Journal of open research software

Open-Source MUltiple Tests Corrections and FOrmatted Tables Software (MUFOS)

SOFTWARE METAPAPER

NIKOLAY PETROV
VASIL ATANASOV
TREVOR THOMPSON
*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

The *p* value statistic remains a ubiquitous indicator of the verisimilitude of experimental hypotheses. However, testing multiple hypotheses poses a problem as Type I error rate is inflated. Despite known solutions, this problem remains largely neglected for two reasons: 1) most data analysis tools offer limited multiple tests correction options; 2) the learning curve of existing tools requires hefty time investment. To address these concerns, we present a free, easy-to-use and convenient Software, built around established python libraries, that allows users to apply a MUltiple tests correction and get the results in a readily-understood, FOrmatted table (MUFOS) – *https://github.com/nikbpetrov/mufos*.

CORRESPONDING AUTHOR:
Nikolay Petrov
School of Human Sciences, University of Greenwich, UK
nikbpetrov@gmail.com

KEYWORDS:
Multiple tests adjustment; bonferroni; p value; open-source; software; statsmodels

TO CITE THIS ARTICLE:
Petrov N, Atanasov V, Thompson T 2022 Open-Source MUltiple Tests Corrections and FOrmatted Tables Software (MUFOS). *Journal of Open Research Software*, 10: 5. DOI: *https://doi.org/10.5334/jors.350*

1. OVERVIEW

1.1. BACKGROUND

Scientists usually rely on probability in order to establish the validity and reliability of their findings. The most common framework is Null Hypothesis Significance Testing (NHST) in which the probability of an experimental hypothesis (that there is a relationship between variables) is tested against the probability of a null hypothesis (that there is no relationship between variables).

To verify their experimental hypotheses when working under the NHST framework, scientists most commonly use the *p* value statistic, which is the probability of obtaining results as extreme as the observed ones, if the null hypothesis is true. Although the best approach to scientific discovery often starts with a minimal but sufficient set of theoretically-justified hypotheses, in exploratory research, where little is known about the topic, testing multiple hypotheses is often unavoidable. However, when multiple hypotheses are put forward, the chance of Type I error – or a false positive – increases proportionally. For instance, using the accepted threshold of alpha = 0.05, the probability of a false positive is only 1/20. However, if 14 tests are performed, then it is more likely than not[1] that at least one test will be heralded as a potentially important finding, even if there is *no effect present*. Fortunately, this problem is well-known and solutions to an inflated Type I error have been developed.

There are two primary ways to prevent Type I errors when conducting multiple testing. The first is to control the family-wise error rate (FWER), which is probability of making at least one Type I error in a set of tests. The most popular and easiest way to do this is to use a Bonferroni correction, in which a specific alpha threshold criterion (typically .05) is adjusted for the number of tests performed such that the adjusted alpha = .05 / number of tests (e.g. if five tests are conducted, the adjusted alpha criterion would be .05/5, or .01). In this way, the 5% is 'spread out' amongst the competing tests. An alternative and equivalent method is to make no changes to the adjusted alpha threshold, but instead to adjust the obtained *p* values. For example, if a Bonferroni correction is applied, the observed *p* value is multiplied by the number of tests (assumed to be independent) conducted and all hypothesis tests with adjusted *p* value above .05 are rejected. Other FWER controlling procedures include the Šidák [1], the Simes-Hochberg [2] and the Hommel [3] corrections as well as the Holm-Bonferroni and the Holm-Šidák [4]. The second most commonly used way to avert an inflated Type I error rate is to control the false discovery rate (FDR), which is the proportion of false positive tests to all tests that reject the null. Controlling the FDR has the advantage of increasing power and is most valuable when conducting a large number of tests. FDR-controlling procedures include the Benjamini-Hochberg [5] and Benjamini-Yekutiely [6] procedures as well as an adaptive, two-stage version of both of them [7, 8]. For in-depth performance comparison and treatise of different methods, the interested reader is referred to the relevant literature [9–11]. Several excellent sources are also available for reviews of more advanced correcting procedures [12, 13] and for recommendations when a multiple tests correction is warranted [14–19].

Although the problem of multiple testing is well known and solutions have already been put forward, it remains dangerously neglected in applied research. For instance, in a survey of published work in neuroimaging journals for the year 2008, Bennet and colleagues [20] found that between 15% and 40% of studies failed to include adjustments for multiple testing. Austin and colleagues [21] have also shown how failure to adjust for testing multiple exploratory hypotheses can result in implausible practical results. In another survey of 800 pathology papers published in 2003, it was found that out of the 37 studies who had performed multiple comparisons, 56% failed to account for inflated Type I error rate [22]. In yet another review of articles published in major optometry journals between 2003 and 2013, 47 out of 142 (33%) failed to correct for multiple comparisons, while out of the remaining 95 that applied a correction, merely 9% of them provided justification for their choice of procedure [14].

The problem – no tool for the job

One reason for the lack of implementation of multiple testing corrections is likely to be the lack of a quick, easy and convenient way to perform them. Specifically, we consider that there are certain conditions that an ideal tool should satisfy in order to reduce the friction between the desire to apply multiple tests corrections and the practical application. These conditions are 1) having a friendly user interface that makes applying multiple tests adjustments quick, free and easy; 2) implementing a wide range of multiple tests corrections to suit different research goals; 3) flexibility in accepted formats of the input data to suit different preferences; 4) producing convenient, easy to use and readily-understood output.

To our awareness, there is currently no tool that satisfies all these conditions. On the one hand, any programming tools – such as Matlab [23], R [24] or Python [25] – involve a steep learning curve, which is likely to preclude beginner non-statisticians from using them. On the other hand, user-interface driven tools, such as SPSS Statistics [26] and Microsoft Excel [27] do not have a readily-available, convenient and reliable way to apply a multiple test correction to any set of tests. If performing a multiple test correction is not arduous enough, then comes the added difficulty of getting the results quickly and easily into a readily-understood, formatted output, which places significant and unnecessary demands on time for scientists – programmers or not [28].

One of the two pieces of software that comes close to fulfilling the above conditions is JASP [29], which is

user-interface driven, free and relatively easy to use compared to most other commercial software, and outputs the data in formatted tables according to the guidelines of the American Psychological Association [30]. Critically, however, JASP does not have the option of applying multiple tests corrections in any of its statistical procedures. The second piece of software that comes close is Lesack and Naugler's Bonferroni Calculator software [22]. It is free to use and allows the user to apply a Bonferroni, Bonferroni-Holm or Benjamini-Hochberg correction to any number of *p* values. Unfortunately, however, on top of having a paucity of correction options, the software must be run in the command line as it has no user interface, and it is not convenient due to limited options for inputs and outputs.

Thus, the aim of this paper is to present a piece of software that meets the conditions set above. Specifically, the MUltiple tests corrections and FOrmatting tables Software (MUFOS) is free, has a simple user interface, allows a wide range of multiple tests corrections to any set of tests, the input data can take various formats and produces an easy to read, formatted output. MUFOS is built on top of established statistical libraries, which strengthens the reliability of its output.

1.2. IMPLEMENTATION AND ARCHITECTURE

MUFOS is a user-interface driven application (see ***Figure 1***), written in the programming language Python, version 3.8 [25]. There are ten multiple tests corrections users can apply: Bonferroni, Šidák, Holm-Šidák, Holm-Bonferroni, Simes-Hochberg, Hommel, Bejnamini-Hochberg, Benjamini-Yekutieli, Benjamini-Hochberg (2-stage), Bejamini-Yekutieli (2-stage). These corrections can be applied to the *p* values of one of four tests: correlations, multiple regression, independent samples t-test, or paired samples t-test; or they can be applied to any standalone list of *p* values.

The software allows users to input data in multiple different formats as long as they can be fit into an excel (.xlsx) or a comma-separated values (.csv) file. Input file types include a variety of formats such as raw data, summary statistics (e.g. means and standard deviations) and inferential statistics (e.g. *p* values) as described below:

a) raw data, in which each variable is a separate column and each row is a new value for the variable (i.e. the typical rectangular dataset) – see ***Figure 2A***;
b) correlation table data, in which one column contains one variable's name, a second column contains the other variable's name and third and fourth columns which contain the correlation coefficients and associated *p* values, respectively. Each row represents an additional correlational test between variables – see ***Figure 2B***;
c) independent t-test statistics, in which there is one column for each dependent variable name, six columns that contain information about the mean,

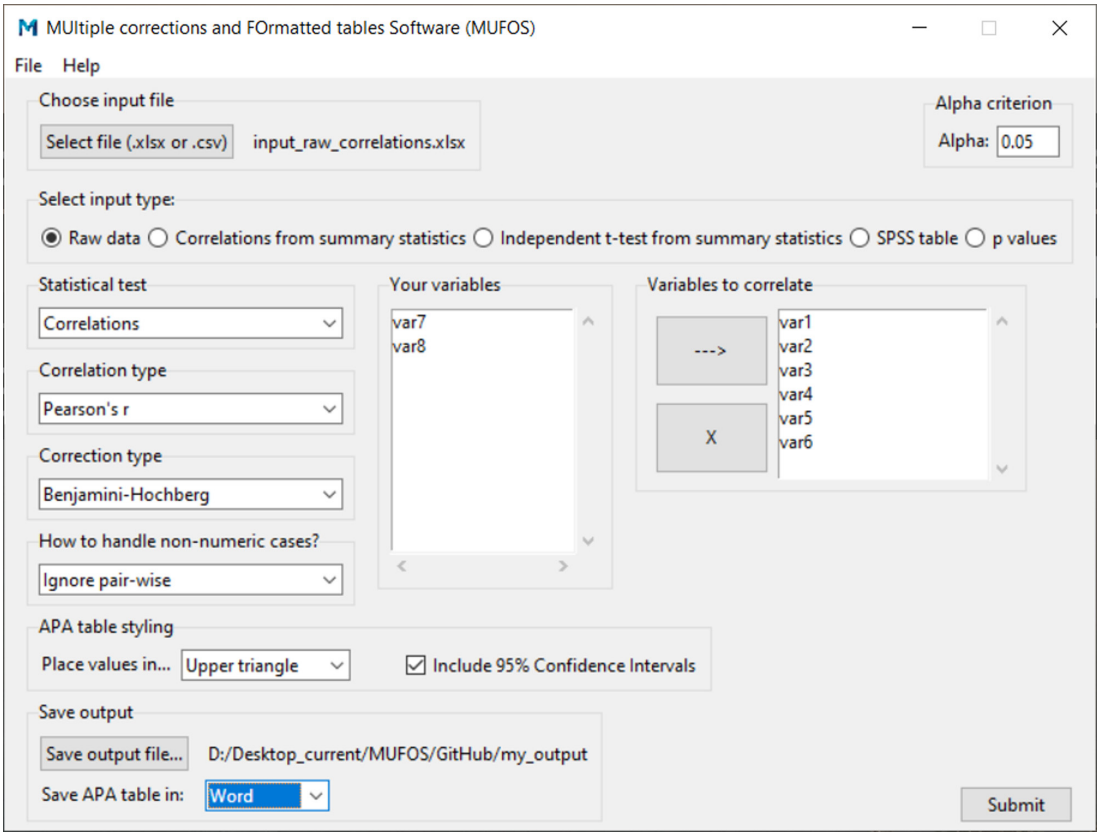


Figure 1 Look and feel of the MUFOS user interface for one option.

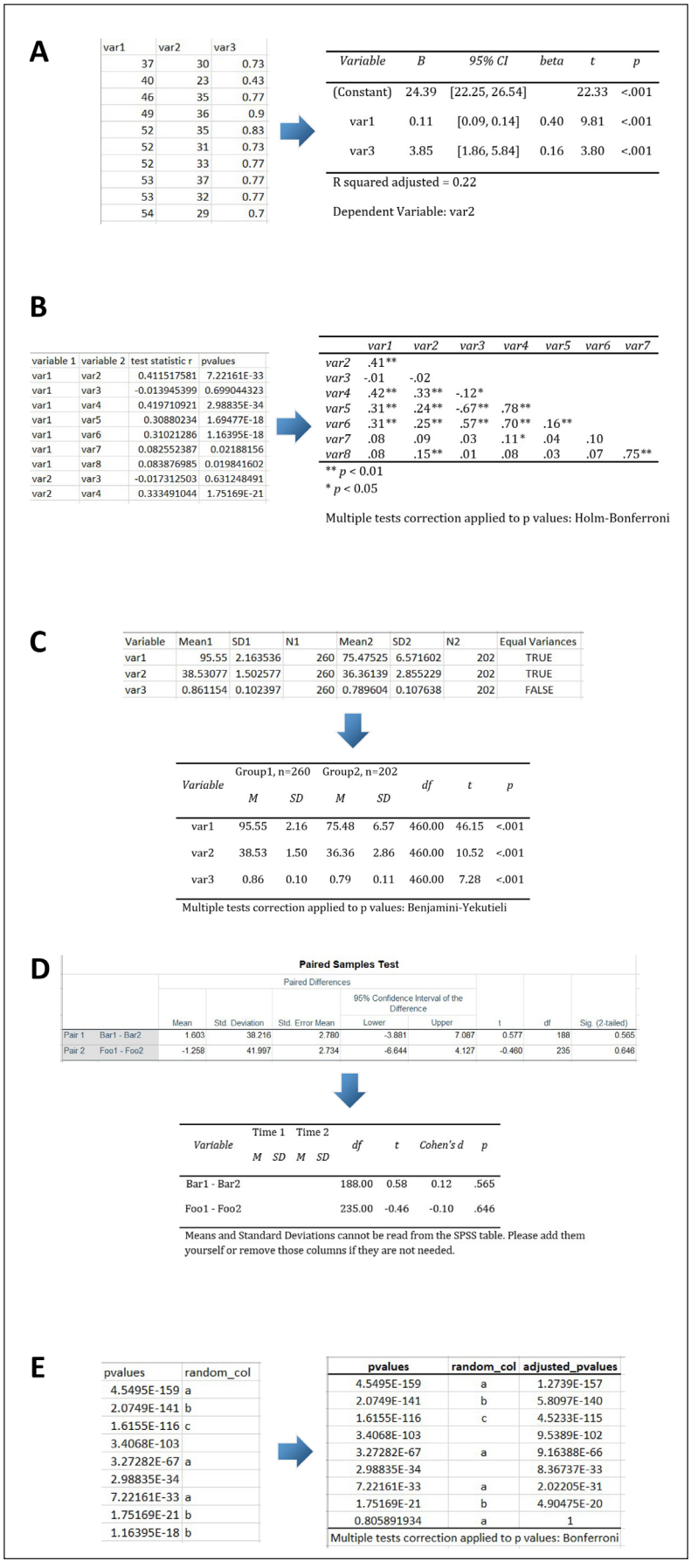


Figure 2 Examples of input files and the resultant output files after processing with MUFOS. **1A** presents a sample raw data input file and a formatted multiple regression table; **1B** presents a sample input file for correlations from summary statistics and a formatted correlation table; **1C** presents a sample input file for independent samples t-test from summary statistics and a formatted table; **1D** presents a sample SPSS Statistics input file for paired samples t-test and the resultant formatted table; **1E** presents a sample *p values* input file and the resultant output table.

standard deviation and sample size for each group (3 statistics by 2 groups) and an optional column whether equality of variance is assumed (if this column is missing, equality of variance is assumed). Each row represents a new group comparison – see ***Figure 2C***;

d) SPSS table, which is an excel (.xlsx) file, produced by exporting a specific table from the SPSS Output window (for step-by-step instructions on how to export an SPSS table, see the *MUFOS Help Guide and Documentation* in the software repository) – see ***Figure 2D***;

e) *p* values table, which must contain a column of *p* values, with other optional columns – see ***Figure 2E***.

The output files are provided as formatted tables, currently following the American Psychological Association (APA) 7th edition guidelines [30], but more styling options are to be made available. For all input data options except a *p* values list, an APA-formatted table is available as either an excel or word document. When a list of *p* values is provided, the output file is the same as the input file with an additional column for the adjusted *p* values. See ***Figure 2*** for example input and output files.

Software development

To develop the software, the main libraries used are *pandas* (version 1.1.0) [31] and *numpy* (version 1.19.1) [32], which are Python's core libraries for data analysis. The multiple test corrections are performed using the *multitest* module from the *statsmodels* (version 0.11.1) library [33]. Relying on established and widely used libraries ensures the stability and reliability of any calculations. For more information on how each calculation is done, see *the MUFOS Help Guide and Documentation* document in the online repository (details below). The user interface is built with Python's native *Tkinter* library [34]. Additional Python's native libraries used are *sys*, *os*, and *traceback,* used for file handling and error logging, *threading* and *math*, used for custom in-house progress bar algorithm implementation, and *webbrowser* for opening system folders and external links. Other external libraries used are *python-docx* (version 0.8.10) for writing .docx files [35], *openpyxl* (version 3.0.4) for writing .xlsx files [36], and *requests* (version 2.24.0) for creating a REST request to check if a new version is released [37]. To create an executable file from the source code, the *pyinstaller* (4.0) library is used [38].

Architecture

The software's root file is where the user interface code resides. After the user interface setup, upon clicking the *Submit* button, three processes are executed (***Figure 3***). The first one is setting the global variables. These variables are extracted out in a separate file that contains the data that the user has provided from the user interface (e.g. input filename, input type, statistical test, correction to apply and others) as well as variables shared across files (e.g. table styling options). The second process is to validate the user's input – here a series of logical checks verify that the user has not missed filling out a required field and that the provided input is as

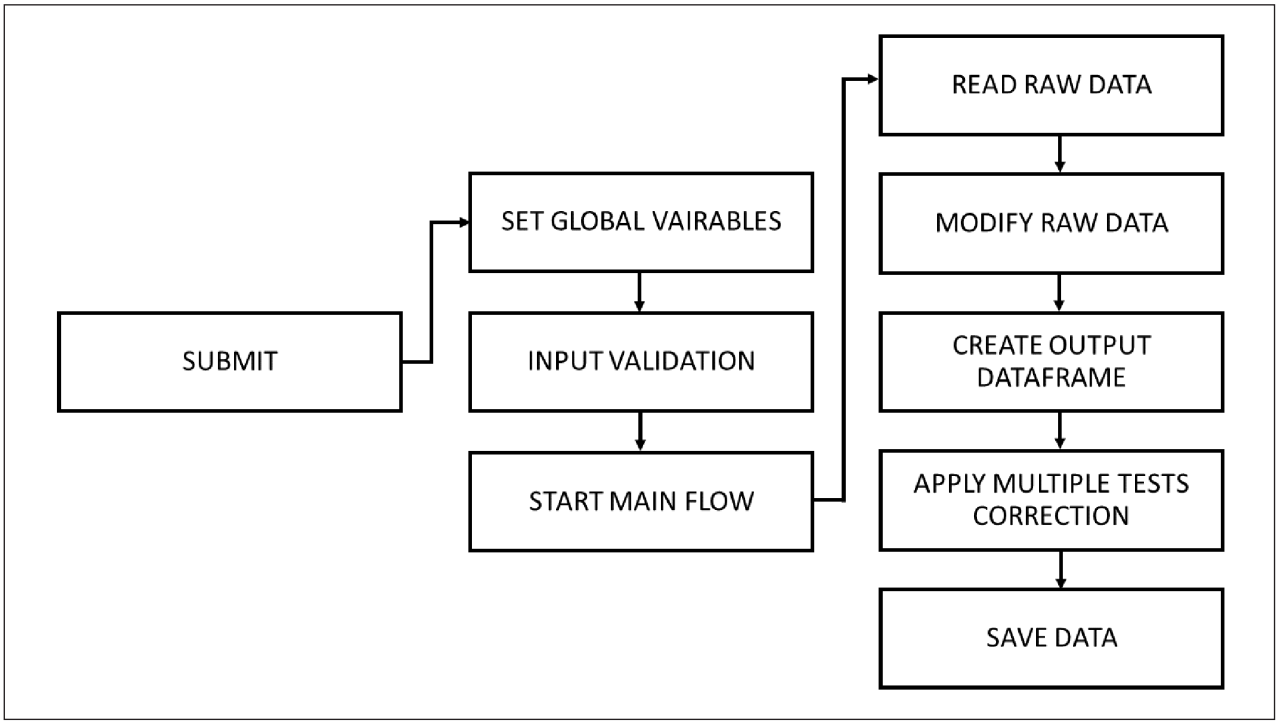


Figure 3 Flowchart of MUFOS architecture. On submit, three subsequent processes are executed – setting global variables, validating input and then main flow is started. The main flow consists of five subsequent steps – reading the raw data, modifying it, creating an output dataframe, applying a multiple tests correction and finally saving the output.

expected. The third core process that is triggered upon clicking submit is starting the main software flow. This third process is further divided into five Steps. The first Step is to read the provided input file. The second Step modifies and verifies the input file. In this step further error handling and logical checks ensure that the data is ready for manipulation. The third Step uses the modified input file to create an output dataframe which contains all relevant information and statistics that need to be used later for creating the formatted table. The fourth Step adds an additional column with adjusted *p* values to the output dataframe from the previous step. The fifth and final Step saves a formatted table based on the output dataframe.

Steps 2, 3, and 5 are decision functions, which perform a different action based on the user input. There are 11 different actions that can be performed at each Step: four actions (one for each statistical test) when the user input is raw data, four actions (one for each statistical test) when the user input is an SPSS table, one action for each of correlations from summary statistics, independent samples t-test from summary statistics and *p* values. Those 11 actions make up the main functionality of the software. Each of the 11 actions are in a separate file, the name of which is prefixed with *main_funcs,* followed by the input type (*raw*, *summ*, *pvalues*) and a statistical test (*correlations*, *mr*, *indttest*, *pairttest*). Each separate file contains functions that reflect Steps 2 (modifying the raw data), 3 (creating an output dataframe), and 5 (saving the data). Any functions that are shared across different paths or are used as helper functions are extracted into a separate file (*helper_funcs*).

1.3. QUALITY CONTROL

One of the key strengths of MUFOS is that it is built around established, up-to-date and thoroughly validated Python libraries. And yet, as a further quality control measure, two core sets of unit tests have been developed. The first one is a set of forty tests that verifies that all statistical procedures are implemented and working correctly. The expected output of each calculation is calibrated against SPSS and R. These tests compare the output dataframe from Step 3 within MUFOS against expected output files that contain the same calculations. These calculations are verified against both R and SPSS for statistical tests (correlations, multiple regression, independent and paired samples t-tests), while for multiple tests corrections and effect size estimates, the calculations are verified against R only as SPSS does not provide an out-of-the-box way to perform them. The second set of 70 unit tests verifies that the resultant formatted tables are in the expected format. In these tests, the output tables from MUFOS are compared against expected tables, which are manually checked and verified. For .docx files, the XML of the MUFOS output table is compared against the XML of the expected table, while for .xlsx files, the properties of each cell of the output table are compared against the properties of the respective cell from the expected tables. The source code for the tests is in the files prefixed with *tests_* in the main directory of the software's source code folder, while the files used for the tests are located in the *unit_testing* folder.

All unit tests are executed as part of an automated process that generates the compiled version of the software from the source code. Specifically, upon each new release, a set of automated actions within the software repository is triggered on a virtual machine. These actions consist of installing Python and all of the code's dependencies, running the unit tests and compiling the source code using *pyinstaller*. For the windows distribution, the resultant archive file from the automated process is manually converted to an installation file using the Nullsoft Scriptable Install System [39] tool. Due to an issue with *pyinstaller*, the automated process fails for the Mac distribution, hence the compiling process for each release is done manually on a local computer.

Despite the assiduity with which error handling is implemented and the diligence in writing unit tests, software failures can still occur. To account for that, an error logging procedure is implemented such that if an error occurs, the user will see a pop-up with an error message. Given the extensive error handling throughout, hopefully most of these error messages will make sense and allow the user to quickly troubleshoot and rectify the problem. However, in the rare cases that the error message is unexpected and nonsensical, the user will be prompted to save the error details in a log file and asked to email it to the developers. Notably, the log files will contain the minimum necessary information to reproduce the error to protect the privacy of the user's data; the user will also be encouraged to check the log file before sending to ensure they are not breaching data privacy protocols.

Such feedback loop from the user to the developers via error logging, however, is only useful if there is an efficient process to quickly update the software and notify all users. To address this, users will be notified of a new release as they use the software. Specifically, every time a user runs MUFOS, a REST request is triggered that checks if there are any new releases in the software repository. If so, the user is notified of what the latest changes are and is encouraged to download the latest release.

2. AVAILABILITY

2.1. OPERATING SYSTEM

MUFOS will run both under Windows (preferably Windows 7 and above) and MacOS (preferably MacOS 10.9 and above) operating system.

2.2. PROGRAMMING LANGUAGE
Python 3.

2.3. ADDITIONAL SYSTEM REQUIREMENTS
Overall up to 500 megabytes of free disk space.

2.4. DEPENDENCIES
Given the comprehensive packaging done on the production side, there are no dependencies for MUFOS for the end user.

2.5. SOFTWARE LOCATION
Archive
Name: Zenodo
Persistent identifier: *10.5281/zenodo.4058358*
Licence: GNU General Public Licence version 3.0
Publisher: Nikolay Petrov
Version published: 1.0
Date published: 29th September 2020

Code repository
Name: GitHub
Identifier: *https://github.com/nikbpetrov/mufos.git*
Licence: GNU General Public Licence version 3.0
Date published: 26th September 2020

User-friendly software page
The software can also be found at a more user-friendly page – *www.nikolaybpetrov.com/mufos*. Users can download the latest installation files from there as well as the software's help guide and sample input/output files.

2.5. LANGUAGE
English

2.6. INSTALLATION
To use the software, simply go to software location (either its repository or webpage) and download the installation file based on the operating system you have (Windows or MacOS).

For Windows users, simply download the installation file and follow the instructions on the screen. After a short installation is completed, run the MUFOS *.exe* file.

For MacOS users, first, download the installation archive. Then use an archive tool to unzip the files. After this process is complete, the MUFOS *.app* file can be run. Note that MacOS users might have to change their system preferences for MUFOS to allow their computer to open software for unidentified developers. More information on how to change preferences to open apps from unidentified developers can be found online (*https://www.macworld.co.uk/how-to/mac-software/mac-app-unidentified-developer-3669596/*).

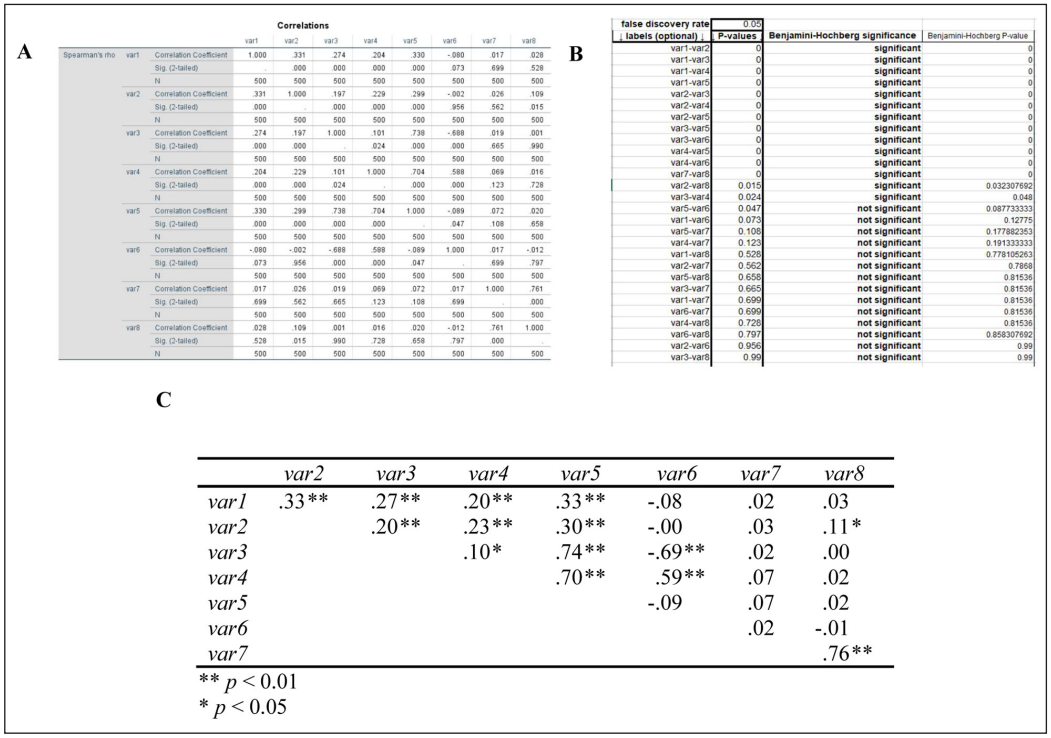
3. USE AND REUSE POTENTIAL

To showcase the MUFOS potential, let's examine a case study. A researcher has collected data across eight continuous variables and wants to run a correlational analysis using spearman's rho coefficient with a Benjamini-Hochberg (BH) correction. Suppose the researcher is not familiar with any programming language and prefers to use user interface-driven tools, such as SPSS Statistics and Microsoft Excel, to get the job done. This process would involve three steps (see ***Figure 4***). The first step is to run the correlational analysis in SPSS; by default, this produces a table like the one shown in ***Figure 4A***. The second step is to apply a BH correction. To do that, the researcher finds a template excel spreadsheet, the most popular one being [29], and manually copies over the p values from the SPSS table to the excel spreadsheet (see ***Figure 4B***). The third and final step is to create an APA-styled table in Microsoft Word, in which the researcher has to manually copy across all relevant information while cross-referencing both the original SPSS table and the corrected *p* values (see ***Figure 4C***). This process is laborious, non-replicable and error prone.

MUFOS makes this entire process easy, convenient and easily replicable. By simply clicking a few buttons (***Figure 5***), the researcher is rewarded with a perfectly formatted APA table (***Figure 4C***) ready to be inserted into their manuscript. MUFOS has the added benefits that researchers can customize the style of their APA table by selecting where they want their values to appear (lower triangle, upper triangle or both) and can even include confidence intervals, an option otherwise not readily available within either SPSS or Excel (see ***Table 1***).

The convenience and time-saving benefits of MUFOS are further complemented by a potential for expansion of the software and its core features. Specifically, users of MUFOS are warmly encouraged to suggest new features, such as new statistical tests, new multiple tests correction procedures or new styles for formatting a table. Researchers can implement this themselves if they wish, provided they abide by the licensing agreement. Expanding the software is made relatively easy as the architecture is built with this possibility in mind.

EXPANDING MUFOS – ADDING A NEW STATISTICAL TEST
Let's say a chi-square test needs to be implemented as a new option for statistical test if the user's input is raw data. There are four major changes that need to be implemented. The first is to update the user interface to include the test as a dropdown option and add any other options that user might need (e.g. multiple tests correction choice, how to handle non-numeric data). The second change is to update the Step 2, Step 3 and Step 5



	var2	*var3*	*var4*	*var5*	*var6*	*var7*	*var8*
var1	.33**	.27**	.20**	.33**	-.08	.02	.03
var2		.20**	.23**	.30**	-.00	.03	.11*
var3			.10*	.74**	-.69**	.02	.00
var4				.70**	.59**	.07	.02
var5					-.09	.07	.02
var6						.02	-.01
var7							.76**

** $p < 0.01$
* $p < 0.05$

Figure 4 A three-step process to run a correlational analysis using user interface driven tools. **4A** shows the default SPSS output for a correlations table; **4B** shows the copied variable names and *p* values within a publicly available template spreadsheet for applying a Benjamini-Hochberg correction; **4C** provides an example APA-formatted table which is produced by manually copying over the results from the previous steps.

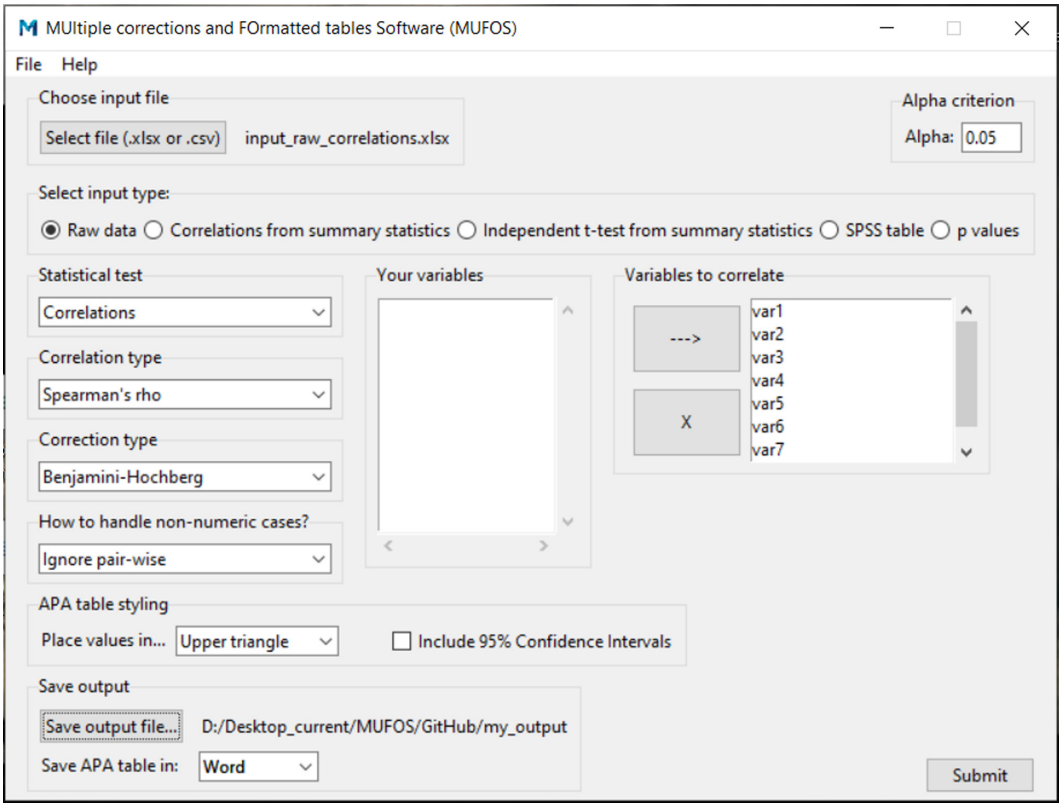


Figure 5 Options to be selected in MUFOS to recreate the three-step process from Figure 4.

decision functions (see 1.2. above) to include chi-square test from raw data as a new path. Then the third step is to create a new file in the same style as the other *main_funcs* files that contains a function that modifies the raw data (Step 2), creates an output dataframe (Step 3) and saves the output in a formatted table (Step 5). The fourth

	VAR1	VAR2	VAR3	VAR4	VAR5	VAR6	VAR7
var2	.33 **						
	[.25, .41]						
var3	.27 **	.20 **					
	[.19, .35]	[.11, .28]					
var4	.20 **	.23 **	.10 *				
	[.12, .29]	[.14, .31]	[.01, .19]				
var5	.33 **	.30 **	.74 **	.70 **			
	[.25, .41]	[.22, .38]	[.70, .78]	[.66, .75]			
var6	−.08	−.00	−.69 **	.59 **	−.09		
	[−.17, .01]	[−.09, .09]	[−.73, −.64]	[.53, .64]	[−.18, −.00]		
var7	.02	.03	.02	.07	.07	.02	
	[−.07, .10]	[−.06, .11]	[−.07, .11]	[−.02, .16]	[−.02, .16]	[−.07, .10]	
var8	.03	.11 *	.00	.02	.02	−.01	.76 **
	[−.06, .12]	[.02, .19]	[−.09, .09]	[−.07, .10]	[−.07, .11]	[−.10, .08]	[.72, .80]

Table 1 Example APA-formatted correlations table from MUFOS with Benjamini-Hochberg correction applied, values placed in the lower triangle and with 95% confidence intervals.

** $p < 0.01$, * $p < 0.05$.

and final step is to write unit tests for both the output dataframe (to ensure calculations are correct) and the formatted table.

EXPANDING MUFOS – ADDING A NEW MULTIPLE TESTS CORRECTION PROCEDURE

If a new multiple tests correction procedure needs to be implemented, the process is even simpler. There are three major changes that need to be done. The first is to update the user interface by mainly adding the new procedure as an option to the dropdown. The second is to update the Step 4 function (found in the *decision_funcs* file) which appends a column with adjusted *p* values to the output dataframe created in Step 3. Note that the new multiple tests correction procedure needs to be able to produce adjusted *p* values, not just correct the given alpha threshold. The third change is to write unit tests to ensure the new procedure works as expected.

EXPANDING MUFOS – ADDING A NEW FORMATTED TABLE OPTION

Adding a new style for formatting table for any path requires three major changes. The first is to update the user interface so that this option appears where necessary. The second is to create a new function within the path's file. For example, if a new style for a multiple regression table is needed when the input is raw data, then a new styling function needs to be created in the *main_funcs_raw_mr* file. Once this function is created, the Step 5 (saving data) decision function needs to be updated in order to execute the newly created styling function if the user selects that option. The third and final change is to write unit tests for the formatted table to ensure it comes out in the expected format.

CONTACT AND SUPPORT

Users have access to a *MUFOS Help Guide and Documentation* document located in the code repository. Although no official support is available, users are encouraged to report problems, ask questions, and provide feedback/suggestions either on GitHub or via email to the lead author.

NOTE

1 The probability of at least one false positive in 14 tests is 1 – (1 – 0.05) [14], or over 50%.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Nikolay Petrov *orcid.org/0000-0002-1305-0547*
School of Human Sciences, University of Greenwich, UK
Vasil Atanasov
Software Engineer, Manchester, UK
Dr Trevor Thompson *orcid.org/0000-0001-9880-782X*
School of Human Sciences, University of Greenwich, UK

REFERENCES

1. **Šidák Z.** Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*. 1967; 62(318): 626–633. DOI: https://doi.org/10.1080/01621459.1967.10482935
2. **Hochberg Y.** A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*. 1988; 75(4): 800–802. DOI: https://doi.org/10.1093/biomet/75.4.800
3. **Hommel G.** A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*. 1988; 75(2): 383–386. DOI: https://doi.org/10.1093/biomet/75.2.383
4. **Holm S.** A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*. 1979; 65–70.
5. **Benjamini Y, Hochberg Y.** Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*. 1995; 57(1): 289–300. DOI: https://doi.org/10.1111/j.2517-6161.1995.tb02031.x
6. **Benjamini Y, Yekutieli D.** The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*. 2001; 29(4): 1165–1188. DOI: https://doi.org/10.1214/aos/1013699998
7. **Benjamini Y, Krieger AM, Yekutieli D.** Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*. 2006; 93(3): 491–507. DOI: https://doi.org/10.1093/biomet/93.3.491
8. **Benjamini Y, Hochberg Y.** On the adaptive control of the false discovery rate in multiple testing with independent statistics. *Journal of educational and Behavioral Statistics*. 2000; 25(1): 60–83. DOI: https://doi.org/10.3102/10769986025001060
9. **Bretz F, Hothorn T, Westfall P.** *Multiple Comparisons Using R*. Chapman and Hall/CRC; 2016. DOI: https://doi.org/10.1201/9781420010909
10. **Farcomeni A.** A review of modern multiple hypothesis testing, with particular attention to the false discovery proportion. *Statistical Methods in Medical Research*. 2008; 17(4): 347–388. DOI: https://doi.org/10.1177/0962280206079046
11. **Lee S, Lee DK.** What is the proper way to apply the multiple comparison test? *Korean Journal of Anesthesiology*. 2018; 71(5): 353–360. DOI: https://doi.org/10.4097/kja.d.18.00242
12. **Tamhane AC, Gou J.** Advances in p-Value Based Multiple Test Procedures. *Journal of Biopharmaceutical Statistics*. 2018; 28(1): 10–27. DOI: https://doi.org/10.1080/10543406.2017.1378666
13. **Wang D, Li Y, Wang X, Liu X, Fu B, Lin Y, Larsen L, Offen W.** Overview of multiple testing methodology and recent development in clinical trials. *Contemporary Clinical Trials*. 2015; 45: 13–20. DOI: https://doi.org/10.1016/j.cct.2015.07.014
14. **Armstrong RA.** When to use the Bonferroni correction. *Ophthalmic and Physiological Optics*. 2014; 34(5): 502–508. DOI: https://doi.org/10.1111/opo.12131
15. **Gyorffy B, Gyorffy A, Tulassay Z.** The problem of multiple testing and its solutions for genom-wide studies. 2005; 146(12): 559–563.
16. **Lakens D.** The 20% Statistician: Error Control in Exploratory ANOVA's: The How and the Why; 2016. URL http://daniellakens.blogspot.com/2016/01/error-control-in-exploratory-anovas-how.html. [Online; accessed 17-July-2020].
17. **Lakens D.** The 20% Statistician: Why you don't need to adjust your alpha level for all tests you'll do in your lifetime; 2016. URL http://daniellakens.blogspot.com/2016/02/why-you-dont-need-to-adjust-you-alpha.html. [Online; accessed 17-July-2020].
18. **Noble WS.** How does multiple testing correction work? *Nature biotechnology*. 2009; 27(12): 1135–1137. DOI: https://doi.org/10.1038/nbt1209-1135
19. **Streiner DL.** Best (but oft-forgotten) practices: the multiple problems of multiplicity—whether and how to correct for many statistical tests. *The American Journal of Clinical Nutrition*. 2015; 102(4): 721–728. DOI: https://doi.org/10.3945/ajcn.115.113548
20. **Bennett CM, Miller, MB, Wolford GL.** Neural correlates of interspecies perspective taking in the post-mortem Atlantic Salmon: An argument for multiple comparisons correction. *Neuroimage*. 2009; 47(Suppl 1): S125. DOI: https://doi.org/10.1016/S1053-8119(09)71202-9
21. **Austin PC, Mamdani MM, Juurlink DN, Hux JE.** Testing multiple statistical hypotheses resulted in spurious associations: a study of astrological signs and health. *Journal of clinical epidemiology*. 2006; 59(9): 964–969. DOI: https://doi.org/10.1016/j.jclinepi.2006.01.012
22. **Lesack K, Naugler C.** An open-source software program for performing Bonferroni and related corrections for multiple comparisons. *Journal of pathology informatics*. 2011; 2. DOI: https://doi.org/10.4103/2153-3539.91130
23. **MATLAB.** *9.7.0.1190202 (R2019b)*. The MathWorks Inc.; 2018.
24. **R Core Team.** *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing; 2020. URL https://www.R-project.org/.
25. **Van Rossum G, Drake FL.** *Python 3 Reference Manual*. CreateSpace; 2009.
26. **IBM Corp.** *IBM SPSS Statistics for Windows*; 2017. URL https://hadoop.apache.org.
27. **Microsoft Corporation.** *Microsoft Excel*; 2018. URL https://office.microsoft.com/excel.
28. **Baker M.** Scientific computing: Code alert. *Nature*. 2017; 541(7638): 563–565. DOI: https://doi.org/10.1038/nj7638-563a
29. **JASP Team.** *JASP (Version 0.13.1)[Computer software]*; 2020. URL https://jasp-stats.org/.
30. **American Psychological Association.** *Publication manual of the American Psychological Association*. American Psychological Association; 2020. DOI: https://doi.org/10.1037/0000165-000

31. **McKinney W, others.** Data structures for statistical computing in python. In: *Proceedings of the 9th Python in Science Conference*. Austin, TX; 2010. DOI: *https://doi.org/10.25080/Majora-92bf1922-00a*
32. **Oliphant TE.** *A guide to NumPy*. Trelgol Publishing USA; 2006.
33. **Seabold S, Perktold J.** Statsmodels: Econometric and statistical modeling with python. In: *9th Python in Science Conference*; 2010. DOI: *https://doi.org/10.25080/Majora-92bf1922-011*
34. **Van Rossum G.** *The Python Library Reference, release 3.8.2*. Python Software Foundation; 2020.
35. n.d. Python-docx. URL *https://python-docx.readthedocs.io/en/latest/#*. [Online; accessed 13-August-2020].
36. **Gazoni E, Clark.** openpyxl – A Python library to read/write Excel 2010 xlsx/xlsm files, version 3.0.4; 2020. URL *https://openpyxl.readthedocs.io/en/stable/index.html*. [Online; accessed 13-August-2020].
37. **Reitz K.** requests: Python HTTP for Humans; 2020. URL *http://python-requests.org/*.
38. **Cortesi D.** PyInstaller Manual, Release 4.0; 2020. URL *https://pyinstaller.readthedocs.io/en/stable/*.
39. **NSIS Team.** Nullsoft Scriptable Install System (NSIS), version 3.06.1; 2020. URL *http://nsis.sourceforge.net/*.

TO CITE THIS ARTICLE:
Petrov N, Atanasov V, Thompson T 2022 Open-Source MUltiple Tests Corrections and FOrmatted Tables Software (MUFOS). *Journal of Open Research Software*, 10: 5. DOI: *https://doi.org/10.5334/jors.350*

Submitted: 29 September 2020 Accepted: 03 March 2022 Published: 17 March 2022