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 project called **galatic_empire**. It should be public. You may create a private project 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.
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 "PLANETS.csv".

Question 1

Create a data folder to store the files. Copy or move the main data set in this directory, and commit this main data set to your repository. Do that **only** for the main data set.

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 708 imaginary planets from a science fiction universe. Each planet is described by a collection of 8 variables:

- "`mass (earth)`": mass of the planet, expressed in mass of the Earth unit
- "`RELATIVE RADIUS`": radius of the planet, expressed in radius of the Earth unit
- "`density (earth)`": density of the planet, expressed in density of the Earth unit
- "`Planet`": name of the planet
- "`Planet_number`": a unique key for each planet
- "`STAR_IDX`": the key of the star of the planet
- "`Type_terrain`": types of terrain that can be found on the planet
- "`CLIMATE TYPE`": types of climate that can be found on the planet

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

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 planets 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. planets)
- 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.

Question 5

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

Question 6

Report in a table the most extreme planets with respect to the numerical variables. This should include, for instance, the densest and least dense planets.

Commit and push

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

2 Improved data representation

The terrain and climate variables are *multivalued*: a given planet can have multiple climates and multiple types of terrain. In the main data set, all values of e.g. terrain are kept in a single variable, Type_terrain. The different values are separated by the “,” symbol.

Question 7

Using `reframe` and `str_split`, extract the terrain types and the climate types into two different data frames containing only two columns: the planet identifier (Planet_number) and the quantity of interest (terrain type or climate type). This last column must contain only a single value: there will be multiple rows per planet in general.

Question 8

Represent graphically the distribution of the number of different climates per planet. Provide a similar representation for the terrain types.

Question 9

Represent graphically the distribution of the radius of the planets conditioned by the climates. In other words, we want to see the distribution of the radius of the planets that have some *arid* regions, some *humid* regions, etc. Notice that each planet will be accounted for multiple times owing to the multivalued nature of the terrain and climate variables. This is not a problem!

3 Star data

The planet data set is complemented by a star data contained in the file “data+stars.csv”. It contains descriptions of 551 imaginary stars from a science fiction universe. Each planet is described by a collection of 8 variables:

- “`star`”: the name of the star
- “`distance (lightyears)`”: distance in light years from the Sun
- “`TYPE OF STAR`”: stellar type of the star
- “`STAR_IDX`”: a unique key for each star
- “`star temperature`”: effective temperature in Kelvin
- “`RADIUS`”: radius of the star, expressed in solar radii
- “`Relative Mass`”: mass of the star, expressed in solar masses
- “`Star Luminosity`”: luminosity of the star, relative to the Sun’s luminosity

Question 10

Add the star data set to the data directory. Add code to load the data set and to verify the content.

Question 11

Assess the consistency of the two data sets (stars and planets) by printing the name of the planets whose star key (STAR_IDX) does not correspond to an actual star. Give also the number of stars with no associated planet in the planet data set.

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.

In the star data set, the three components are merged in the `TYPE OF STAR` variable.

Question 12

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

Question 13

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

Question 14

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

Question 15

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 16

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

4 Combined analysis

Question 17

Represent graphically the distribution of the number of planets per star

Question 18

Represent graphically the distribution of the number of planets per star conditioned by the class of the star.

Question 19

Compute for each star the average of the mass of its planets and plot this quantity as a function of the mass of the star.

Question 20

Computer for each star the total number of distinct terrain types seen on its surrounding planets.

Question 21

Represent graphically the distribution of the quantity computed in the previous question for the different classes of stars.

Question 22

Find a way evaluate the validity of the remark that "planet names are built from star names" and to represent the statistical property of this construction (recurrent features, dependency to other variables, etc.)