last step, plot the data using `geom_col()` from `ggplot2`. The following two layers may come in handy:
    - `coord_flip()`
    - `scale_y_continuous(labels = scales::percent)`

Deliverables for Project 2

As for all project parts, the deliverable consists of 2 files:

- a **pdf or html** report that you generated using Quarto,
- the corresponding working **.qmd** file that you used to render your report.

The **pdf report** should hold the solutions to the tasks listed above, including

- the code you used to solve the task,
- the code execution output, whenever it is relevant for your answer/solution.

The report must have a professional writing style, structure and layout. This includes

- title, author, date, and one subsection for each task;
- a brief intro and conclusion section so that your report is a "readable document",
- code comments in natural language in the sense of literate programming, so that the reader understands what is happening in the different steps of your code,
- numbered figures, captions describing the figures, and figures are discussed/mentioned in the text;
- warnings of code executions are suppressed, except they are relevant for your answer.

Please find the **upload link** for the deliverables **on Moodle**. The deadline is specified in the semester program.