Brixius Engineering Consulting

GUIDELINE FOR FORMATTING TABULAR DATA

Version 1.0

www.bec.com

01.07.2026

Table of Contents

1. Purpose of this Document

This document provides rules for formatting tabular data, with the aim of making it easy to process via scripting languages such as JULIA, MATLAB, PYTHON or R. Especially when working with larger tables, manual data processing becomes unfeasible and prone to errors. While errors may also occur when scripting, they tend to be systematic and not slip-ups, which are harder to detect and therefore to correct. Data evaluation via scripting may cost more time initially, but re-usability of scripts, easy-to-change processing via code adjustments and transparency of the performed operations outweigh the disadvantages.

2. Formatting Text and Numbers

2.1 LANGUAGE

Every kind of text is to be written in English. This includes variable names, abbreviations etc.

2.2 ABBREVIATIONS

All abbreviations that are used in the file must be common abbreviations or must be explained in the file, see Chapter 3.2. “Common” in this context means “**common for a person who just graduated from a university from a master program in an associated field of study**”. Abbreviations that might be common inside a team or even the company must still be explained.

2.3 NUMBERS

Dots are to be used as a decimal separator. **Do not** use any form of separator for decimal powers like placing a comma after thousand, million etc. This means **do not** use the German, Spanish,… way of writing numbers but the International / American way. E.g.: 1000.01. How very big or small numbers can efficiently be stored, is explained in chapter 3.5

3. Data Type

Tabular data is stored as a CSV with a comma separated cells and a semicolon used for line breaks. UTF-8 is the character standard to be used. Keep in mind that a CSV is not meant to store 3 or even higher dimensional data. Each 3D data set must be divided in regular 2D tables, stored in a separate CSV file each.

Alternatively, if not avoidable, XLSX might be used to store multiple tables as separate sheets in a single file.

SQL databases are to be used if the complexity of the data structure demands it.

4. Table Format

4.1 THE FIRST ROW

The first row and **only** the first row of a table contains all information intended for human readers. To indicate that it does not contain any data, the first three characters in the first cell (field 1-1) shall have 3 “#” to allow the programmer to easily filter out said row. For example, “###Title: Example_Table”. The first row **must** include the following information:

- **Title:** A meaningful title for the file, which is **not** the actual file name. E.g. the file name might be “20250204_BP_GL” but in the first field it will say: “Title: Birthday Party Guest List”
- **Date of Creation:** This makes it easy to identify the time during which the file was worked on when the file is accessed later.
- **Authors:** This is not for determining potential copyright, but to give a reader a contact to reach out to, in case of troubleshooting. So not only the actual creator of a file should be named, but everyone who was a major contributor to the project. The convention shall be: Firstname Lastname (XXX) where XXX is the team identifier. For example: John Doe (USA). For external personal, enter the company name, for example: Steve Jobs (Apple).
- **List and Explanation of ALL Non-Common Abbreviations:** Self-explanatory, see chapter 2.2.

Additionally, the first row **may** include:

- **Sources:** If a sheet is generated based on data stored in another sheet, the location and file name are to be mentioned here.

- **Description:** Can contain anything not covered by the other categories. Especially, since the reference feature from EXCEL will be lost, the author shall describe the formulars and calculation steps.
- **Change Log:** Changes made to the file can be documented with the author, date and contents of the changes.
- **File Version:** While the name of a file will usually indicate its version, it can make sense to store said information also inside the file. However, one shall pay attention to keeping the version-labelling consistent.

4.2 FIRST COLUMN, SECOND AND THIRD ROW

The second row and the rest of the first column contain the actual data. Because the second and third column store information on the **unit** and the **decimal power** of a column, the first row to contain values is the 4th, see below and Chapter 3.4 and 3.5. The second row shall contain the headers. Ideally, they are one-word expressions of what the data inside said column represents, e.g. “temperature”. Notice that the first letter must be a lower case. **ATTENTION:** Keep in mind that when scripting, one will filter the columns by string comparison, hence spacebar characters or other characters known to cause problems for computers **must** be avoided. For that reason, DO NOT use spacebars as word separators, but underscores, e.g., “Average_inflow_per_hour”. Make sure that you use no protected spacebars, leading or tailing spacebars, special characters etc.

By convention all headers shall be the singular form of the word, so “unit” instead of “units” or “object” instead of “objects”. Only when one cell contains an array of data, e.g., [AT+DE+CH] so a list of countries to consider, the header shall be a plural so “Countries_to_consider”.

The first column must contain the name object the data in the corresponding row belongs to, e.g. “Germany”. Its header should be something like “Country”, “Object”, “Power_plant”, “RDF_ID”, “Name”,… The second row must store the units of each column, e.g. “°C”, “MW/h”. It can also be used to store the data type instead, e.g., “float”, “bool”, “int”, “str”, “pf.DataObject”, … The third row is used for conveying the decimal power, see chapter 3.5. As row 2 and 3 contain special information that is not data, their row-headers are written in all caps similar to the convention in SQL.

country	averge_temperature	population	…
UNIT	°C	int	…
DECIMALPOWER	0	6	…
austria	21	83	…
germany	18	21	…

4.3 SPECIFYING UNITS

To specify the units of the data in each column, the second column must be used. The column specifier (field 1-2) shall be “UNIT”. The International System of Units (SI system) and its official unit symbols are to be used. For example, metric tons have the symbol “t” and not “tons”, a second is denoted by “s” and not “sec” etc. The same goes for letters abbreviating decimal powers like n, u, m, k, M, G etc. Do **not** put rectangular brackets around a unit. *To be decided: using u instead of µ and/or Ohm instead of Ω*. **If no unit is specified a “-“ must be entered in the respective field!**

4.4 SPECIFYING DECIMAL POWER

To avoid numeric instability, big or small numbers should be stored with a separate decimal power. The third row with the column identifier (field 1-3) is set to “DECIMALPOWER”. Every field of this row will contain the exponent of the imaginary 10 to be multiplied with the rest of the column. Keep in mind that by mathematical convention every number to the power of 0 is 1, so enter 0 for each column that is not to be changed. E.g. “10000” becomes “3” and “10” indicating the two cells used. **ATTENTION:** As empty rows are prohibited, as described in the next section, one always needs to fill the second row when specifying decimal powers. **Even if no decimal power is specified for column X, the field X-4 must be filled with a 0!**

4.5 KEY COLUMN

Imitating the concept of a KEY_COLUMN from SQL, the first column must always have unique, meaning pairwise different, entries.

4.6 ONE DATATYPE PER COLUMN

Frameworks like “pandas” can have problems when reading tables that store different datatypes in the same column. Hence, except from the “UNIT” and “DECIMALPOWER” row, each column shall only contain one type of data, which can be indicated in the “UNIT” row, see chapter 3.3.

4.7 SHEET NAMES

Keep in mind that Excel can only process sheet names with 31 characters or less.

4.8 NO EMPTY CELLS, COLUMNS OR ROWS

No columns, rows or cells are allowed to contain no data. Instead, to leave a cell blank on purpose, use “-“ or “MISSINGDATA”. Complete columns/rows free of data are under no circumstances allowed as there is no reason for such a thing except for better readability with the human eye which is not the purpose the tabular data file format described here. With the presented format Julia, MATLAB, Python etc. can be used for data visualisation including compact, good-looking tables to include in a PowerPoint, Word-Document or PDF.

5 Responsibility for Upholding the Format

5.1 RESPONSIBILITY ACCORDING TO THE 'POINT-OF-ENTRY' PRINCIPLE

The responsibility for implementing the format described in this document shall follow the "Point of Entry" principle. This means that:

a) A person inside the company who creates a tabular data file must make sure that the rules described above are implemented.
b) The first person inside the company to receive a tabular data file from an external company or organisation, must immediately check if the file complies with the rules described above.

A person inside the company receiving a tabular data file from another employee can demand reworking of said file till it complies with this standard. This also applies when the file was created externally and there was an intermediate recipient inside the company. If multiple persons are involved in the creation or acceptance of a file, they bear responsibility equally. Reworking can be demanded from each individually. If someone is on vacation, sick leave etc. the team leader of said person bears responsibility for the reworking or must delegate this task to someone.

If tabular data is received by outside entities against which one has no bargaining power, it still falls into the responsibility of the person receiving the file to rework it if necessary. Their team leader is to provide the resources for that, for example by hiring a working student to write scripts for file conversion or considering the persons workload in their planning.

5.2 FILING AN INTERNAL COMPLAINT ABOUT FORMAT NON-CONFORMITY

While the rules for filing a complaint at acceptance of a file from an external company follow the basics of German or international law, especially § 377 HGB, and most importantly the contract with said company, a complaint regarding a file exchange between two persons inside the company must be filed within one week of receiving the file. The complaint is to be filed via the company e-mail address and the supervisor of the recipient of the complaint shall be set in CC. This week is prolonged by days of sick leave, business trips etc. of the file recipient if they happen on a workday.

Once the week passes and no complaint was filed, the recipient of the file is threated as its creator. The rules set forth in chapter 4.1 apply to him or her from that point on.