

Tables and spreadsheets

Tips for organizing and managing tabular data

LiRI Data Stewards

Contents

| | |
|---|---|
| Recommendations for organizing tables or spreadsheets | 1 |
| 1. Make tables machine-readable | 1 |
| 2. Use interoperable file formats | 2 |
| 3. Include a codebook or data dictionary | 2 |
| 4. Keep them ‘tidy’ | 2 |
| 5. Keep raw data | 2 |
| Examples | 2 |
| Frequent actions with tables | 4 |
| Data validation | 4 |
| Combining tables | 4 |
| Reshaping tables: wide and long formats | 5 |
| References | 6 |

Recommendations for organizing tables or spreadsheets

Almost all research projects use some form of *tabular data*. They may contain descriptive information, experimental task performance, filenames of image or scan files, recording parameters, measurements or any other information for analysis. Further, *metadata tables* are also essential in a project as a form of structured metadata with essential information to interpret the data.

Guidelines for increasing the machine-readability of spreadsheets are given in (Perkel 2022) and (Broman and Woo 2018). We recommend these guidelines as they are concise and accessible reads. They highlight the need for consistency in formatting and machine-readability to enable automation of data validation, preprocessing, synthesis, and analysis of tabular data or metadata. Additional online material is provided in the chapter [data organisation in spreadsheets](#) of the e-book for reproducible research from ‘The Turing way’(The Turing Way Community 2022)

1. Make tables machine-readable

Tables MUST be **structured** and **formatted** to be **machine-readable**. For example: do not encode any information with *formatting* (e.g., in color-coding schemes), create additional columns that can be used as filter, do not merge cells and do not use empty cells (see [examples](#))

2. Use interoperable file formats

Publish and share them in an **interoperable** format, that is, a format compatible with most tools and programming languages. **Comma-separated values (.CSV)** is recommended. Working on spreadsheet software like MS Excel or Google Spreadsheets can be very convenient and in some cases it may be preferred for collaborative editing of tabular data. When using spreadsheet software, researchers should ensure that the table can be exported to more interoperable formats like CSV and read programmatically without losing or distorting the information.

3. Include a codebook or data dictionary

The tables **MUST** be shared/published together with a **codebook** or **data dictionary** that explains the meaning of the variables, abbreviations, naming conventions and units. This can be in documented a table format (e.g., variable names as rows in the first column and description in a second column) or in text format (e.g., in a README file).

4. Keep them 'tidy'

- Include a **single piece of information in a cell** (e.g., ,instead of a cell in the column weight 20_kg better make an entry 20 in a column weight_kg)
- Keep formatting and naming **consistent** and follow broadly established **standards** when possible (e.g., write dates following ISO 8601 standard YYYY-MM-DD (this should be described in the data dictionary). There can be also standards referring to file or variable naming, units or number formats among others.

5. Keep raw data

Researchers may use functions in spreadsheet software or manual transformations to change the source data into more reusable format. If these transformations are necessary, keep track of them (document in a README file and/or share the script if there is one available) and make sure the results are read correctly when the files are exported into more interoperable formats like .CSV.

Examples

A poorly formatted table

The example in Table 1 illustrates errors that are often found in tables, specially when they are edited by multiple researchers . The structure is not adequate for data processing or analysis and there are inconsistencies in the cell values (e.g., subject labels) and formats (e.g., dates, decimals in weight). White spaces in variable names led to automatically adding a dot '.' when reading the table with R code. Color code seems to highlight images with artifacts.

A better formatted version

The example in Table 2 can be easily read using code, **without ambiguities**. It is accompanied by a codebook (Table 3) describing the variables.

Table 1: A poorly formatted table

| Scanner | Subject ID | Scan_Date | Modality | Sex | Age (years) | Weight_kg | Experimental Group | Notes |
|---------|------------|-------------|----------|--------|-------------|-----------|--------------------|-----------------|
| Siemens | | | | | | | | |
| | S01 | 01/01/2023 | MRI | M | 25 | 70 | Ctrl | |
| | S002 | 01/02/2023 | fMRI | F | 26 | 72.04 | Exp | Motion artifact |
| | S005 | Feb 5, 2023 | MRI | M | | NA | Exp | Blurring |
| | S008 | 08/01/2023 | T1 | M | 60 | 90 | Exp | NA |
| Philips | | | | | | | | |
| | sub003 | 03/01/2023 | DTI | Male | 27 | 75 | Exp | High noise |
| | S006 | 06/01/2023 | fMRI | F | 30 | 82kg | Control | |
| | S009 | | MRI | F | 33 | | Control | |
| GE | | | | | | | | |
| | S004 | 04/01/2023 | T1 | Female | 28 | 80 | Control | Good scan |
| | S007 | 07/01/2023 | DTI | NA | 31 | 85 | Exp | Strong signal |
| | S010 | 10/01/2023 | fMRI | Male | 34 | N/A | Exp | |

Table 2: Example of a table formatted for machine-reability

| Subject_ID | Scan_Date | Modality | Sex | Age_years | Weight_kg | Exp_Group | Scanner_Model | Discard | Notes |
|------------|------------|----------|-----|-----------|-----------|-----------|---------------|---------|-----------------|
| S001 | 01/01/2023 | MRI | M | 25 | 70 | Control | Siemens | FALSE | None |
| S002 | 02/01/2023 | fMRI | F | 26 | 72.04 | Exp | Siemens | TRUE | Motion artifact |
| S003 | 03/01/2023 | DTI | M | 27 | 75 | Exp | Philips | TRUE | High noise |
| S004 | 04/01/2023 | T1 | F | 28 | 80 | Control | GE | FALSE | None |
| S005 | 05/01/2023 | MRI | M | 29 | 82 | Exp | Siemens | FALSE | Good scan |
| S006 | 06/01/2023 | fMRI | F | 30 | 85 | Control | Philips | TRUE | Blurring |
| S007 | 07/01/2023 | DTI | NA | 31 | 88 | Exp | GE | FALSE | None |
| S008 | 08/01/2023 | T1 | F | 32 | 90 | Exp | Siemens | FALSE | Strong signal |
| S009 | NA | MRI | M | 33 | 92 | Control | Philips | FALSE | None |
| S010 | 10/01/2023 | fMRI | F | 34 | 95 | Exp | GE | FALSE | None |

Table 3: Example of a codebook

| Variable | Description |
|---------------|--|
| Subject_ID | Unique subject identifier. Possible values: S001, S002, S010, etc. |
| Scan_Date | Date when the scan was performed. Possible values: YYYY-MM-DD. |
| Modality | Imaging modality used. Possible values: MRI, fMRI, DTI, or T1. |
| Sex | Biological sex of the subject. Possible values: M (male), F (female). |
| Age_years | Age of the subject in years. Possible values: 25, 30, 34. |
| Weight_kg | Subject's weight in kilograms. Possible values: e.g., 10.00. |
| Exp_Group | Experimental group assignment. Possible values: Control, Exp (experimental). |
| Scanner_Model | MRI scanner brand. Possible values: Siemens, Philips, or GE Discovery. |
| Discard | Discard images with artifacts TRUE or FALSE. |
| Notes | Additional notes on scan quality or issues. Possible values: Free text. |

Frequent actions with tables

Researchers often need to do some minor transformations or operations with tabular data. The main recommendations are:

- Use **code** as it will be a more reproducible and transparent process than doing it manually. With a version control system, tracking changes Irrespective of the programming language, the code for these operations is usually easy to implement for users without advance programming skills. In R language `dplyr` and `tidyr` are two useful packages for handling and manipulating tables.
- Keep the code and preferably keep track of changes in the code and the tables: **version control**.

Data validation

There are a few initial checks and data validation actions that be conducted before archiving, publishing, sharing or conducting any analysis on tabular data to ensure they do not contain errors. Some frequent checks are:

- Variable types (e.g., numeric or character) and variable names (do they conform with the documented naming convention and codebook?)
- Missing values and complete cases (e.g., are they as expected?)
- Number and/or date formats (e.g., are they consistent?)
- Unique values (e.g., is sex consistently described with the same label?)
- Implausible values (e.g., negative age value, percentiles above 100)
- The ‘tail’ of the data (e.g., researchers may have calculated the mean of a column in the last row)

Some useful functions in R programming language are:

- `str()` (shows the structure of the data, including variable types and lengths)
- `unique()` (for unique values)
- `is.na()` (for missing values)
- `summary()` (summary statistics).

There are also dedicated packages for data validation. For a more detailed guideline, together with the code (in R) to perform different data validation actions we recommend the cookbook from the [validate package for R](#). For python users, a recent package for data validation is the [pointblank python package](#)

For a more conceptual overview, without code, we recommend the summary tables in the *framework for initial data analysis* (Huebner et al. 2018) used in the working group 3 from the [STRATOS initiative](#) (STRengthening Analytical Thinking for Observational Studies).

Combining tables

Researchers often need to combine information from two tables. A common scenario is to have a table with as many rows as subjects (e.g., subject characteristics) and another table with multiple rows per subject (e.g., indicating image files, slices, scans, etc.). If we want to merge these two tables we need a **key variable** that is consistent in both tables, for example, a `subjectID` variable uniquely identifying each subject. Here it is essential that each subject is labeled *consistently* and that the key variables have identical names in each table.

R code snippet to join two tables:

```
library(dplyr)

# Read csv tables tables
tbl_subjects <- read.csv('dummy_table_OK.csv')
tbl_files <- read.csv('dummy_table_filenames.csv')

# Join by variable "subject_ID"
tbl_joined <- dplyr::full_join(x=tbl_subjects,
                              y = tbl_files,
                              by=join_by("Subject_ID"))
```

Table 4: Tables file information

| Subject_ID | Files | Session |
|------------|---------------|---------|
| S001 | filename1.ext | 1 |
| S002 | filename1.ext | 1 |
| S003 | filename1.ext | 1 |
| S003 | filename2.ext | 1 |
| S003 | filename3.ext | 2 |

Table 5: Combined table after adding subject information

| Subject_ID | Scan_Date | Modality | Sex | Age_years | Weight_kg | Exp_Group | Scanner_Model | Notes | Files | Session |
|------------|------------|----------|-----|-----------|-----------|-----------|---------------|-----------------|---------------|---------|
| S001 | 01/01/2023 | MRI | M | 25 | 70.00 | Control | Siemens | None | filename1.ext | 1 |
| S002 | 02/01/2023 | fMRI | F | 26 | 72.04 | Exp | Siemens | Motion artifact | filename1.ext | 1 |
| S003 | 03/01/2023 | DTI | M | 27 | 75.00 | Exp | Philips | High noise | filename1.ext | 1 |
| S003 | 03/01/2023 | DTI | M | 27 | 75.00 | Exp | Philips | High noise | filename2.ext | 1 |
| S003 | 03/01/2023 | DTI | M | 27 | 75.00 | Exp | Philips | High noise | filename3.ext | 2 |

In more complex settings, it is possible to specify the expected relationship between two tables (e.g., if each row from table x is expected to have a match in table y, etc.) see [mutate-joins](#).

Reshaping tables: wide and long formats

We can distinguish two basic data formats:

- **Wide format:** values that do not repeat are usually in the first column and every measure that varies occupies a set of columns. E.g., One row per subject, columns specifying features of the subjects.
- **Long format:** multiple records for each individual. Some variables may not vary are identical in each record, whereas other variables vary across the records. E.g., Multiple rows per subject, listing filenames of images associated with each.

Research projects often need to combine both types of tables, and some statistical analysis or visualizations may required one specific table format. Researchers often need to ‘reshape’ the tables.

NOTE: these transformations can become quite complex when manipulating large tables with many variables involved. **Always** check the output of these operations (e.g., runnig validation or descriptive summaries) to make sure that no information was missplaced or data loss.

R code to convert from wide to long format:

```
library(tidyr)
library(dplyr)

tbl_long <- tbl_wide %>%
  pivot_longer(cols = c(Test_A, Test_B), names_to = "Test", values_to = "Value")
```

Table 7: Table converted to long format

Table 6: A table in wide format

| Subject_ID | Test_A | Test_B |
|------------|--------|--------|
| S001 | 5.4 | 3.2 |
| S002 | 6.1 | 5.8 |
| S003 | 7.3 | 6.5 |
| S004 | 4.8 | 4.1 |

| Subject_ID | Test | Value |
|------------|--------|-------|
| S001 | Test_A | 5.4 |
| S001 | Test_B | 3.2 |
| S002 | Test_A | 6.1 |
| S002 | Test_B | 5.8 |
| S003 | Test_A | 7.3 |
| S003 | Test_B | 6.5 |
| S004 | Test_A | 4.8 |
| S004 | Test_B | 4.1 |

R code to convert from long to wide format

```
library(tidyr)
library(dplyr)

tbl_wide_again <- tbl_long %>%
  pivot_wider(names_from = Test, values_from = Value)
```

References

- Broman, Karl W., and Kara H. Woo. 2018. "Data Organization in Spreadsheets." *The American Statistician* 72 (1): 2–10. <https://doi.org/10.1080/00031305.2017.1375989>.
- Huebner, Marianne, Saskia Le Cessie, Carsten O. Schmidt, and Werner Vach. 2018. "A Contemporary Conceptual Framework for Initial Data Analysis." *Observational Studies* 4 (1): 171–92. <https://doi.org/10.1353/obs.2018.0014>.
- Perkel, Jeffrey M. 2022. "Six Tips for Better Spreadsheets." *Nature* 608 (7921): 229–30. <https://doi.org/10.1038/d41586-022-02076-1>.
- The Turing Way Community. 2022. *The Turing Way: A Handbook for Reproducible, Ethical and Collaborative Research (1.0.2)*. Zenodo. <https://doi.org/10.5281/ZENODO.3233853>.