**Scaled Conversions: Developers’ Guide**

**Overview**

Scaled Conversions converts other statistical test scores into scaled scores. It was developed for use in neuropsychology practices to make it easier for the neuropsychologists to communicate the results of their evaluations with patients or other medical professionals.

Patients are referred for neuropsychological assessment when a neurological condition impacts or may impact cognitive function. The neuropsychologist uses a battery of multiple tests to assess the current level of functioning in each cognitive domain. Broadly, these domains include Language, Spatial, Memory, Attention, Executive Function. Individual tests come from a variety of sources and utilize different statistical tests to reflect patient performance. Scores are most commonly reported using standard scores, t-scores, z-scores, or scaled scores.

Scaled Conversions automatically converts each score entered by the user into a scaled score, allowing for direct comparisons between all the tests within each domain. The program produces an Excel workbook with a separate worksheet for each domain. Each worksheet lists the tests administered, the originally reported score type, the patient reported score, and the converted scaled score.

**Running Scaled Conversions**

Scaled Conversions can be accessed through GitHub: <https://github.com/JBiars/ScaledConversions>. Click Clone to download the zip file.

Originally developed using Visual Studio Code 2, this remains the recommended application for running Scaled Conversions. However, it can also be run using the command line terminal. It will require Python 3.7 or higher and Microsoft Excel. No additional prerequisites are required.

Scaled Conversions is actually two separate scripts: one to create a .p file and one to create the final .xlsx file. Start by running define\_tests.py. This will create the file test\_lists.p that contains the dictionaries of test names and original statistical score types. This file will not include the patient’s scores. After the test\_lists.p file is created, run convert\_scores.py to enter the patient’s original scores and produce the file ScoreConversions.xlsx.

**User Interaction**

Part 1: define\_tests.py

The user begins by running define\_tests.py. The script prompts the user to indicate whether they want to add a new test to a specific domain by pressing 1 for yes or 0 for no. From there, the user inputs the name of the test, stored as a string. Then, after displaying a dictionary of possible score types, the user is prompted to select the type of score type for the previously entered test by entering its assigned number, stored as an integer. Both the test name string and the score type integer are then stored in a dictionary. After each test, the user is again prompted to indicate whether they wish to continue adding additional tests. This process repeats until the user enters 0 to indicate that they are ready to move on to the next domain.

After the user has entered tests in each domain, they are prompted to indicate if there are any additional tests that need to be added, again pressing 1 for yes or 0 for no. If they press 1, a dictionary of the domains will appear and they will select the domain they wish to add more tests to by entering its assigned number, stored as an integer. Then the process of adding a test and score type to that domain’s dictionary will repeat. This process will continue until the user presses 0 to indicate that no more tests need to be added.

Once the user presses 0 for the last time, define\_tests.py will combine all of the domain dictionaries into a single nested dictionary (d) stored in a file named test\_lists.p using the publicly available pickle module.

Part 2: convert\_scores.py

The convert\_scores.py script starts by importing the pickle module and accessing the test\_lists.p file created define\_tests.py. Each dictionary with the nested dictionary (d) is separated into two lists: [domain]\_t with the names of the tests and [domain]\_s with the number codes assigned to each score type.

The user is prompted to enter the score to be converted for each test in the [domain]\_t list and those scores are saved as floats in a third [domain]\_scores list. After a score has been entered for each test, convert\_scores.py performs necessary calculations to obtain a scaled score, based on the number code stored in the [domain]\_s list. The resulting scaled score is saved in a final list, [domain]\_c.

After all the scores have been entered, converted, and stored each domain has four associated lists: \_t, \_s, \_scores, and \_c. The script then imports the publicly available xlsxwriter module and creates a workbook, ScoreConversions.xlsx with a separate worksheet labeled for each domain. The worksheets contain labels in the first four columns: Tests Administered, Score Type, Patient Score, and Converted Scaled Score. Under Tests Administered, the contents of the \_t list, the test name strings, are printed in subsequent rows. Under Score Type, the names of the score types are printed in subsequent rows by using the integers stored in the \_s list as keys to call up the corresponding values in the score\_types dictionary. Under Patient Score, the contents of the \_scores list are printed in subsequent rows. And under Converted Scaled Score, the contents of the \_c list are printed in subsequent rows.

**Known Issues**

There is currently no way for users to delete a test, score type, or patient score from a dictionary or list once it has been entered.

**Ongoing Development**

Both scripts in the Scaled Conversions program contain many repetitions that could be more efficiently implemented as functions. This would allow the five larger domains to be divided into more specific sub-domains.

Given that the target audience may not be familiar with Python or Command-Line Interfaces, it would also be advisable to create a GUI.