Data management graded lab

Fabrice Rossi

! Instructions

This lab is dedicated to the analysis of a collection of data sets described below. Your work must be submitted as a github project and as a zip file. You must:

- create a github repository called **star_systems**. It should be public. You may create a private repository but you have then to invite me as a contributor;
- create a R project on your computer from the github project and make an initial commit with the classical R project configuration files;
- write all your answers in a quarto document **named after your last name**: commit the initial version of this document but dot not include the rendering of the document in the repository;
- each time you are satisfied by your answer to a question, commit the modifications;
- push the commits on a regular basis;
- at the end of the session, make a final commit with a push and then prepare to upload a zip file on moodle with at least:
  - the R project file (ending with `.Rproj`);
  - the data files (csv or rds format);
  - the quarto document;
  - the result of rendering your document to html.

All graphical representations must be done with ggplot2 and all calculations must be done with dplyr and tidyr.

Individual instructions

The instructions contained in this document are student specific. Your data sets are unique to you and parts of the instructions depend on the data sets. In particular, most names are unique. For instance, the name you have to use for the git repository is specific to you. Any failure to use those specific personal instructions will lead to an automatic fail of the assessment (0/20).

1 Main data set import

> Data loading
>
> Beware that you may have to use specific parameters of `vroom` or `read_csv` to properly load the data, for instance the `delim` parameter for `vroom` (or `read_delim` instead of `read_csv`). The decimal point may also be replaced by a comma. Missing data are represented by specific conventions given for each data set. Notice that fixing the content of the data set *after* reading it is generally much more complicated and error prone that using valid reading options. It is therefore strongly recommanded to avoid post loading fixes.
> You are expected to check the validity of the reading procedure: take care of any error or warning, and make sure you have loaded the expected number of variables (with proper data types).

You main data set is contained in the file "data_systems.csv".

Question 1

Create a data folder to store the files. Copy or move all the data files in this directory but commit **only** the main data file to your repository. You wil have the opportunity to commit the other files latter in the process.

Question 2

Include at the beginning of your quarto document a link to your github repository. Make sure that the title and the author of the document are set properly. You can choose the title as you wish.

The main data set contains descriptions of 1035 imaginary star systems from the BattleTech science fiction universe. Each star system is described by a collection of 8 variables:

- "`star_name`": the name of the star
- "`Distance`": distance in light years from the Sun
- "`STELLAR CLASS`": stellar type of the star (see below for details)
- "`STAR_NUMBER star_number`": a unique key for each star
- "`Temperature`": effective temperature of the star in Kelvin
- "`Radius`": radius of the star, expressed in solar radii
- "`mass`": mass of the star, expressed in solar masses
- "`RELATIVE LUMINOSITY`": luminosity of the star, relative to the Sun's luminosity

The file contains some missing data encoded by the following value: missing.

Question 3

Add code to your quarto file to load the main data set. To verify the validity of the loading process, check that the number of star systems is equal to the one specified above and that you get the correct number of variables. This should be done in a programmatic way (i.e. include a different message in the rendered document depending on the validity of the loading process). Verify also that all the numerical variables are recognised as numerical variables.

Question 4

Describe the data set with a properly formatted table (either using `kable` or via a markdown table) according to the following metrics:

- number of observations (i.e. star systems)
- number of variables
- number of missing values for each variable with missing values
- number of observations with at least one missing value

All values must be computed by the report, not entered as fixed quantities in the text.

2 Introductory analysis

Question 5

Represent graphically the mass of a star as a function of its temperature. Make sure to get rid of any warning or error!

Question 6

Report in a table the most extreme stars with respect to the numerical variables. This should include, for instance, the most and least massive stars. Do that variable by variable.

> **i** Commit and push
>
> Do not forget to commit your modifications after each question and to push the commits on a regular basis.

3 Factions

In the Battletech universe, star systems are controlled by factions. The factions are described in two data files:

- the "`ALLBTFACTIONS.csv`" file contains information about all Battletech faction;
- the "`Sytem+faction.csv`" file associate each star system with its controlling faction in year 2830.

Question 7

Commit these two data files into your repository. This is a good time to push your modifications to github.

The faction file contains the following variables:

- "`Faction_key`": a unique key for each faction
- "`faction name`": the name of the faction
- "`Founding Year`": founding year of the faction
- "`final year`": dissolution year of the faction

The controlling faction file contains the following variables:

- "`Faction_key`": a unique key for each faction
- "`STAR_NUMBER star_number`": a unique key for each star
- "`detailed structure`": local organisation of the faction (see below for details)

Question 8

Add code to load the two data files.

The faction list contains some fake" factions:

- **A** stands for abandoned star systems;
- **I** stands for independent systems (they are not controlled by any faction);
- **U** stands for systems that are not yet discovered during year 2830.

Question 9

Include in your document a table that gives the number of abandoned, independent and undiscovered star systems during year 2830.

In addition to these three specific factions, there might be some missing data in the files.

Question 10

Report in your document the number of star systems that cannot be found in the Sytem+faction.csv file.

Question 11

Represent graphically the number of star systems controlled by each faction that control at least one system. Make sure to use the full name of the factions.

Question 12

Represent graphically the distance to the Sun of star systems conditioned by the controlling faction, including only real factions (excluding therefore the three special factions described above).

Question 13

Include in your document a table that reports for each real faction the closest to the Sun star it controls.

Question 14

The faction description file contains foundation and dissolution years for each faction (so data may be missing). Verify that each faction that controls at least one star system in 2830 is active during this year (i.e. was founded before and was not yeat disolved).

4 Improved data representation

The stars are classified into different stellar types. A stellar type is made of three elements:

- a letter according to the *Harvard system*, which gives essentially the temperature *class* of the star, see the `star_types.csv` file;
- a digit from 0 to 9 that refines this temperature into a *magnitude* (0 for maximal temperature, 9 for minimal);
- a short text according to the *Yerkes spectral classification* which gives the *luminosity* class of the star (it is also related to its surface gravity), see the `Yerkes_luminosity_classes.csv` file.

Question 15

Add the stellar type description files to your repository.

In the star data set, the three components of the stellar type are merged in the `STELLAR CLASS` variable.

Question 16

Split the stellar type into three variables, one per component.

5 Advanced analysis

Question 17

Represent graphically the distribution of the stars in the data set according to their temperature class.

Question 18

Represent graphically the distribution of the stars in the data set according to their temperature class and to their magnitude simultaneously.

Question 19

Verify programmatically that the numerical characteristics of the stars (temperature, mass, radius and luminosity) are compatible with the ones given in the reference table. Ideally, this should be done without manual operation on the reference table, but this is somewhat difficult and manual imputation of test values is acceptable. It should be done in a CSV file that can be then be loaded by the main document.

Question 20

Display the distribution of the distance to the Sun for each class of stars (on a single) graphical representation.

Question 21

Display graphically the distribution controlling factions conditioned on the class of stars.

6 Advanced faction analysis

In the controlling faction file, the detailed structure describes the hierarchical organisation of the controlling faction that operates on each system, using commas to separate the levels. For instance a value of the form `"X,Y,Z"` corresponds to the faction `"X"` organized in a sub-faction `"Y"` itself subdivided in a lower category "`Z`".

Question 22

Extract from the data set all the unique strings that described such organisation.

Question 23

Process the content of the strings to produced a data frame with a first column that contains the faction id and a second column with the first level of subdivision.

Question 24

Represent graphically the number of first level subdivisions for each faction.