2. Decoding Trial Data, Scoring Tasks, and Widening

*Thomas Pronk*

[*tpronk@uva.nl*](mailto:tpronk@uva.nl)

*Version 1.7, 2017-02-15*

In Tutorial 1 you installed an empty SANDRA Analysis Framework (SAF); in the folder you specified, an interim, script, and original folder were created, and a FileIO object was setup for those folders. However, the SAF did not contain any data nor analysis scripts. In this tutorial you’ll install a SAF that contains pre-made scripts and example data for common scenarios, such as:

* **t.1 - Decoding task data encoded in a LOTUS results file into trial data, metadata, and optionally screen and slideshow data.**
* **t.2 - Calculating scores from trial data**
* **t.3 - Merging scores together into a single wide file**

# ****Part 1. Setup the pre-provided SANDRA Framework****

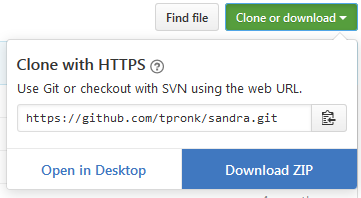
In this part you’ll download the pre-made SAF.

Figure . Screenshot of the SANDRA repository, showing where the "Download ZIP" button can be found.

1. Download the whole SANDRA repository in a ZIP via this URL: <https://github.com/tpronk/SANDRA>. Click the green “Clone or download” button in the top right corner of the screen, then click “Download ZIP” (Figure 2).
2. Unzip the file. The folder *SAFs/templates* contains the pre-made SAF.
3. Move the contents of *SAFs/templates* to a comfortable location on your computer, then run “Install SANDRA (from GitHub).R” to set up the framework for that location. This is similar to what you did in the Tutorial 1, Step 2, except that in this case the framework already contains pre-made data and scripts.

If everything went well, you’ll see some messages saying that various directories already exist, and finally the message:

sandra::FrameworkFileIO. Succesfully constructed FileIO

# ****Part 2. Test the processing script with the example data****

The *scripts* folder contains a collection of scripts for processing your data. In general, you’ll execute a t.1, t.2, and t.3 script in order:

* **t.1a decodes task data embedded in a LOTUS results file into separate output tables.** Depending on whether the task data is in JASMIN1, JASMIN2, or SPRIF1 format, different output tables are produced (for trials in each of the tasks, screen data, or slideshow data). Additionally, a *metadata* or *task\_data* table is produced which contains general information about each participation.
* **t.2.a calculates scores from trial data.** There is the option to calculate d-scores and aggregations such as “difference-of-medians/means for correct responses”.
* **t.2.b calculates reliabilities of scores.** To assess the reliability of sets of d-scores or difference-medians/means, one can calculate a repeated randomized split-halve correlation.
* **t.3 joins scores across tasks and sessions into one file.** The joined file is in wide format, with one row per participant and columns denoting scores, tasks, and sessions in the following format: “score.task1.pretest”, “resp\_1\_n.task2.posttest”, etc.

By default, the scripts are set up to process demo datasets, with t.2 and t.3 setup for further processing of the JASMIN2 demo dataset. However, t.2 and t.3 can be used to process JASMIN1 data as well. Run t.1.a, t.1.b t.1c, t.2.a, t.2.b, and t.3, in order to test whether the framework is set up correctly. If no errors occurred, then a set of data files should be produced in the *interim* folder with names such as “jasmin1\_data.vpt.csv”, “sprif1\_data.metadata.csv”, and “jasmin2\_data. aat.medians.csv”. Check “date modified” to find out whether these files were actually just created.

# ****Part**** 3. Download a LOTUS Results File

Time to download your own task data into a results file. If you are downloading JASMIN1 or JASMIN2 data, follow the instructions in 3.a; if you are downloading SPRIF1 data, follow the instructions in 3.b.

## ****Part**** 3.a. JASMIN1 or JASMIN2 Data

1. Login to LOTUS.
2. Go to *Your project 🡪 View participants 🡪 Participants 🡪 Results*.
3. Click *Export* in the top right of the screen. At the export screen:
   1. For Participant Parameters, make sure to select any participant parameters you are interested in (such as *username* or *age)*.
   2. For Result information:
      1. Be sure that **Session**, **Run ID**, and **Name** are ticked.
      2. At **Separator**, remove the ‘;’ and enter ‘\t’ instead.
      3. At **Escaped By**, remove the ‘\’ and just leave that field empty.
      4. At **Enclosed By**, remove the ‘”’ and just leave that field empty.
      5. Click OK.
4. Unzip the downloaded file and put it in the *original* folder.
5. Be sure to give your unzipped results file an informative name, such as *pretest\_alcohol.csv*. Note that by default, Windows Explorer does not display the *extension* of a filename (the part after the dot, which is *csv* in the example. In that case, you don’t need to enter the extension when renaming the file; just name it *pretest\_alcohol*. However, when adjusting *fileSource* in the processing script, this extension needs to be specified. To summarize, in Explorer your file could be displayed as *pretest\_alcohol*, but in the scripts you need to refer to this file as *pretest\_alcohol.csv*.

## ****Part**** 3.b. SPRIF1 Data

1. Login to LOTUS.
2. Go to *Your project 🡪 View participants 🡪 Participants 🡪 Results*.
3. At the *Result name* drop-down menu at top of the screen, select *trial*
4. Click *Export* in the top right of the screen. At the export screen:
   1. Make sure to select any participant parameters you are interested in (such as *username* or *age)*.
   2. Click OK.
5. Unzip the downloaded file and put it in the *original* folder.
6. Be sure to give your unzipped results file an informative name, such as *pretest\_alcohol.csv*. Note that by default, Windows Explorer does not display the *extension* of a filename (the part after the dot, which is *csv* in the example. In that case, you don’t need to enter the extension when renaming the file; just name it *pretest\_alcohol*. However, when adjusting *fileSource* in the processing script, this extension needs to be specified. To summarize, in Explorer your file could be displayed as *pretest\_alcohol*, but in the scripts you need to refer to this file as *pretest\_alcohol.csv*.

# ****Part 4. The structure of JASMIN1, JASMIN2, and SPRIF data****

The t.1 scripts produce different types of files, depending on whether they are processing JASMIN1, JASMIN2 or SPRIF encoded files. The sections below contain more information about the files produced in processing each of these three formats.

## ****Part 4.a.** **The structure of**** JASMIN1 data

JASMIN1 encoded data is decoded this into separate files containing trial data of each task and one **metadata** file containing general information. A set\_id variable identifies participations across tables. Each task data file can be joined with the metadata file on set\_id, wherein each value of set\_id that is present in metadata is present in one of the task data files. The document **TP - CBM Tasks JASMIN1 Configuration.docx** provides more information about JASMIN1 trial data. You can find this document in ADAPT Shared/Documentation/RT Tasks. Table 1 contains an explanations of columns in the metadata file.

**Table 1.** Explanation of JASMIN1 metadata variables

|  |  |
| --- | --- |
| **Folder** | **Explanation** |
| taskType | Identifies the type of task |
| task\_name | Identifies task variant |
| run\_from | Earliest RunID in results file with data for this participation |
| run\_to | Latest RunID results file with data for this participation |
| lotus\_says | Tells you how the task ended. There is only trialdata produced if the task completed successfully. Possible values:   * **task\_done**. Task completed successfully * **task\_start**. Task was restarted before it completed * **task\_error**. Task reported that an error occurred |
| event\_count | The number of events logged during the task, may vary depending on participant behavior. |
| sequence\_report | If this variable is not empty, then the trial data of this participation are likely not useful. Possible values:   * **inconsistent.** Events with the same sequence number but different data. This can happen if multiple participants are taking part in a task using the same LOTUS account. * **missing.** Sequence numbers missing. This indicates that certain task data was not received by LOTUS. * **negtime.** Client time decreased with successive sequence numbers. This indicates that the participants’ computer had an unreliable clock. * **invalid.** Task produced invalid (or no) data. This can happen when the participant did not complete any trials. |

## ****Part 4.b.** **The structure of**** JASMIN2 data

JASMIN2 encoded data is decoded this into separate files for trial data of each type of task; **slideshow** and **screen** data; and **task\_start**, which contains general information. A **participation\_id** variable identifies participations across tables. Each task data file can be joined with the slideshow, screen, and task\_start files on participation\_id, wherein each value of participation\_id that is present in task\_start, slideshow, and screen is present in one of the task data files. The document **JASMIN2 Data Manual.docx** provides more information about JASMIN1 trial data. You can find this document in ADAPT Shared/Documentation/RT Tasks. Table 2 contains an explanations of columns in the task\_start file.

**Table 2.** Explanation of JASMIN2 task\_start variables

|  |  |
| --- | --- |
| **Folder** | **Explanation** |
| taskType | Identifies the type of task |
| task\_name | Identifies task variant |
| sequence\_report | If this variable is not empty, then the trial data of this participation are likely not useful. Possible values:   * **inconsistent.** Events with the same sequence number but different data. This can happen if multiple participants are taking part in a task using the same LOTUS account. * **missing.** Sequence numbers missing. This indicates that certain task data was not received by LOTUS. * **negtime.** Client time decreased with successive sequence numbers. This indicates that the participants’ computer had an unreliable clock. * **invalid.** Task produced invalid (or no) data. This can happen when the participant did not complete any trials. |
| userAgent | Contains information about the browser used by the participant in order to take part. For more information, see: <https://en.wikipedia.org/wiki/User_agent#Use_in_HTTP> |

## ****Part 4.c.** **The structure of**** SPRIF1 data

SPRIF1 encoded data is decoded this into separate files containing trial data of each task and one **metadata** file containing general information. A set\_id variable identifies participations across tables. Each task data file can be joined with the metadata file on set\_id, wherein each value of set\_id that is present in metadata is present in one of the task data files. Table 3 contains an explanations of columns in the metadata file.

**Table 3.** Explanation of SPRIF metadata variables

|  |  |
| --- | --- |
| **Folder** | **Explanation** |
| sequence\_report | If this variable is not empty, then the trial data of this participation are likely not useful. Possible values:   * **incomplete.** The number of values in SPRIF data was not a whole multiple of the number of elements in (sprifVars + 1). This indicates that trials in the task produced different numbers of variables. * **norowsep.** One of the rows of SPRIF data did not start with a SPRIF row separator. This indicates that trials in the task produced different numbers of variables. * **rowsepinvar.** One of the SPRIF variables contained a SPRIF row separator. This indicates that trials in the task produced different numbers of variables. * **invalid.** Task produced invalid (or no) data. This can happen when the participant did not complete any trials. |
| taskName | Identifies the type of task administered this participation. |

# ****Part 5. Process your own data****

Now let’s give a go! Configure the right t.1, t.2, and t.3 scripts and process your data. Note that with big datasets, t.1 may take a while.

# ****Acknowledgements****

My gratitude goes to many researchers that provided help. In particular, I would like to thank Joeri van Wijngaarden for co-developing tutorial materials, and Marilisa Boffo and Ruby Smits for testing tutorials and providing feedback on usability.