Using R to score personality scales*

William Revelle
Northwestern University

February 27, 2013

Contents

Abstract

The *psych* package (Revelle, 2013) was developed to perform most basic psychometric functions using R (R Development Core Team, 2012) One frequently called upon is the need to take a set of items (e.g., a questionnaire) and score one or more scales on that questionnaire. Scores for subsequent analysis, reliabilities and intercorrelations are easily done using the `score.items` function.

Suppose you have given a questionnaire with some items (n) to some participants (N). You would like to create scale scores for each person on k different scales. This may

*Part of a set of tutorials for the psych package.

be done using the *psych* package in R. The following assumes that you have installed R and downloaded the *psych* package.

1 Overview for the impatient

Remember, before using *psych* you must make it active:
```
library(psych)
```

1. Enter the data into a spreadsheet (Excel or Numbers) or a text file using a text editor (Word, Pages, BBEdit). The first line of the file should include names for the variables (e.g., Q1, Q2, ... Qn).
2. Copy the data to the clipboard (using the normal copy command for your spreadsheet or word processor).
3. Read the data into R using the read.clipboard command. (Depending upon your data file, this might need to be read.clipboard.csv (for comma separated data fields) or read.clipboard.tab (for tab separated data fields).
4. Construct a set of scoring keys for the scales you want to score. This is simply the item numbers that go into each scale. A negative sign implies that the item will be reverse scored.
5. Use the `score.items` function to score the scales.
6. Use the output for `score.items` for further analysis.

2 An example

Suppose we have 12 items for 20 subjects. The items represent 4 different scales: Positive Energetic Arousal (EAp), Negative Energetic Arousal (EAn), Tense Arousal (TAp) and negative Tense Arousal (TAn, also known as being relaxed). These four scales can also be thought of a forming two higher order constructs, Energetic Arousal (EA) and Tense Arousal (TA). EA is just EAp - EAn, and similarly TA is just TAp - TAn.

2.1 Getting the data

There are of course many ways to enter data into R. Reading from a local file using `read.table` is perhaps the most preferred. You first need to find the file and then read it. This can be done with the `file.choose` and `read.table` functions:
```
file.name <- file.choose()
my.data <- read.table(file.name)
```

`file.choose` opens a search window on your system just like any open file command does. It doesn't actually read the file, it just finds the file. The read command is also necessary.

2.1.1 Copy the data from another program using the copy and paste commands of your operating system

However, many users will enter their data in a text editor or spreadsheet program and then want to copy and paste into R. This may be done by using `read.table` and

specifying the input file as "clipboard" (PCs) or "pipe(pbpaste)" (Macs). Alternatively, the `read.clipboard` set of functions are perhaps more user friendly:

`read.clipboard` is the base function for reading data from the clipboard.

`read.clipboard.csv` for reading text that is comma delimited.

`read.clipboard.tab` for reading text that is tab delimited (e.g., copied directly from an Excel file).

`read.clipboard.lower` for reading input of a lower triangular matrix with or without a diagonal. The resulting object is a square matrix.

`read.clipboard.upper` for reading input of an upper triangular matrix.

`read.clipboard.fwf` for reading in fixed width fields (some very old data sets)

For example, given a data set copied to the clipboard from a spreadsheet, just enter the command

```
> my.data <- read.clipboard()
```

This will work if every data field has a value and even missing data are given some values (e.g., NA or -999). If the data were entered in a spreadsheet and the missing values were just empty cells, then the data should be read in as a tab delimited or by using the `read.clipboard.tab` function.

```
> my.data <- read.clipboard(sep="\t")   #define the tab option, or
> my.tab.data <- read.clipboard.tab()   #just use the alternative function
```

For the case of data in fixed width fields (some old data sets tend to have this format), copy to the clipboard and then specify the width of each field (in the example below, the first variable is 5 columns, the second is 2 columns, the next 5 are 1 column the last 4 are 3 columns).

```
> my.data <- read.clipboard.fwf(widths=c(5,2,rep(1,5),rep(3,4))
```

2.1.2 An example data set

Consider the data in Table 1. Read them into the clipboard and go. (These data are the first 20 cases from the `msq` data set in the *psych* package).

```
library(psych)
my.data <- read.clipboard.tab()  #tab delimited data from a spreadsheet or
my.data<- read.clipboard()   #data from a text editor with spaces between the fields.
describe(my.data)  # to make sure you got the right data in.
```

```
> describe(my.data)  # to make sure you got the right data in.
        var  n mean   sd median trimmed  mad min max range  skew kurtosis   se
active    1 20 0.75 0.91    0.5    0.62 0.74   0   3     3  0.87    -0.37 0.20
alert     2 20 0.80 0.70    1.0    0.75 0.74   0   2     2  0.25    -1.06 0.16
aroused   3 20 0.40 0.60    0.0    0.31 0.00   0   2     2  1.06    -0.01 0.13
sleepy    4 20 1.55 0.94    1.0    1.56 1.48   0   3     3  0.22    -1.10 0.21
tired     5 20 1.65 0.93    2.0    1.69 1.48   0   3     3 -0.05    -1.06 0.21
drowsy    6 20 1.50 0.89    1.0    1.50 1.48   0   3     3  0.21    -0.89 0.20
anxious   7  4 0.50 0.58    0.5    0.50 0.74   0   1     1  0.00    -2.44 0.29
jittery   8 20 0.50 0.76    0.0    0.38 0.00   0   3     3  1.70     3.00 0.17
nervous   9 20 0.15 0.49    0.0    0.00 0.00   0   2     2  2.94     7.68 0.11
calm     10 20 1.60 0.94    1.5    1.62 0.74   0   3     3  0.09    -1.10 0.21
```

Table 1: A sample data file with 12 items for 20 subjects.

	active	alert	aroused	sleepy	tired	drowsy	anxious	jittery	nervous	calm	relaxed	at-ease
1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	0	1	1	1	0	0	0	1	1	1
3	1	0	0	0	1	0	0	0	0	1	2	2
4	1	1	1	1	1	1	1	3	2	1	2	1
5	2	1	2	1	1	1	NA	1	0	3	3	3
6	2	1	1	2	2	2	NA	0	0	2	2	1
7	0	1	0	2	3	3	NA	0	0	2	2	1
8	0	0	0	1	2	1	NA	0	0	1	2	0
9	1	0	1	2	0	2	NA	1	0	0	2	2
10	0	2	0	2	2	2	NA	1	0	2	2	1
11	0	0	0	3	2	2	NA	0	0	2	2	2
12	1	1	0	1	1	1	NA	1	0	1	1	0
13	0	0	0	3	3	2	NA	1	0	0	2	0
14	2	1	1	1	0	0	NA	0	0	2	2	1
15	0	2	0	0	2	1	NA	0	0	3	3	3
16	0	0	0	3	3	3	NA	1	0	1	1	1
17	0	1	1	1	1	1	NA	0	0	1	1	1
18	3	2	0	2	2	3	NA	0	0	3	3	3
19	0	0	0	3	3	2	NA	0	0	2	1	0
20	0	1	0	1	2	1	NA	0	0	3	2	2

```
relaxed  11 20 1.85 0.67    2.0    1.81 0.00   1   3     2  0.15    -0.93 0.15
at.ease  12 20 1.30 0.98    1.0    1.25 1.48   0   3     3  0.38    -0.96 0.22
```

3 Scoring scales: an example

To score particular items on particular scales, we must create a set of *scoring keys*. These simply tell us which items go on which scales. Note that we can have scales with overlapping items.

Two things to note. The number of variables is the total number of variables (columns) in the data file. You do not need to include all of these items in the scoring keys, but you need to say how many there are. For the keys, items are scored either +1, -1 or 0 (not scored). Just specify the items to score and their direction.

```
my.keys <- make.keys(nvars=12,list(EA=c(1:3,-4,-5,-6),TA=c(7:9,-10,-11,-12),
                    EAp =1:3,EAn=4:6,TAp =7:9,TAn=10:12))
my.scales <- score.items(my.keys,my.data)
my.scales   #show the output
my.scores <- my.scales$scores
```

Produces this output:

```
> my.keys <- make.keys(nvars=12,list(EA=c(1:3,-4,-5,-6),TA=c(7:9,-10,-11,-12),
+                     EAp =1:3,EAn=4:6,TAp =7:9,TAn=10:12))
> my.scales <- score.items(my.keys,my.data)
> my.scales   #show the output
Call: score.items(keys = my.keys, items = my.data)
```

```
(Unstandardized) Alpha:
        EA   TA  EAp  EAn  TAp  TAn
alpha 0.77 0.73 0.57 0.86 0.78 0.82

Average item correlation:
            EA   TA EAp  EAn  TAp TAn
average.r 0.36 0.31 0.3 0.68 0.54 0.6

 Guttman 6* reliability:
           EA   TA  EAp  EAn  TAp TAn
Lambda.6 0.92 0.91 0.78 0.94 0.89 0.9

Scale intercorrelations corrected for attenuation
 raw correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
       EA     TA   EAp    EAn   TAp   TAn
EA   0.77 -0.215  1.15 -1.102  0.21  0.38
TA  -0.16  0.728 -0.47  0.032  0.84 -1.15
EAp  0.76 -0.301  0.57 -0.569  0.29  0.73
EAn -0.90  0.026 -0.40  0.863 -0.12 -0.11
TAp  0.16  0.630  0.19 -0.097  0.78 -0.24
TAn  0.30 -0.885  0.50 -0.091 -0.20  0.82

 In order to see the item by scale loadings and frequency counts of the data
 print with the short option = FALSE
```

Two things to notice about this output is a) the message about how to get more information (item by scale correlations and frequency counts) and b) that the correlation matrix between the six scales has the raw correlations below the diagonal, alpha reliabilities on the diagonal, and correlations adjusted for reliability above the diagonal. Because EAp and EAn are both part of EA, they correlate with the total more than would be expected given their reliability. Hence the impossible values greater than |0.0|.

3.1 Long output

To get the scale correlations corrected for item overlap and scale reliability, we print the object that we found, but ask for long output.

```
print(my.scales,short=FALSE)
```

```
Call: score.items(keys = my.keys, items = my.data)

(Unstandardized) Alpha:
        EA   TA  EAp  EAn  TAp  TAn
alpha 0.77 0.73 0.57 0.86 0.78 0.82

Average item correlation:
            EA   TA EAp  EAn  TAp TAn
average.r 0.36 0.31 0.3 0.68 0.54 0.6
```

```
 Guttman 6* reliability:
           EA   TA  EAp  EAn  TAp TAn
Lambda.6 0.92 0.91 0.78 0.94 0.89 0.9

Scale intercorrelations corrected for attenuation
 raw correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
       EA     TA   EAp    EAn   TAp   TAn
EA   0.77 -0.215  1.15 -1.102  0.21  0.38
TA  -0.16  0.728 -0.47  0.032  0.84 -1.15
EAp  0.76 -0.301  0.57 -0.569  0.29  0.73
EAn -0.90  0.026 -0.40  0.863 -0.12 -0.11
TAp  0.16  0.630  0.19 -0.097  0.78 -0.24
TAn  0.30 -0.885  0.50 -0.091 -0.20  0.82

Item by scale correlations:
 corrected for item overlap and scale reliability
           EA    TA   EAp   EAn   TAp   TAn
active   0.55 -0.29  0.75 -0.30  0.06  0.40
alert    0.40 -0.42  0.58 -0.20  0.07  0.57
aroused  0.57  0.06  0.60 -0.43  0.40  0.17
sleepy  -0.79  0.20 -0.40  0.86 -0.03 -0.27
tired   -0.85 -0.08 -0.60  0.82 -0.20 -0.02
drowsy  -0.73 -0.05 -0.18  0.90 -0.05  0.04
anxious  0.03  0.40  0.24  0.10  0.77 -0.05
jittery  0.12  0.59  0.17 -0.06  0.86 -0.24
nervous  0.27  0.55  0.23 -0.23  0.91 -0.16
calm     0.17 -0.78  0.45  0.04 -0.30  0.81
relaxed  0.26 -0.65  0.48 -0.06 -0.06  0.79
at.ease  0.38 -0.74  0.53 -0.21 -0.14  0.85

Non missing response frequency for each item
           0    1    2    3 miss
active  0.50 0.30 0.15 0.05  0.0
alert   0.35 0.50 0.15 0.00  0.0
aroused 0.65 0.30 0.05 0.00  0.0
sleepy  0.10 0.45 0.25 0.20  0.0
tired   0.10 0.35 0.35 0.20  0.0
drowsy  0.10 0.45 0.30 0.15  0.0
anxious 0.50 0.50 0.00 0.00  0.8
jittery 0.60 0.35 0.00 0.05  0.0
nervous 0.90 0.05 0.05 0.00  0.0
calm    0.10 0.40 0.30 0.20  0.0
relaxed 0.00 0.30 0.55 0.15  0.0
at.ease 0.20 0.45 0.20 0.15  0.0
```

3.2 Get the actual scores for analysis.

Although we would probably not look at the raw scores, we can if we want by asking for the scores object which is part of the my.scales output. For printing purposes, we round them to two decimal places for compactness.

```
my.scores <- my.scales$scores
round(my.scores,2)
```

```
> round(my.scores,2)
     EA   TA  EAp  EAn  TAp  TAn
```

```
1  1.50 1.50 1.00 1.00 1.00 1.00
2  1.33 1.00 0.67 1.00 0.00 1.00
3  1.50 0.67 0.33 0.33 0.00 1.67
4  1.50 1.83 1.00 1.00 2.00 1.33
5  1.83 0.25 1.67 1.00 0.50 3.00
6  1.17 0.75 1.33 2.00 0.17 1.67
7  0.33 0.75 0.33 2.67 0.17 1.67
8  0.83 1.08 0.00 1.33 0.17 1.00
9  1.17 1.08 0.67 1.33 0.50 1.33
10 0.83 0.92 0.67 2.00 0.50 1.67
11 0.33 0.58 0.00 2.33 0.17 2.00
12 1.33 1.42 0.67 1.00 0.50 0.67
13 0.17 1.42 0.00 2.67 0.50 0.67
14 2.00 0.75 1.33 0.33 0.17 1.67
15 1.33 0.08 0.67 1.00 0.17 3.00
16 0.00 1.25 0.00 3.00 0.50 1.00
17 1.33 1.08 0.67 1.00 0.17 1.00
18 1.17 0.08 1.67 2.33 0.17 3.00
19 0.17 1.08 0.00 2.67 0.17 1.00
20 1.00 0.42 0.33 1.33 0.17 2.33
```

4 The example, continued

Once you have the results, you should probably want to `describe` them and also show a graphic of the scatterplot using the `pairs.panels` function (Figure 1).

```
describe(my.scores)
pairs.panels(my.scores}
```

```
P> describe(my.scores)
     var  n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
EA     1 20 1.04 0.57   1.17    1.05 0.49 0.00 2.00  2.00 -0.38    -1.05 0.13
TA     2 20 0.90 0.47   0.96    0.91 0.43 0.08 1.83  1.75 -0.07    -0.81 0.11
EAp    3 20 0.65 0.55   0.67    0.60 0.49 0.00 1.67  1.67  0.42    -1.01 0.12
EAn    4 20 1.57 0.82   1.33    1.56 0.74 0.33 3.00  2.67  0.27    -1.35 0.18
TAp    5 20 0.38 0.45   0.17    0.29 0.12 0.00 2.00  2.00  2.34     5.62 0.10
```

5 Even more analysis

Far more analyses could be done with these data, but the basic scale scoring techniques is a start. Download the vignette for using *psych* for even more guidance. http://cran.r-project.org/web/packages/psych/vignettes/overview.pdf. On a Mac, this is also available in the vignettes list in the help menu.

In addition, look at the examples in the help for `score.items`.

5.1 Exploring a real data set

.

The 12 mood items for 20 subjects were taken from the much larger data set, `msq` in the *psych* package. That data set has 92 variables for 3896 subjects. We can repeat

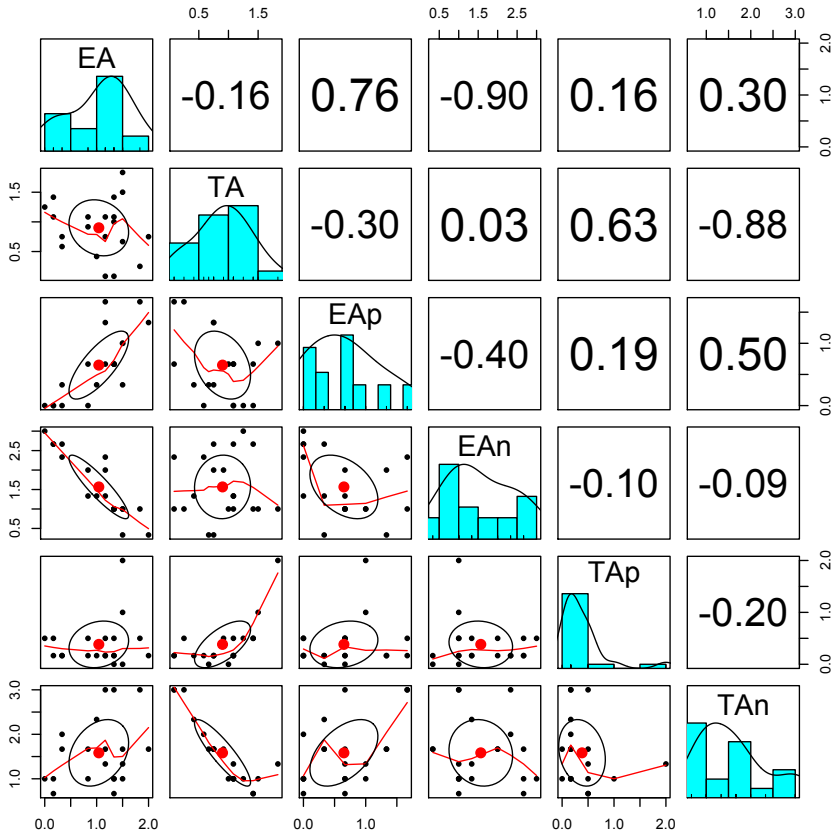

Figure 1: A simple scatter plot matrix shows the histograms for the variables on the diagonal, the correlations above the diagonal, and the scatter plots below the diagonal. The best fitting loess regression is shown as the red line.

our analysis of EA and TA on that data set.

First we get the data for the items that match our small example. Then we describe the data, and finally, find the 6 scales as we did before.

```
select <- colnames(my.data)
select[12] <- 'at-ease'
small.msq <- msq[select]
describe(small.msq)
msq.scales <- score.items(my.keys,small.msq)
msq.scales   #show the output
```

```
       var    n mean   sd median trimmed  mad min max range  skew kurtosis   se
active     1 3890 1.03 0.93      1    0.95 1.48   0   3     3  0.47    -0.76 0.01
alert      2 3885 1.15 0.91      1    1.09 1.48   0   3     3  0.33    -0.76 0.01
aroused    3 3890 0.71 0.85      0    0.59 0.00   0   3     3  0.95    -0.04 0.01
sleepy     4 3880 1.25 1.05      1    1.18 1.48   0   3     3  0.40    -1.04 0.02
tired      5 3886 1.39 1.04      1    1.36 1.48   0   3     3  0.22    -1.10 0.02
drowsy     6 3884 1.16 1.03      1    1.08 1.48   0   3     3  0.46    -0.93 0.02
anxious    7 2047 0.67 0.86      0    0.54 0.00   0   3     3  1.09     0.26 0.02
jittery    8 3890 0.59 0.80      0    0.45 0.00   0   3     3  1.24     0.81 0.01
nervous    9 3879 0.35 0.65      0    0.22 0.00   0   3     3  1.93     3.47 0.01
calm      10 3814 1.55 0.92      2    1.56 1.48   0   3     3 -0.01    -0.83 0.01
relaxed   11 3889 1.68 0.88      2    1.72 1.48   0   3     3 -0.17    -0.68 0.01
at-ease   12 3879 1.59 0.92      2    1.61 1.48   0   3     3 -0.09    -0.83 0.01

Call: score.items(keys = my.keys, items = small.msq)

(Unstandardized) Alpha:
        EA   TA  EAp  EAn  TAp TAn
alpha 0.87 0.75 0.81 0.93 0.64 0.8

Average item correlation:
            EA   TA  EAp  EAn  TAp  TAn
average.r 0.54 0.34 0.58 0.81 0.37 0.57

 Guttman 6* reliability:
          EA   TA  EAp EAn  TAp  TAn
Lambda.6 0.9 0.77 0.76 0.9 0.59 0.74

Scale intercorrelations corrected for attenuation
 raw correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
        EA      TA    EAp    EAn    TAp    TAn
EA   0.874 -0.0207  1.004 -1.006  0.218  0.168
TA  -0.017  0.7515 -0.011  0.024  1.096 -1.140
EAp  0.842 -0.0084  0.806 -0.618  0.360  0.246
EAn -0.906  0.0197 -0.534  0.927 -0.067 -0.076
TAp  0.163  0.7590  0.258 -0.052  0.638 -0.512
TAn  0.141 -0.8837  0.198 -0.065 -0.366  0.800
```

References

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Revelle, W. (2013). *psych: Procedures for Personality and Psychological Research*. Northwestern University, Evanston, http://cran.r-project.org/web/packages/psych/. R package version 1.3.1.

Index