(1) Overview

Title
CSVDataMerge: A Simple and Free Program for Concatenating Experimental Data Files

Paper Authors
1. Schmidt, James R.

Paper Author Roles and Affiliations
1. Associate Professor, LEAD-CNRS UMR 5022, Université Bourgogne Franche-Comté, France

Abstract
In experimental psychology and other applications, researchers will often have numerous datafiles (e.g., one for each participant) that need to be joined together into one larger dataset before data analyses. Many experimental software programs, including some increasingly popular ones (e.g., PsychoPy or OpenSesame), do not include data merging functionality. Copy-and-pasting (potentially error prone) or the writing of situation-specific scripts (potentially difficult and time consuming) may be necessary. CSVDataMerge was created as a free Java application that merges CSV (or comma-separated TXT) data with little more than a double-click. The program also appropriately deals with datasets that have different column orders in different datafiles or empty cells. More trivially, it can also concatenate datafiles that do not contain headers and allows the user to specify which columns to keep and in what order.

Keywords
data merging; experimental psychology; CSV files; Java; sorting; empty cells

Introduction
CSVDataMerge was primarily developed with the objective of merging together experimental datafiles (e.g., from individual participants in a psychology experiment). In principle, the program could be used (without modification) for merging any type of CSV (comma-separated values) files with or without column headers (whether saved with .csv or .txt file extensions), but I will talk about the merging of experiment datafiles in the rest of this paper. The program represents a simple solution to an inconvenience of certain experimental software programs (e.g., for designing and running psychology experiments). Many such programs (e.g., PsychoPy [7] and OpenSesame [4]) create individual datafiles for each participant, generally in CSV format. These programs are growing in popularity, especially for online

data collection (e.g., because they are free software and some popular paid software programs, like E-Prime [8], do not allow for online data collection), perhaps even more so recently with limitations on the ability to run experiments in the lab during the COVID-19 pandemic. The absence of a simple manner of concatenating together the individual datafiles in many such software programs represents a rather large inconvenience.

Indeed, after completing data collection, most experimenters will want one large dataset with the data of all participants combined into one file (e.g., to import into R [9] or another statistics program for data analysis). While some programs have automatic, but proprietary, data merging programs (e.g., E-Merge for E-Prime experiments), others do not. For some programs, it also depends on the options selected. For instance, when run online, PsychoPy can either store individual CSV files or create an organized database, but some forum discussions seem to indicate that the former option is more trustworthy than the latter. Simple copy-pasting of each individual dataset into a larger datafile is possible (e.g., in Excel), but this is both time consuming (e.g., if there are a large number of participants) and prone to error. The researcher or student could also imagine writing an experiment-specific script (e.g., in R, MATLAB [12], or Python), but this could also be time consuming and may be beyond the programming capabilities of the individual. I note that some other external tools exist that require a bit more work to use, for instance, by creating a Visual Basic script for Excel [2]. There are also paid (previously free) programs that require the researcher to upload data to the web [3], which may not be acceptable for data security reasons. Other free-to-use programs exists that are not as easy to use and do not deal with formatting inconsistencies [5], unlike CSVDataMerge (to be explained shortly). Another useful R package, prepdat, does exist [1], which also allows users to compute things like condition means and perform data trimming (using R commands), which may be a useful alternative for many users.

The objective of CSVDataMerge was therefore to provide a simple script that requires minimal setup and technical capabilities. Indeed, the user is only required to know how to install the Java Runtime Environment (JRE [6]; if they have not already), create and name a folder, copy-and-paste, and double-click. CSVDataMerge is a free-to-use Java program that automatically merges all datafiles placed within a “data” folder into a larger dataset file. The program is free to use, edit, and redistribute for non-commercial purposes (CC BY-NC-SA) and I provide open-source code with this manuscript.

In addition to simply merging files, CSVDataMerge addresses a few added complications. For example, most experiment files will have headers on the first line for the variable names in each datafile, which the experimenter will normally only want to copy once (e.g., before Participant 1) in the concatenated dataset (i.e., not repeatedly for each participant). Again, the redundant headers could be deleted manually (or filtered out during data analysis), but this is obviously not ideal. Simply concatenating together CSV files is therefore suboptimal. CSVDataMerge copies the column headers only once.

Furthermore, the author was disappointed to find out that at least one popular program (PsychoPy) does not always order the columns the same for all participants. For example, the "response time" variable might be in Column 5 for one participant and Column 35 for the next. Obviously, if these files are simply concatenated, values will not be in the correct columns for all participants, again leaving to the experimenter stuck either copy-and-pasting or writing some kind of situation-specific script to correct the inconsistencies. To my knowledge, no existing data merging applications can deal with this inconsistency. CSVDataMerge automatically reorders the columns to match the order of columns in a headers array that is either automatically extracted from the individual datafiles or specified explicitly by the user (see the different use cases below). For example, if the "response time" variable is in Column 5 for Participant 1 and Column 35 for Participant 2, all the values for Participant 2 in Column 35 will be moved to Column 5 to match Participant 1 (and the column headers).

The program also does not have problems with empty cells. The row lengths might sometimes be shorter than the header length in a CSV file if there are empty cells in the last columns of certain trials (i.e., rows). For example, if the last column is response time, but the participant did not respond on a certain trial, a blank value will not be recorded in this line. That is, the row will be shorter, with one less comma-separated value. CSVDataMerge detects this and simply adds a blank value for the corresponding cell (i.e., a comma). Further, on a more technical side, the line parser also correctly distinguishes between comma separators and commas within cell values (in addition to quotation marks).

More trivially, the program also allows the user to concatenate datafiles that do not contain headers (though the columns must be in the same order; see more details below). A header row can also be added to the concatenated file in the process (optional). For datafiles with headers, it is also possible to

indicate which columns to keep (e.g., if the experimental software generates "garbage" columns that are not needed for the final dataset) and the user can also indicate the order of the columns.

Implementation and architecture

Using the CSVDataMerge program has limited requirements. First, you will have to install Java installed on your computer, if you have not done so already. If not, download the Java Runtime Environment (JRE). This can be downloaded for free from Oracle [6]. Developers looking to edit the code will, of course, also need the Java Development Kit (JDK), also available from Oracle, and one of the many available IDEs (e.g., NetBeans, Eclipse, or simply Notepad++). Second, either (a) all of your datafiles should contain headers, or (b) none of the datafiles should contain headers (some different use cases will be described shortly). If the files do contain headers, then the headers need not be in the same order in each file, because CSVDataMerge will rearrange them if necessary. The program first cycles through all datafiles to extract all of the unique headers and then can match the headers in each individual file with the recovered headers list. Headers for the same column should match exactly, but normally this will be the case in data produced by experimental software. Third, your datafiles should use a comma-separated format, whether saved with the .csv extension or .txt. In the case that your datafiles are in a different format (e.g., XLSX) the program might be relatively easily adapted, though this would require at least some programming knowledge (albeit, very little). It is also worth noting that CSV is not a standardized format. Indeed, "CSV" file formats exist that do not even use commas as the separator (e.g., using a semicolon instead). Again, adaptations to the code would be necessary in the case of non-standard formats (e.g., not RFC4180).

There are a few different use cases possible. The simplest default use is with data that do contain headers and where users do not want to (a) reorder columns or (b) delete some columns in the concatenation process (both of which, of course, can be done easily after the fact). Readers that merely want to use the program without editing the source code can simply place the CSVDataMerge.jar file in whichever folder they wish, let us call this "MyExperimentFolder" (but you can use whatever name you want). In the same folder, they should create a new folder and name it "data" (case sensitive). This is illustrated in Figure 1. All the individual-participant CSV files from the experiment with either a .csv or .txt file extension should be placed in this "data" folder. Note that no other non-data files with the same file extension should be placed in the same folder, but files with other file extensions (e.g., .log) will be ignored. Note also that the program will first

check for files with a .csv extension. If none are found, it will check for files with a .txt extension. Thus, “MyExperimentFolder” should have CSVDataMerge.jar and a folder called “data” (which, in turn, is filled with your CSV or TXT files). Double-clicking on the JAR file will produce a new file in the “MyExperimentFolder” called dataset.csv, also shown in Figure 1. This is your concatenated datafile.

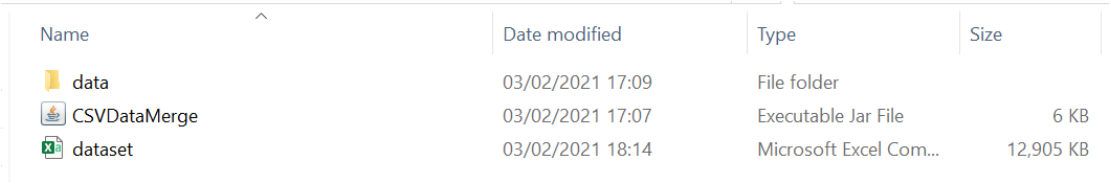

Figure 1. To use CSVDataMerge, simple copy the JAR file into a folder and place the CSV datafiles into a folder named “data” (case sensitive). Double click on CSVDataMerge.jar and a file called dataset.csv will appear.

Note that you will need to delete all files in the “data” folder before merging new datafiles. Alternatively, given that the JAR file is so small (7 KB), it can simply be copied into the experiment folder of each individual experiment that you want to use it for. Additionally, dataset.csv is overwritten each time that the JAR file is executed, so make sure that you have either renamed or moved dataset files that you would like to keep before merging new data into the same folder.

As a second use case, the user can specify which columns to keep and/or reorder the columns. To do this, a new file called “headers.csv” should be saved to the same folder illustrated in Figure 1. This file should contain only one comma-separated line with the headers in the order that the user wishes. For instance, the user can exclude columns that are not needed and/or reorder the columns as appropriate. Column names must match exactly those in the individual datafiles, including for instance trailing white spaces. Note that this CSV file can be created in one of two ways. First, by putting the headers in the first row of an Excel document (e.g., the first three headers in cells A1, B1, and C2), then saving the file as a comma-separated data (.csv) file. Alternatively, the user can open a simple text editor (e.g., Notepad), and enter a comma-separated list of headers (e.g., “Participant,RT,ACC”) on a single line. Do not add extra white spaces between columns (e.g., “Participant, RT, ACC”). Then save the file as a CSV file (i.e., not TXT). An example headers.csv file is provided in the GitHub repository. Note that if a headers.csv file exists in the relevant folder, the program will use this file for determining the columns. Thus, if you instead want to use the default approach preceding this one, either delete or rename the headers.csv file.

As a third use case, datasets without headers can also be merged. Some experimental software programs, such as PsyToolkit [10,11], do not save column headers. Merging together headless datafiles is rather trivial, but CSVDataMerge also allows for the merger of these types of datafiles. In this case, a headers.csv file *must* be created. The first line must read, exactly "no headers", with no trailing characters (note that if saved from Excel, trailing commas may have to be removed manually with a text editor, such as Notepad). A second line must also be added to this file. The second line should either (a) contain a comma-separated list of headers that you want to print at the top of the concatenated file, or (b) read (again, exactly) "leave blank". In the second case, no headers will be added to the concatenated file. Note that for this third use case, it is assumed that all datafiles have the same columns in the same order. This should hopefully be the case for any experimental software program that does not print column headers in individual datafiles. It is impossible to reorder columns if there are no headers to compare across datasets. Example headers.csv files are also included in the repository for these use cases.

For those looking to edit the Java source code, the program is relatively straightforward, but the following pseudocode indicates the processing logic of the program:

```
If "headers.csv" file exists
   Determine whether:
      Case 1: No headers exist or should be added
      Case 2: No headers exist but a header array is specified
      Case 3: A custom headers array is specified
For each file in "data" folder:
   Add to list of file names
If no "headers.csv" file exists:
   For each file in list:
      Split first row into separate header values
      Add new header values to headers array
If a headers array exists:
   Print headers to first line of file
   For each file in file list:
      Split first row into header values and store in array
      If header order does not match headers array:
         For each header in headers array:
            Find corresponding header in new file
            Set position index for new file
         For each row (excluding header row) in new file:
            Split row into values and store in array
            Write values in adjusted order to "dataset.csv"
      Else (i.e., the headers do match):
         Write each row in file to "dataset.csv"
Else (i.e., a "headers.csv" file exists):
   For each file in file list:
      Read line
```

```
Print line to "dataset.csv"
```

Quality control

CSVDataMerge was tested for correct functionality with a large set of "problematic" datafiles. Some initial datafiles were created with PsychoPy (more precisely with the PsychoJS JavaScript-converted code for data collection online via Pavlovia.org). All datafiles contained the same columns of data, but (a) the order of these columns, for unclear reasons, were not consistent from one participant to another, and (b) numerous cells were empty for a variety of reasons (e.g., if the participant did not respond fast enough, a blank value for their response time was recorded; other columns track trials within certain blocks and cells were therefore empty for trials from other blocks). It was confirmed that CSVDataMerge deals with both problems (a) and (b) above by assuring that the merged data (a) correctly reordered all cell data to match the extracted headers array and (b) correctly coded blank values and maintained the column orders for all values.

These original datasets are not included in the repository, as they contain sensitive information (Prolific.ac participant numbers), which would be difficult to remove. Other test data files were created and are available in the GitHub repository. These files contain the same two problems plus two additional ones: cell values that contain commas and cell columns that contain quotation marks. Both of these extra additions would create problems if the program was not appropriately parsing the lines of the datafiles to recognize when a comma represents a separator and when it is simply part of the cell value. The program was confirmed to deal appropriately with these cases. Though it is difficult to imagine how the program could sort data incorrectly, to ensure that the resulting datafile has been merged correctly, the user can simply scroll through the datafile in Excel to check that the correct values are presented in each column, especially the final columns. If anything is incorrect, the errors should be very apparent to the eye (e.g., empty cells where there should not be). The source code and JAR file are available on the GitHub. The GitHub project is also linked to the Open Science Framework (https://osf.io/g3qw8/) and the webpage of the author (http://leadserv.u-bourgogne.fr/~jschmidt/CSVDataMerge/).

(2) Availability

Operating system

The JAR file will work on Windows, Mac, and Linux systems.

Programming language
Java (programmed with JDK build 11.0.1). The user does not require the JDK (Java Development Kit), however, unless wanting to adapt the code (e.g., with NetBeans, Eclipse, Notepad++, or another code editor).

Additional system requirements
Memory and processing requirements are trivial. The user will need an internet connection for downloading Java and the JAR file (and/or JAVA file, as applicable) before first use.

Dependencies
Java Runtime Environment (JRE), version 1.8.0_271 or higher.

List of contributors
NA.

Software location:

Archive 1

Name: Institutional repository
Persistent identifier: http://leadserv.u-bourgogne.fr/~jschmidt/CSVDataMerge/
Licence: GNU General Public License v3.0
Publisher: James R. Schmidt, Université Bourgogne Franche-Comté (UBFC)
Version published: 1.0
Date published: 25/10/21

Archive 2

Name: Open Science Framework
Persistent identifier: https://osf.io/g3qw8/
Licence: GNU General Public License v3.0
Publisher: James R. Schmidt, Université Bourgogne Franche-Comté (UBFC)
Version published: 1.0
Date published: 25/10/21

Code repository

Name: GitHub
Identifier: https://github.com/james-schmidt/CSVDataMerge
Licence: GNU General Public License v3.0
Date published: 05/02/21

Language
Executable (JAR), source code (JAVA).

(3) Reuse potential

The program can be used to merge any type of compatible (truly comma-separated) CSV datasets (including comma-separated TXT files) into one larger dataset. If the datasets contain column headers on the first row, they need not be in the same order. In the datasets do not contain headers, all columns must be in the same order. I provide readers with an executable JAR file and the Java source for two separate purposes, both available as supplementary materials and available in online repositories. The Java source is provided for developers that may wish to edit the source (e.g., for adapting the script to a different file format).

Acknowledgements

NA.

Funding statement

This work was supported by the French “Investissements d’Avenir” program, project ISITE-BFC (contract ANR15-IDEX-0003) to James R. Schmidt.

Competing interests

The author declares that he has no competing interests.

References

[1] Allon, A S & Luria, R 2016, ‘prepdat: An R package for preparing experimental data for statistical analysis’, *Journal of Open Research Software* 4: e43. DOI: http://doi.org/10.5334/jors.134
[2] de Bruin, R 2020, *Merge all CSV or TXT files in a folder in one worksheet*, viewed 4 February 2021, <http://www.rondebruin.nl/win/s3/win021.htm>.
[3] *gridoc.com* 2021, computer software, viewed 4 February 2021, <https://gridoc.com/>.
[4] Mathôt, S, Schreij, D & Theeuwes, J 2012, ‘OpenSesame: An open-source, graphical experiment builder for the social sciences’, *Behavior Research Methods* 44: 314-324. DOI: 10.3758/s13428-011-0168-7
[5] Mueller, S T & Piper, B J 2014 ‘The Psychology Experiment Building Language (PEBL) and PEBL Test Battery’, *Journal of Neuroscience Methods* 222: 250–259. DOI: 10.1016/j.jneumeth.2013.10.024
[6] Oracle 2021, ‘Java Runtime Environment’, computer software, <https://www.oracle.com/java/>.
[7] Peirce, J W, Gray, J R, Simpson, S, MacAskill, M R, Höchenberger, R, Sogo, H, Kastman, E & Lindeløv, J 2019, ‘PsychoPy2: experiments in behavior made easy’, *Behavior Research Methods* 51: 195–203. DOI: 10.3758/s13428-018-01193-y

[8] Psychology Software Tools, Inc 2016, computer software, viewed 4 February 2021, <https://support.pstnet.com/>.
[9] R Core Team (2018). 'R: A language and environment for statistical computing', R Foundation for Statistical Computing, Vienna, Austria, <https://www.R-project.org>.
[10] Stoet, G 2010 'PsyToolkit - A software package for programming psychological experiments using Linux', *Behavior Research Methods* 42: 1096–1104. DOI: 10.3758/BRM.42.4.1096
[11] Stoet, G 2017 'PsyToolkit: A novel web-based method for running online questionnaires and reaction-time experiments', *Teaching of Psychology* 44: 24–31. DOI: 10.1177/0098628316677643
[12] The MathWorks Inc., 2018, computer software, viewed 18 October 2021, <https://www.mathworks.com/products/matlab.html>.