Instructions for using TestScorer 1.0
Manel Salamero

manelsalamero@gmail.com

Revision date: 2013.04.08

1. Introduction

The TestScorer package allows users to score different kind of tests, questionnaires or clinical scales. The user has to previously define the scoring instruction. This task is also facilitated by the package. All the functions are launched from a user-friendly graphical user interface (GUI). This document provides an overview of the usage of the TestScorer package.

To install R on your computer (free under GNU General Public License), go to the home website of R: http://www.r-project.org/. And do the following:
- Click "download packages CRAN" in the left bar
- Choose the download site nearest to you
- Choose the version for your operating system
- Click "base"
- Install the program when downloaded.

Open R clicking on the icon which was created on your screen. From the menu bar choose the option "Packages" and then "Install package(s)". Finally, locate "TestScorer" in the list and select it. These operations need only be done for the first time. You can also load the package from a local file: Packages → Install package(s) from a local zip file... .

To load the package enter at the R prompt:

```
> library(TestScorer)
```

And then:

```
> TestScorerGUI()
```

The program will ask for a directory where to score questionnaires Tree example tests and this document are copied to the directory. A batch file with name "TestScorer.cmd" is created in this directory only for Windows system. In subsequent uses, clicking over this file will open R and launch the TestScorer. Also you can create a direct link pointing to "TestScorer.cmd", and drag it to the desktop or other more convenient directory.

The GUI has two main functions: entering items and scoring an existing test (see section 2) or managing the catalog of tests including deleting or defining new ones (see section 3). A button allows to quit from TestScorer" and from R. If you quit R from the console

a window would emerge, asking if you want “Save the workspace image”. Answer “No”.

2. Entering items and scoring a test

2.1 Choosing a test

Note that all gray windows are non editable, they show useful information. The first step is choosing the test whose items you want to introduce and score. In the upper left corner you will find a window labeled “Which test would you like to score”. Find the test name using the scrolling bar if necessary, and click on it. The test’s name will be highlighted. Now you can see the characteristics of this test on the window labeled “Test details”. Beyond the name of the test, its authors and some cooment, you will find the number of items, the valid answers, and the codes for missing answers.

2.2 Entering the identification data of the subject

In the middle part of the window you can enter some data on the subject. These are optional fields which admit any characters, except for sex which is recorded through radio buttons.

2.3 Entering the items

On the upper right corner of the main window you will find the area for entering the items which is labeled “Entry items window”. The first line shows the number of the item to be entered. You have to put the cursor in the white rectangle located on the left. *Do not move the cursor from this position while entering items*, all the necessary actions are done through the keyboard. The numerical keypad is the most convenient way to introduce the answers, but the numerical keys and the spacebar of the main keyboard can also be used.

Enter the answer of the items using the keyboard. Only the valid and missing characters shown in the “Test details” window are allowed. If you enter an invalid character a pop-up window appears highlighting the error and the program is blocked until the error window is closed.

Every time you enter a valid answer the counter of items will increase in one unit until all the items are entered. The progression is reflected in two ways. First, the “Item to enter” is updated. Second, in the lower part of the window will appear the answers just introduced and an asterisk (*) showing the position of the next answer to be entered (see figure 1). As any gray window of the GUI, this one is non editable. Every line shows ten items, corresponding to the numbers shown in the right margin, and the upper margin indicates the position in the group of ten.

The window allows the representation of 100 items, but the ones which exceed the length of the test are crossed out with an equal symbol (=). Had the test more than one hundred items, the window would be refreshed showing the next hundred automatically.

In case of error, you can move through the answers using the arrow keys and introduce the correction for the appropriate items. Right and left arrow move to the previous or posterior answer respectively. The up and down arrows jump to the tenth previous or following answer. The use of the arrows updates the number of the new answer to be introduced and the position of the asterisk indicator automatically. If you try to go to an

answer beyond the limits of the test a pop-up window will alert you of the invalid choice.

2.4 Scoring the test

When you have entered all the answers press the "Score" button to get the results which appear on the R's main window. From the R's menu bar you can print or save the results.

2.5 Recording scores and answers to a file

It is possible to record the scores and items of the tests in a file for further statistical analysis. To do so, click on the button labeled "Click here to choose a file for saving the results". This opens a system window for choosing or creating a file for recording. If the file already exists the new information will be appended. The chosen file will be showed in the window labeled "Where would you like to save the scores?". Some tests are carried out to ensure that the old file structure is compatible with the new data, but the user must be careful and take the necessary precautions themselves. The radio buttons beneath the "Would you like to save the items" label allow recording the answers (as introduced through the keyboard without further processing) in addition to the scores.

The data are written to an ascii text file with semicolons between fields as delimiters. These kinds of file are easily imported by almost any program and also by R.

2.6 Other buttons

At the bottom of the screen you will find different buttons. "Score" is for scoring as explained previously. "Exit Scorer & R" closes both the TestScorer window and the R session. "Clean Items", cleans the items but maintains the identification information of the subject. So you can introduce and score another test of the same subject without reintroducing his or her data. "Clean all", cleans both the items and the identification data. "Test manager" is for creating new test score instruction or deleting the existent. Its functions will be explained in section 3. "Help" shows a brief summary of the information contained in this document.

3. Test manager

3.1 Deleting a test

Clicking the "Test Manager" button gives you access to two functions: deleting an existing test and creating a new one. If you choose to delete a test, the test is not actually deleted but the file extension is changed from ".r" to ".bak", giving you the possibility to recover it later.

3.2 Creating a new test

This option opens a menu for defining the general characteristics of the test. In this window you should introduce the information of the test. Any entry, unless otherwise specified, should be filled in. When the "Ok" button is pressed the program checks the validity of the information and a pop-up window is opened in case of errors. When closing this window you can correct the invalid entries.

Next a window for each scale opens. The required information should be entered and is checked for errors when pressing the “Ok” button. This process is repeated for each scale of the test.

Using this information the program generates a script which is written in a file in the working directory. The catalog test is automatically updated and you can begin to score the new test.

4 Editing the script manually to enhance the scoring capacities

4.1 The general structure of TestScorer

The scoring of tests usually consists of computing the scores of each scale by summing or computing the mean of their items. Sometimes the raw scores are transformed to T-scores according to the normative mean and standard deviation for each sex. Other times, the transformation is done through a table of equivalences. The instructions for these computations are automatically generated when a new test is defined through the GUI. Nevertheless, sometimes the test requires other data manipulations not performed by the basic script and in these cases it is necessary to edit the instruction manually.

Only a basic knowledge of R language is needed to adapt the script to the scoring peculiarities demanded by a test. To do this, it is worth to knowing the internal scoring process. When the GUI is launched, the working directory is scanned to detect any script for scoring tests. These scripts are identified when their name begins with the prefix “TST_”. The characteristics of the tests are read from these scripts, especially the number of items and the valid answers.

The GUI is governed by a main function which takes care of the introduction of items and the identifying information of the subject. During the introduction of the answers, the function checks the validity of the pressed keys. When the “Score” button is clicked, the main function reads the script for scoring this test and launches the scoring function. It sends (as parameters) a character vector with the answers and the data of the subject. These last parameters are used to control the transformation of the raw scores if required (e.g.: transformation is different for each sex).

When the scoring process is finished, the scoring function returns a list to the main function with three (optional) elements: “results.lst”, “results.df”, and “results.scores”. The “results.lst” is a list with character strings. The “results.df” is a data frame with the scores of the test’s scales. It can comprise the acronym and name of the scale, the raw score and a graphic representation of the score (this two last elements are optional). Finally, “results.scores” is a data frame with only one row with the obtained scores. The main function prints the “results.lst” and the “results.df” to the R console and, if required, writes the “results.scores” to an external ascii data file.

4.2 Editing the script

Creating a new test means creating an R script with the necessary commands defining the entering template of answers and the scoring procedure. The automatically generated script (section 3) is saved in the chosen directory, appending the test acronym to the prefix “TST_” with the usual R extension type “.r”. It can be edited using any text editor. The whole process is illustrated bellow through an example.

Since comments are included in the script, it is easy to follow the scoring process and, if necessary, include modification to accommodate the atypical demands of some tests. As an example, see the three public domain test which scoring script are copied to the installation directory of the TestScorer.

The first one, the "Multidimensional Health Locus of Control" (MHLC, http://www.nursing.vanderbilt.edu/faculty/kwallston/mhlcscales.htm), shows a simple raw scoring. In this case, the score is the mean of the answers corresponding to each of the three scales. No additional code was added to the script created by the program.

The "Depression, Anxiety and Stress Scores" (DASS, http://www2.psy.unsw.edu.au/dass/), includes a percentil transformation which is the same for both sexes. Some score conversions are plain percentile while other are ranges. Any character, no only numbers, is allowed in the conversion which gives great flexibility to the procedure. However, if characters are included, it prevents using this transformation for generating a graphical representation, since the program has no way for computing the position of the score in a continuum.

The "Revised Adult Attachment Scale" (RAAS, www.openpsychassessment.org/wp-content/uploads/2011/06/AdultAttachmentScale.pdf) shows a T'scores conversion using the mean and standard deviation which are different for males and females. Also a schematically graphical representation of the profile is included. All these instructions were created through the menu. The resulting script was further edited to enhance the results.

In the following paragraphs the structure of the script is commented to facilitate the task of manually modifying the instructions to the interested users (see appendix). The automatic script is comprised of two main parts: a list with information about the test (lines 5 to 13) and a main function (lines 15 to 137

The list "testChar" includes the characteristics of the test. As mentioned before, when the GUI is launched, it looks in the directory for all the files whose name begins with "TST_". The program reads the "testChar" list to form the catalog which is displayed in the main window. In this way the program knows which tests can be corrected, and for each test the number of items, the valid answers and the characters defined as missing. This information is used to construct the entry items window and scans the validity of the answers introduced by the user.

When the "Score" button is pressed the program loads the script corresponding to the test being considered and reads and executes the main function "scoring.fun". The main program sends to it a character string with the answers and the identification characteristics of the subject. This last information is useful in transforming scores (e.g.; differences between sexes). Lines 17 to 23 convert the characters answers to numbers and, if appropriate, invert the answers scoring of selected items. Line 24 initializes a data frame for storing the results.

If requested a function for converting raw scores (in this case into T'scores, lines 26 to 31) and another for producing a simple graphic display of the T-scores (lines 33 to 43) are added.

Next you will find the scoring commands for each scale (lines 5 to 76). In each block the number of missing and the raw score of the scale is calculated. TestScorer allows to compute the raw scores as a sum of items with or without prorating missing and, also, as a mean of items. If required, raw scores are transformed. The options are transforming to T'scores using the mean and standard deviation or through a table and other transformations (e.g., percentiles) using a table. All this information is added to the results data frame (lines 63 to 65).

In lines 80 to 83 a one row data frame with the scores is created. This data frame is used to append to a text file intended for further statistical analysis. The data frame will be passed to the main program which manages these tasks.

In lines 117 to 132 a list is constructed which initially only contains the number of missing answers. This list, as described below, can be extended to add any information which can not be transferred through a data frame.

Finally, line 136 brings the results back to the main program which will print them to the R console and write to a file if required.

All the previous instructions were written by the GUI and were later edited to include some peculiarities specific to the RAAS test. Lines 87 to 113 compute new indexes ("CD" and "style") combining the scores of two of the original scales. It is worth noting that one of these indexes was incorporated to the results data frame ("CD" in lines 89 to 91), while the other was transmitted via the results list (variable "style" in line 127). The graphical capabilities of R also allowed creating a plot (lines 102 to 112). Finally, lines 117 to 130 modify the results list including textual information and the variable "style" computed before.

Appendix

```
# RAAS scale scoring commands
# Creation date: 2013-04-02
# --------------

testChar <- list(
  acronym="RAAS",
  name="Revised Adult Attachment Scale",
  ref="Collins, 1996",
  n.items=18,
  valid=c(1, 2, 3, 4, 5),
  miss=c(0),
  comm="Public domain: www.openpsychassessment.org/wp-content/uploads/2011/06/AdultAttachmentScale.pdf"
) # end testChar

scoring.fun <- function(answers, sex, age=0, id, date.test, comm)
# "answer" is a *character* vector as introduced through the keyboard.
{
  answers <- as.numeric(answers) # transform to numeric for easier scoring
  answers[answers %in% c(0)] <- NA # missing characters to NA
  blanks <- sum(is.na(answers)) # compute number of missings
  pcnt.blanks <- round((blanks / 18) * 100) # compute % of missings
  reversed.items=c(2, 7, 8, 13, 16, 17, 18)
  answers[reversed.items] <- (5 + 1) - answers[reversed.items] # reverse items
  results <- data.frame(NULL) # Null data frame for results

  toT <- function(raw.score, mean, sd) # compute T score
  {
    T.score <- round(((raw.score - mean) / sd) * 10 + 50)
    return(T.score)
  } # end toT

  makeGraph <- function(T.score) # make graph
  {
    if (!is.na(T.score)) # avoids error
    {
      graph  <- "|    :    :    |    :    |    :    |    :    :    |"
      if (T.score < 0) T.score <- 0
        else if (T.score > 100) T.score <- 100
      position <- round((T.score/2)+1)
      graph <- paste(substr(graph, 1, position-1), substr(graph, position + 1, nchar(graph)), sep="o")
    }
    else graph <- NA
  } # end makeGraph

  # C scale scoring commands
  # --------------
  results[1, "Acronym"] <- "C" # acronym
  results[1, "Scale"] <- "Close" # name of the scale
  items <- c(1, 6, 8, 12, 13, 17) # items making up the scale
  results[1, "Miss"] <- sum(is.na(answers[items])) # number of missings
  results[1, "Raw"] <- round(mean(answers[items], na.rm=TRUE), 2) # raw score (mean answered items
  if (sex=="male") results[1, "T"] <- toT(results[1, "Raw"], 21.54, 5.14) # compute T score
    else results[1, "T"] <- toT(results[1, "Raw"], 21.92, 5.19)
```

```
results[1, "Graph"] <- makeGraph(results[1, "T"]) # make the graph

# D scale scoring commands
# --------------
results[2, "Acronym"] <- "D" # acronym
results[2, "Scale"] <- "Dependent" # name of the scale
items <- c(2, 5, 7, 14, 16, 18) # items making up the scale
results[2, "Miss"] <- sum(is.na(answers[items])) # number of missings
results[2, "Raw"] <- round(mean(answers[items], na.rm=TRUE), 2) # raw
score (mean answered items
if (sex=="male") results[2, "T"] <- toT(results[2, "Raw"], 20.59, 5) #
compute T score
  else results[2, "T"] <- toT(results[2, "Raw"], 19.51, 5.14)
results[2, "Graph"] <- makeGraph(results[2, "T"]) # make the graph

# A scale scoring commands
# --------------
results[3, "Acronym"] <- "A" # acronym
results[3, "Scale"] <- "Anxiety" # name of the scale
items <- c(3, 4, 9, 10, 11, 15) # items making up the scale
results[3, "Miss"] <- sum(is.na(answers[items])) # number of missings
results[3, "Raw"] <- round(mean(answers[items], na.rm=TRUE), 2) # raw
score (mean answered items
if (sex=="male") results[3, "T"] <- toT(results[3, "Raw"], 13.86,
5.36) # compute T score
  else results[3, "T"] <- toT(results[3, "Raw"], 15.7, 6.05)
results[3, "Graph"] <- makeGraph(results[3, "T"]) # make the graph

# One row data frame with scores, for writing scores into a data file
# --------------------
results.scores <- data.frame(t(c(blanks, results[, "Raw"],
results[,"T"])))
names(results.scores) <- c("blanks", paste(results[["Acronym"]],
"raw", sep="_"), paste(results[["Acronym"]], "T", sep="_"))

# Ruler for graph column name
# --------------------
names(results)[6] <- "0    10   20   30   40   50   60   70   80   90
100"

# ===== ADDITIONAL CODE INSERTED MANUALLY
# Combine C & D scales
CD <- round(mean(c(results[1, 'Raw'], results[2, 'Raw'])), 2)
results[4, ] <- c('', '','','','','') # blank row to improve readabi-
lity
results[5, ] <- c('CD', 'Close/Dependent', '', CD, '', '') # no data
for computing T score

# Attachment style assignement
style <- 'Located at midline'
if (is.na(CD)) style <- 'Not evaluable' # if all answers are missings
else if (CD > 3 & results[3, 'Raw'] < 3) style <- 'Secure'
else if (CD > 3 & results[3, 'Raw'] > 3) style <- 'Preoccupied'
else if (CD < 3 & results[3, 'Raw'] < 3) style <- 'Dismissing'
else if (CD < 3 & results[3, 'Raw'] > 3) style <- 'Fearful'
else style <- 'Not classificable'

# Show style as a plot
windows(title="Attachment style")
plot(CD, results[3, 'Raw'], xlim=c(1,5), ylim=c(1,5), pch=3, cex=2,
col='blue', lwd=5,
```

```
105         xlab='Close/Dependent', ylab='Anxiety', main=paste(id,
    date.test),
106         font.sub=2, sub='Position of the subject is represented by a blue
    cross')
107   abline(v=3)
108   abline(h=3)
109   text(2, 2, labels='Dismissing', col='gray60', font=2, cex=2)
110   text(4, 2, labels='Secure', col='gray60', font=2, cex=2)
111   text(2, 4, labels='Fearful', col='gray60', font=2, cex=2)
112   text(4, 4, labels='Preocupied', col='gray60', font=2, cex=2)
113   # ===== END ADDITIONAL CODE INSERTED MANUALLY
114
115   # Output in form of list
116   # ------------------
117   results.lst <- list(paste("Total number of missings: ", blanks, " (",
    pcnt.blanks, "%)", sep=""),
118             # ===== ADDITIONAL CODE INSERTED MANUALLY
119                       "",
120                       "RAAS Collis, 1996. Public domain document downlo-
    adable from:",
121                       "http://www.openpsychassessment.org/wp-
    content/uploads/2011/06/",
122                       "AdultAttachmentScale.pdf, downloaded: 29 sept
    2012.",
123                       "",
124                       "According to the author, attach styles assigne-
    ment 'is quite exploratory...",
125                       "[use] with caution, and only in conjunction with
    the continuous measures.'",
126                       "",
127                       paste("Attach style:", style),
128                       "",
129                       "T scores computed using data from 414 USA college
    students reported",
130                       "by Ledley et al. J Psychopath Behav Assess 2006,
    28:33-40."
131             # ===== END ADDITIONAL CODE INSERTED MANUALLY
132                         )
133
134   # Return results
135   # ------------------
136   return(list(results.lst=results.lst, results.df=results, results.sco-
    res=results.scores))
137
138 } # end of scoring.fun
```