TreeRing: A GameSafe Parser for z-Tree*

Ming Jiang† Jingchao Li‡

February 8, 2019

Abstract

This article presents a software that parses the server-client communication log files (called *GameSafe*) generated by the experimental economics software z-Tree (Fischbacher, 2007). The program parses the file and writes to a spreadsheet that includes the experimental parameters and a complete timeline of subjects' decisions. It can be used to recover data from an experimental session in case of a computer crash, as well as to reconstruct experimental datasets any point in time during the experiment even if variables are overwritten later.

Keywords: experimental economics, software, z-Tree.
JEL Codes: A20, C88, C90.

1 Introduction

Many laboratory experiments in economics are conducted in dedicated computer labs on university campuses. Fischbacher's z-Tree (Fischbacher, 1999, 2007) has become one of the most popular platforms of choice for designing and conducting those computerized experiments. As of this writing, it has been used for at least 8194 times based on the citation count on Google Scholar.

*The software application described here is free of charge and licensed under the MIT open source license. Any use of the software, whether as a whole or in parts, implies the acceptance of the license agreement. The application source code and test data are available for download at https://github.com/mjiangsjtu/treering. Jiang thanks the National Science Foundation of China (Grant 71703038) and Shanghai International Institute of Finance and Economics for research support. Li thanks the Fundamental Research Funds for the Central Universities and Shanghai Municipal Education Commission Research Project (Grant 2017-01-07-00-02-E00008) for research support.

† Antai College of Economics and Management, Shanghai Jiao Tong University, mjiang@sjtu.edu.cn.
‡ School of Business, East China University of Science and Technology, jingchaoli@ecust.edu.cn.

z-Tree is designed as a client-server application (Fischbacher, 2007). For every experiment, the communications between z-Tree server and clients are stored in a binary file called *GameSafe* (`.gsf`). z-Tree provides a built-in functionality to export this binary-format file into a plain-text file. Although now the file is technically human-readable, it is "gigantic", in the author's own word (Fischbacher, 1999), and useful information is not easily extracted.

In many cases, however, understanding the content of a gsf file (i.e., what actually went on during an experiment), and potentially extracting useful information from it, can be very helpful. Therefore, we design this software, named `TreeRing`, implemented in Python, to parse the GameSafe file and convert it into a more useful and human-friendly form, and to recreate usable data tables at any point in time during the experiment. We believe this program will serve at least the following three purposes.

First, this software can recover experimental data in case of a server crash. Although z-Tree is designed to be robust to computer failures during an experiment (e.g. experiments can continue after multiple client computers crash), there are still some cases in which researchers may suffer data loss. For example, z-Tree data file is created only after every subject has finished the last stage of the experiment. If z-Tree server is closed or crashes anytime before that, even though experimenters can recover data from previous periods, the data for the current period is lost. This is particularly a large issue for one-period experiment where a lot of activities are going on, such as long-form standardized tests programmed in z-Tree, or market experiments where a lot of buy/sell orders are placed. Reconstructing data tables by parsing GameSafe up to the point where the server crashes makes data salvage possible. In the best case scenario, when the server is accidentally closed before the last subject finishes his or her result screen, almost all data can be recovered.

Second, this software can recover overwritten variables. z-Tree's data tables only store the state of the variables at the end of each period or session, thus any intermediate state for a variable is lost. This can happen when subjects are allowed to repeatedly submit and change their decisions without leaving a stage, or simply due to programming oversight. Some z-Tree programs, such as GIMS (Palan, 2015), take meticulous care in design such that no variable is ever overwritten. However, not every z-Tree programmer can avoid this, and finished experiment cannot be redone. By reconstructing data tables at any point during the experiment through gsf, we can recover overwritten data ex post facto.

Finally, by understanding what went on during an experiment every step along the way, it helps to more precisely replicate previous experiments, or to understand why replication efforts fail (Camerer et al., 2016). Further, since gsf file is binary, which makes it harder to

intentionally modify its content, reconstructing datasets from gsf can serve as an "integrity check" for experimental data.

2 Features and Usages

Every plain-text exported gsf file consists of a sequence "events" order by time and indexed by an "`Event ID`" starting from zero. There are many types of events, including client connection, submission of input variables by subjects, entering/leaving a stage, etc. The software works by parsing these events in plain-text form into in-memory Python objects.

The software itself consists of three parts. First, after users are prompted to type in the filename for the plain-text exported gsf file, the program parses it from the beginning to the end, and stores all the events as a Python `list`. Next, the program goes through the object and writes the following tables into an Excel spreadsheet named "`timeline.xlsx`": (1) a table that contains the name of each connected z-leaf client, time of connection, and source IP address; (2) a table of experiment parameters, including number of groups, number of periods, number of subjects, show-up fee and exchange rate; (3) a table of events that change the existing data tables (which happen whenever a variable is changed or created on the server side, or when a subject submits a decision); this includes the time and type of the event, period, and tables/subjects/variables affected; and (4) questions and answers to the questionnaire, if applicable.

Finally, the last part of the program will prompt users to make a choice for what to do next. Choices include (1) entering a number for `Event ID`, and the program will recreate all the data tables up to this point in the experiment; (2) entering "`end`", and the program will recreate all the data tables at the end of the experiment (including incomplete experiment); or (3) entering "`all`", and the program will recreate all the data tables whenever there is a data-altering event; the program warns users that a lot of files will be created. Regardless of what users choose, a tab-separated "`txt`" file will be created for each z-Tree table, prefixed with the event ID (e.g. `1274_subjects.txt` for the `subjects` table up to the event with ID 1274).

The program can not only recreate z-Tree's built-in tables such as `subjects` and `contracts`, it can also handle custom tables. It is also designed to take into consideration cases where records in more than one table are modified at the same time (for example, in contract creation, `contracts` and `subjects` tables may be changed simultaneously).

3 Software and Hardware Requirements

3.1 Preparation and Package Dependencies

The first step before running the program is to convert the binary `.gsf` file to a plain text file using z-Tree's built-in gsf-exporting functionality. It can be done in z-Tree's by clicking menu "`File-Export-GameSafe...`", opening the gsf file, and then saving the converted text file[1].

The program is provided as a single-file command-line Python script and written mostly in vanilla Python. Non-standard library package dependencies are `XlsxWriter` and `pandas`, which can be easily obtained from the PyPI repository using the `pip` installer. The former package is used to write multiple tables of various dimensions into one Excel file in an easy-to-read format, and the latter is used to simplify programming by taking advantage of the `pandas.DataFrame` data structure.

Although z-Tree is required at the start to convert the binary gsf file, which in most cases can only be done in Microsoft Windows, the program can be run on any platform that runs Python, including macOS, Linux, and smart phones, once the text file has been generated. For convenience, we also provide a standalone executable for Windows that does not require the installation of Python and associated packages.

3.2 Performance

When the program is used in an "emergency", for example, the z-Tree program crashes before data tables can be saved and subjects paid, it is essential that the parsing and reconstruction of the datasets be done in a reasonable amount of time.

Fortunately, although Python is an interpreted language, this program is fast enough to reconstruct the datasets for use during an experiment session on personal computers and laptops without significant wait, even for very large gsf files with size in the hundred-megabytes and millions of lines. For example, for a 92MB converted gsf file (240MB unconverted) with 2.3 million lines, it takes 3.67 seconds to parse the file into in-memory Python objects, 3.61 seconds to generate the complete timeline spreadsheet, and 10.28 second to reconstruct all data tables at the end of the experiment[2].

[1] Note that GameSafe created with different major releases of z-Tree may not be backward and forward compatible. For example, z-Tree version 4 may not be able to successfully convert gsf files created from experiments conducted with z-Tree version 3.

[2] Tested on an AMD Ryzen R5 2600X desktop computer running Windows 10 and Python 3.7.

4 Limitations

There are several limitations to what this program can do and what type of data can be recovered, although some of those limitations are due to the nature of the gsf file itself.

First, some variables that appear in the normal tables such as `subjects` will not be in the gsf file, hence may not be recovered. Remember that GameSafe only stores the communications between the server and the clients. Therefore, variables that are only generated and stored on the server will not appear during the server-client communication. Most significantly, this may include the variable `Group`. Group matching is handled on the server side, and the server serves the role of a "router" that routes between-subject traffic determined by the matching. Random numbers generated in z-Tree programs are another example.

Second, the time stamp for each event is only accurate to 1 second. This may not be a significant issue for most experiments, however it may still affect studies demanding high time resolution. Moreover, time stamp for table changes resulting from contract creation or selection can be unpredictable.

Third, the current program will only output to spreadsheet the events that change the data tables. Events such as subjects leaving and entering stages are not written. This is done to reduce visual clutter by removing information of secondary importance. However, since all events are still parsed and stored in memory as Python dictionaries, with small changes to the program code, those events can also be written to the timeline file on-demand.

Finally, although the software supports gsf files for both version 3 and 4 as it currently stands, it may not work correctly for the next major release of z-Tree, because the format and organization of the convert gsf files may change from version to version (as it happened from z-Tree version 3 to 4). However, since the software is released under an open-source license, we hope our code can serve the experimental economics community as a foundation for future improvements, both compatibility-wise and feature-wise.

References

Camerer, C. F., Dreber, A., Forsell, E., Ho, T.-H., Huber, J., Johannesson, M., Kirchler, M., Almenberg, J., Altmejd, A., Chan, T., et al. (2016). Evaluating replicability of laboratory experiments in economics. *Science*, 351(6280):1433–1436.

Fischbacher, U. (1999). z-tree-zurich toolbox for readymade economic experiments: experimenter's manual. *Working paper/Institute for Empirical Research in Economics*, 21.

Fischbacher, U. (2007). z-tree: Zurich toolbox for ready-made economic experiments. *Experimental economics*, 10(2):171–178.

Palan, S. (2015). Gims software for asset market experiments. *Journal of behavioral and experimental finance*, 5:1–14.