Instructions for installing and using TestScorer 1.4
Manel Salamero

manelsalamero@gmail.com

Revision date: 2013.11.29

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[1]. These operations need only be done for the first time.

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. When it is created for the first time, the necessary files are copied to this directory.

1.1. Content of the directory for scoring tests

The directory for scoring tests includes the following files: A "Profile", three example tests, help documentation, and, only for Windows system, a launcher of the program.

[1] This instruction can fail if the computer is connected through a proxy. In this case contact with the webmaster or follow this alternative procedure: From http://cran.r-project.org, click "Packages" at the left column. Choose "Table of available packages, sorted by name" and lock for "TestScorer". Download the appropriate version for your operating system. Now from R menu bar choose "Packages" and "Install package(s) from local zip files..."

A local profile with name ".Profile" is created which automates the launch of the GUI when R is opened in this folder. the profile includes these instructions:

```
# Profile to launch TestScorer
suppressPackageStartupMessages(require(TestScorer))
TestScorerGUI()
```

Three example tests, a help file and this document are copied to the folder ("TST_DASS.r", "TST_MHLC.r", "TST_RAAS.r", "Help.txt" and "TestScorerHelp. pdf"). When the user defines new tests (see section 3), they will be also recorded in this folder.

For Windows system only, a batch file with name "TestScorer.cmd" is also written in this directory. In subsequent uses, clicking over this file will open R and launch the TestScorer. You can create a direct link pointing to "TestScorer.cmd", and drag it to the desktop or other more convenient location. "TestScorer.cmd" contains only a Windows batch command line:

```
start "" Drive:/path/R/R-3.0.2/bin/x64/Rgui.exe --no-
restore --quiet
```

The drive and path to R are constructed according to the user installation and should be changed whenever a new version of R replaces the previous one. R is called with the options "no-restore" and "quiet" to begin a fresh session without printing the R's welcome messages.

1.2. Main functions of the GUI

The GUI has two main functions: entering items and scoring an existing test (see section 2) or managing the catalog of tests, which includes deleting or defining new ones (see section 3). A button allows quitting from TestScorer and returning to the console, from which you can exit R from the menu bar or typing "quit()". When logging out a window will 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 (figure 1). 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 comment, 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 respectively. 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 "Show (& write) results" button to show the results on the R console (and optionally write them to a file, see 2.5). From the R menu bar you can print or save these results. "Only write" scores the test without showing it on the console, but recording the results on a file as explained in the next paragraph.

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 "Change option". 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. Show (& write) results" and "Only write" are for scoring as explained previously. "Exit TestScorer" exits and closes the TestScorer. "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 (figure 2). 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 (figures 3 and 4). 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 Show (& write) results" or "Only record" buttons are 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 three examples with increasing complexity.

Since comments are included in the script, it is easy to follow the syntax and, if necessary, include modifications to accommodate the atypical demands of some tests. The tree scripts for scoring three public domain tests, which are copied to the installation directory of TestScorer, will be briefly commented in the following paragraphs.

The "Multidimensional Health Locus of Control" (MHLC, http://www.nursing.vanderbilt.edu/faculty/kwallston/mhlcscales.htm) is an example of a very simple scoring procedure. The script (TST_MHLC.r) shows a plain computing of raw scores. In this case, the score is the sum of the answers corresponding to each of the three scales. No additional code was added to the script created by the program.

A score conversion is shown for the questionnaire "Depression, Anxiety and Stress Scales" (DASS, http://www2.psy.unsw.edu.au/dass/), which includes a percentil transformation which is the same for both sexes (TST_DASS.r). In this case, some score conversions are plain percentiles while others are ranges. Any character, not only numbers, is allowed in the conversion table, 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. Only lines 93 to 97 were inserted manually to add textual information when displaying the results.

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 (TST_RAAS.r). Also a schematically graphical representation of the profile is included. The basic instructions were created through the menu and the resulting script was further edited to enhance the results.

In the following paragraphs the structure of this last script is commented to facilitate the interested users the task of manually modifying the instructions (see appendix). The automatic generated script is comprised of two main parts: a list with information about the test (lines 5 to 11) and a main function (lines 13 to 165.

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 scored, 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 while they are introduced by the user.

When the Show (& write) results" or "Only record" buttons are 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 16 to 22 convert the characters answers to numbers and, if appropriate, invert the answers scoring of selected items. Line 23 initializes a data frame for storing the results.

If requested a function for converting raw scores (in this case into T-scores, lines 25 to 28) and another for producing a simple graphic display of the T-scores (lines 30 to 46) are added.

Next you will find the scoring commands for each scale (lines 48 to 100). 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 items and, also, as a mean of items. If required, raw scores are transformed. In this case, the option is transforming to T-scores using the mean and standard deviation. Other option is transforming through a table. All this information is added to the results data frame (lines 55 to 64, for scale C).

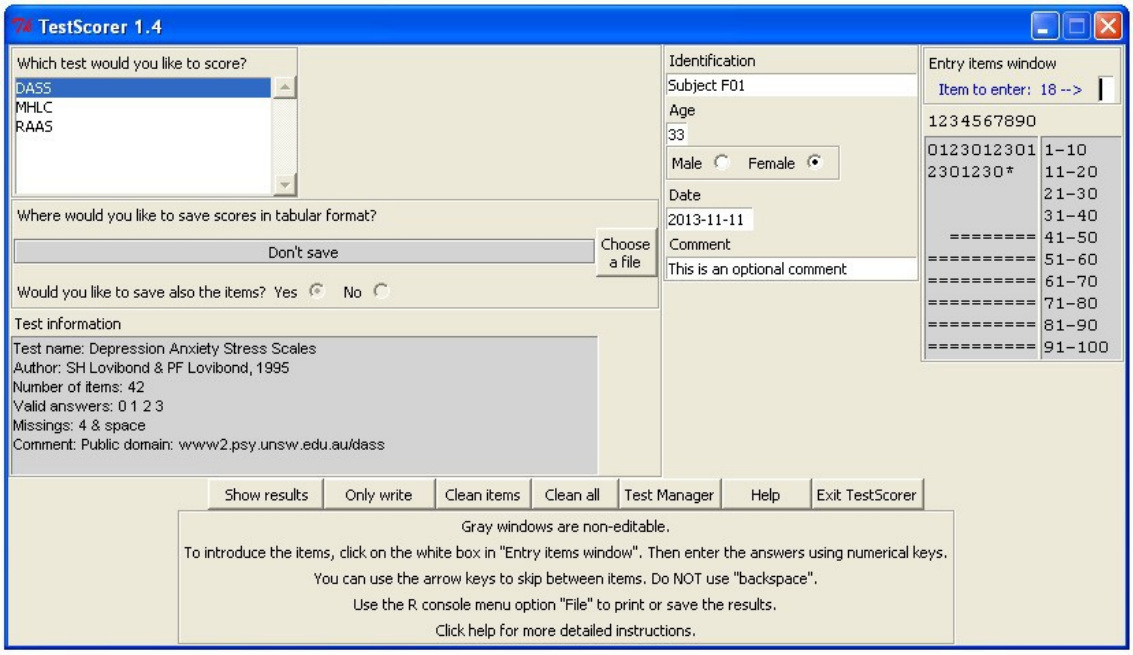
In lines 102 to 106 a one row data frame with the scores is created. This data frame is used to append to a text file intended for further statistical analysis. The data frame will be passed to the main program which manages this task.

In lines 139 to 157 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 the data frame.

Finally, lines 159 to 163 bring the results back to the main program which will print them to the R console or write it to a file as required.

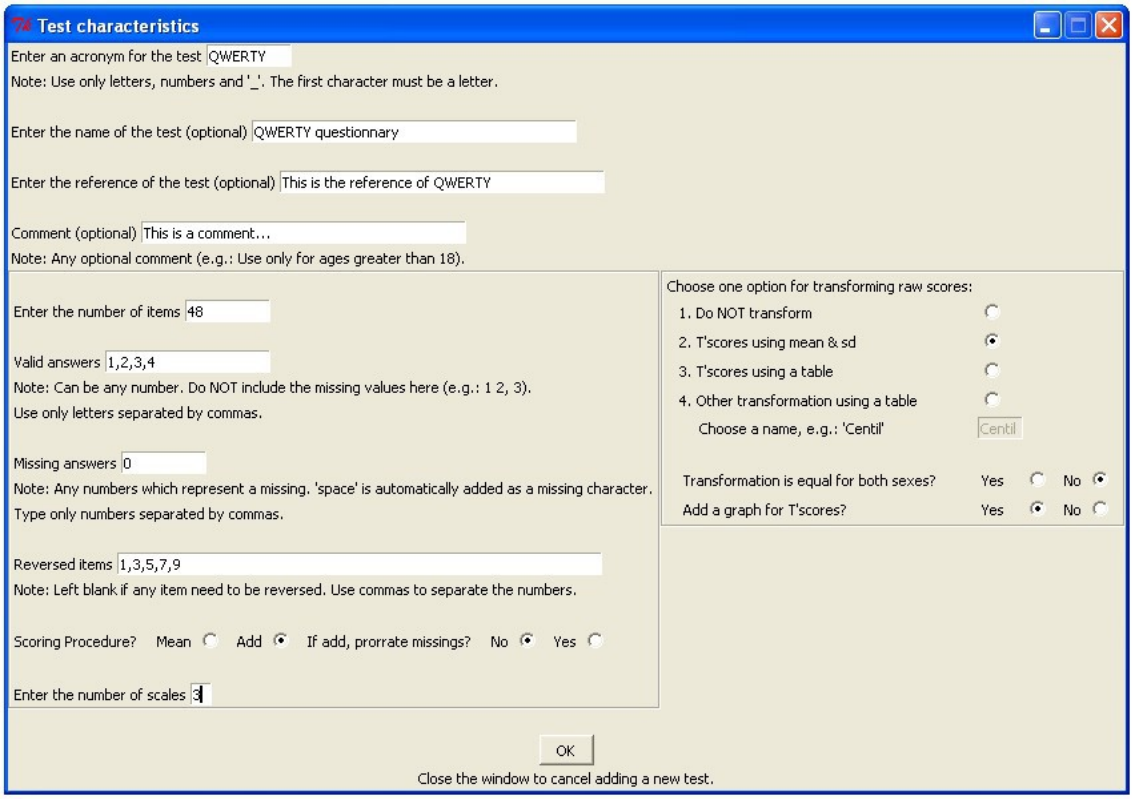
All the previous instructions were written by the GUI and were later edited to include some peculiarities specific to the RAAS test. Lines 112 to 137 compute new indexes ("CD" and "style") combining the scores of two of the original scales. It is worth noting that one of these indexes was incorporated to the results data frame ("CD" in lines 114 to 116), while the other was transmitted via the results list (variable "style" in line 152). The graphical capabilities of R also allowed creating a plot (lines 126 to 136). Finally, lines 147 to 156 modified the results list including textual information and the variable "style" computed before.

Figure 1. **TestScorer main window**



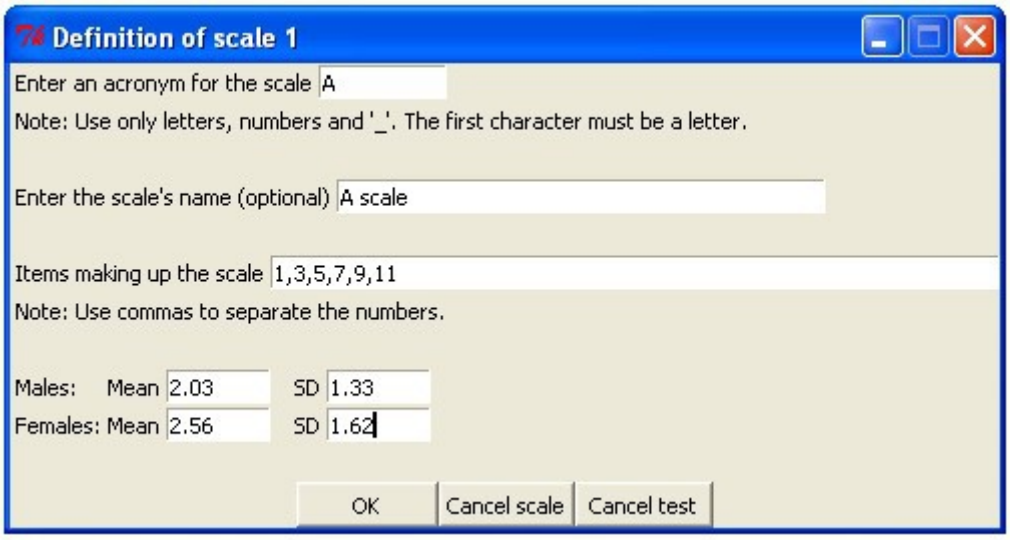
In this example 17 items were already introduced from a DASS questionnaire corresponding to subject “Subject F01”. In the “Entry items windows” you can see that the next item to enter is number 18. Bellow, the gray window shows the items already introduced and an asterisk in the position on which the next answer will be placed. This test has 42 items, so items greater than 42 are cross out in the gray window.

Figure 2. **Window for entering the characteristics of the test**



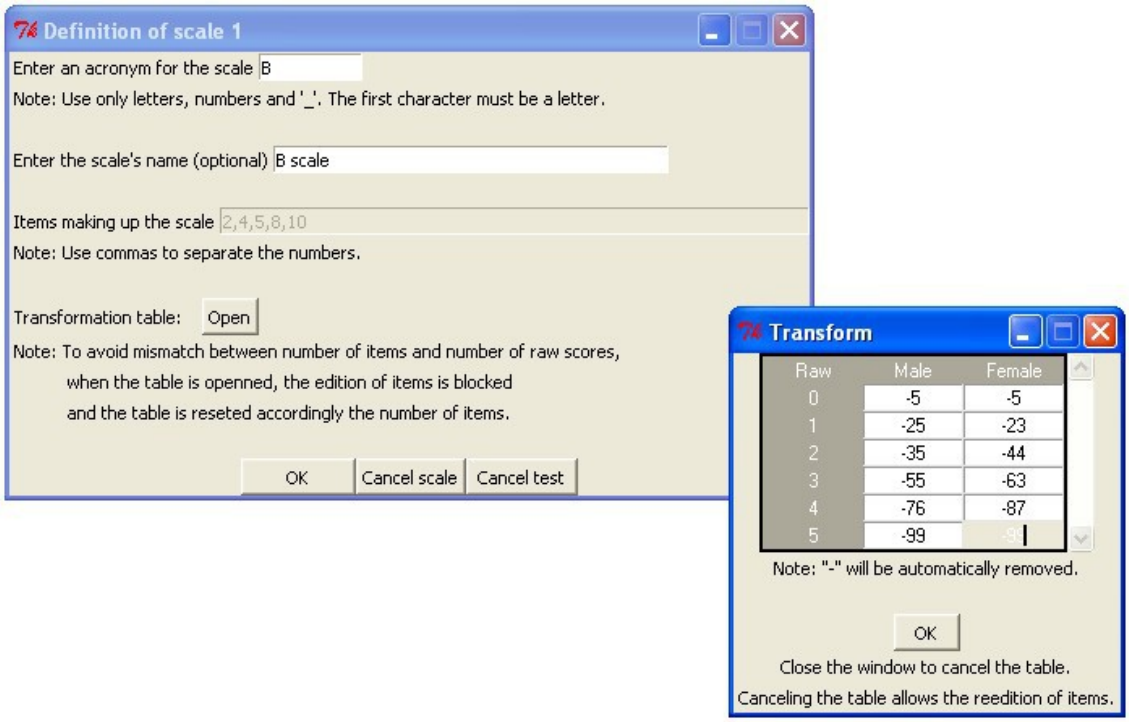
A test has been defined with 48 items. The possible values of the items are 1, 2, 3 or 4, while 0 indicates a missing. Three scales would be scored and the raw scores will be transformed to T-scores using the mean and standard deviations for each sex. A graphical display of the transformed scores is requested.

Figure 3. **Window for defining a scale with T-scores transformation**



A scale named "A" is defined. The items building up the scale are mandatory to compute de scale's raw score. The mean and standard deviation for each sex will be used to transform the raw scores into T-scores, as requested in the previous window.

Figure 4. **Window for defining a scale with percentiles transformation**



A scale named “B” is defined. Typing the items building up the scale are mandatory before opening the transformation window. The table allows the introduction of the corresponding percentiles for each raw score.

Appendix

```
001 # RAAS scale scoring script
002 # Creation date: 2013-10-09
003 # --------------
004
005 testChar <- list(acronym = "RAAS",
006                   name = "Revised Adult Attachment Scale",
007                   ref = "Collins, 1996",
008                   n.items = 18,
009                   valid = c(1, 2, 3, 4, 5),
010                   miss = c(0),
011                   comm = "Public domain: www.openpsychassessment.org/wp-content/uploads/2011/06/AdultAtta-
chmentScale.pdf")
012
013 scoring.fun <- function(answers, sex, age, id, date.test, comm) {
014   # "answer" is a *character* vector as introduced through the keyboard.
015   # "sex", "age", "id", "date.test" & "comm" used if appropiate.
016   answers <- as.numeric(answers) # transform to numeric for easier scoring
017   answers[answers %in% c(0)] <- NA # missing characters to NA
018   blanks <- sum(is.na(answers)) # compute number of missings
019   pcnt.blanks <- round((blanks / 18) * 100) # compute % of missings
020   # Items which should be reversed
021   reversed.items=c(2, 7, 8, 13, 16, 17, 18)
022   answers[reversed.items] <- (5 + 1) - answers[reversed.items]
023   results <- data.frame(NULL) # Null data frame for results
024
025   toT <- function(raw.score, mean, sd) {  # compute T score
026     T.score <- round(((raw.score - mean) / sd) * 10 + 50)
027     return(T.score)
028   } # end toT
029
030   makeGraph <- function(T.score) {  # make a graph
031     template  <- "|    :    :    |    :    |    :    |    :    :    |"
032     options(warn=-1)
033     T.score <- as.integer(T.score)  # can be a numerical string
034     options(warn=0)
035     if (!is.na(T.score)) {
036       if (T.score < 0) T.score <- 0
037         else if (T.score > 100) T.score <- 100
038       position <- round((T.score / 2) + 1)
039       graph <- paste(substr(template, 1, position-1),
040                      substr(template, position + 1, nchar(template)),
041                      sep="o")
042     } else {
043       graph <- "Not graphicable"
044     }
045     return(graph)
046   } # end makeGraph
047
048   # C scale scoring commands
049   # --------------
050   results[1, "Acronym"] <- "C" # acronym
051   results[1, "Scale"] <- "Close" # name of the scale
052   # Items making up the scale
053   items <- c(1, 6, 8, 12, 13, 17)
054   results[1, "Miss"] <- sum(is.na(answers[items])) # number of missings
055   if (results[1, "Miss"] == length(items)) {  # all answers are missings
056     results[1, "Raw"] <- NA
057     results[1, "T"] <- NA
058     results[1, "Graph"] <- "All missings"
059   } else {
060     results[1, "Raw"] <- sum(answers[items], na.rm=TRUE) # sum answered items
061     if (sex=="Male") results[1, "T"] <- toT(results[1, "Raw"], 3.59, 0.87) # compute T score
062       else results[1, "T"] <- toT(results[1, "Raw"], 3.65, 0.87)
063     results[1, "Graph"] <- makeGraph(results[1, "T"]) # make the graph
064   }
065
066   # D scale scoring commands
067   # --------------
068   results[2, "Acronym"] <- "D" # acronym
069   results[2, "Scale"] <- "Dependent" # name of the scale
070   # Items making up the scale
071   items <- c(2, 5, 7, 14, 16, 18)
072   results[2, "Miss"] <- sum(is.na(answers[items])) # number of missings
073   if (results[2, "Miss"] == length(items)) {  # all answers are missings
```

```
074     results[2, "Raw"] <- NA
075     results[2, "T"] <- NA
076     results[2, "Graph"] <- "All missings"
077   } else {
078     results[2, "Raw"] <- sum(answers[items], na.rm=TRUE) # sum answered items
079     if (sex=="Male") results[2, "T"] <- toT(results[2, "Raw"], 3.43, 0.83) # compute T score
080       else results[2, "T"] <- toT(results[2, "Raw"], 3.25, 0.86)
081     results[2, "Graph"] <- makeGraph(results[2, "T"]) # make the graph
082   }
083
084   # A scale scoring commands
085   # --------------
086   results[3, "Acronym"] <- "A" # acronym
087   results[3, "Scale"] <- "Anxiety" # name of the scale
088   # Items making up the scale
089   items <- c(3, 4, 9, 10, 11, 15)
090   results[3, "Miss"] <- sum(is.na(answers[items])) # number of missings
091   if (results[3, "Miss"] == length(items)) {  # all answers are missings
092     results[3, "Raw"] <- NA
093     results[3, "T"] <- NA
094     results[3, "Graph"] <- "All missings"
095   } else {
096     results[3, "Raw"] <- sum(answers[items], na.rm=TRUE) # sum answered items
097     if (sex=="Male") results[3, "T"] <- toT(results[3, "Raw"], 2.31, 0.89) # compute T score
098       else results[3, "T"] <- toT(results[3, "Raw"], 2.62, 1.01)
099     results[3, "Graph"] <- makeGraph(results[3, "T"]) # make the graph
100   }
101
102   # Vector for writing scores to a file
103   # --------------------
104   results.scores <- unlist(results[-c(1, 2)])  # not Acronym & Scale columns
105   names <- paste(results$Acronym, names(results.scores), sep=".")
106   names(results.scores) <- sub("[0-9]+$", "", names)  # delete ending numbers
107
108   # Ruler for graph column
109   # --------------------
110   names(results)[6] <- "0    10   20   30   40   50   60   70   80   90  100"
111
112   # ===== ADDITIONAL CODE INSERTED MANUALLY
113   # Combine C & D scales
114   CD <- round(mean(c(results[1, 'Raw'], results[2, 'Raw'])), 2)
115   results[4, ] <- c("", "", "", "", "", "") # blank row to improve readability
116   results[5, ] <- c("CD", "Close/Dependent", "", CD, "", "") # no data for T score
117
118   # Attachment style assignement
119   if (is.na(CD)) style <- 'Not evaluable' # if all answers are missings
120   else if (CD > 3 & results[3, 'Raw'] < 3) style <- 'Secure'
121   else if (CD > 3 & results[3, 'Raw'] > 3) style <- 'Preoccupied'
122   else if (CD < 3 & results[3, 'Raw'] < 3) style <- 'Dismissing'
123   else if (CD < 3 & results[3, 'Raw'] > 3) style <- 'Fearful'
124   else style <- 'Not classificable'
125
126   # Show style as a plot
127   windows(title="Attachment style")
128   plot(CD, results[3, 'Raw'], xlim=c(1,5), ylim=c(1,5), pch=3, cex=2, col='blue', lwd=5,
129        xlab='Close/Dependent', ylab='Anxiety', main=paste(id, date.test),
130        font.sub=2, sub='The position of the subject is represented by a blue cross')
131   abline(v=3)
132   abline(h=3)
133   text(2, 2, labels='Dismissing', col='gray60', font=2, cex=2)
134   text(4, 2, labels='Secure', col='gray60', font=2, cex=2)
135   text(2, 4, labels='Fearful', col='gray60', font=2, cex=2)
136   text(4, 4, labels='Preocupied', col='gray60', font=2, cex=2)
137   # ===== END ADDITIONAL CODE INSERTED MANUALLY
138
139   # Output in form of list
140   # ------------------
141   results.lst <- list(paste("Total number of missings: ",
142                             blanks,
143                             " (",
144                             pcnt.blanks,
145                             "%)",
146                             sep=""),
147                             # ===== ADDITIONAL CODE INSERTED MANUALLY
148                             "",
149                             "According to the author, attach styles assignement 'is quite
exploratory...",
```

```
150                             "[use] with caution, and only in conjunction with the continuous measures.'",
151                             "",
152                             paste("Attach style:", style),
153                             "",
154                             "T scores computed using mean and standard deviation from 414 USA college
students,",
155                             "reported by Ledley et al. J Psychopath Behav Assess 2006, 28:33-40."
156                             # ===== END ADDITIONAL CODE INSERTED MANUALLY
157                         )
158
159   # Return results
160   # ------------------
161   return(list(results.lst = results.lst,
162               results.df = results,
163               results.scores = results.scores))
164
165 } # end of scoring.fun
```