Using R to score personality scales*

William Revelle
Northwestern University

June 18, 2021

Abstract

The *psych* package (Revelle, 2020) was developed to perform most basic psychometric functions using R (R Core Team, 2020) A common problem 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 `scoreItems` function.

Contents

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

1 Overview of this and related documents

To do basic and advanced personality and psychological research using R is not as complicated as some think. This is one of a set of "How To" to do various things using R (R Core Team, 2020), particularly using the *psych* (Revelle, 2020) package.

The current list of How To's includes:

1. An introduction (vignette) of the *psych* package
2. An overview (vignette) of the *psych* package
3. Installing R and some useful packages
4. Using R and the *psych* package to find $omega_h$ and ω_t.
5. Using R and the *psych* for factor analysis and principal components analysis.
6. Using the `scoreItems` function to find scale scores and scale statistics (this document).
7. Using `mediate` and `setCor` to do mediation, moderation and regression analysis

By following these simple guides, you soon will be able to do such things as find scale scores by issuing just five lines of code:

R code

```
library(psych)  #necessary whenever you want to run functions in the psych package
my.data <- read.clipboard()
my.keys <- make.keys(my.data,list(scale1 = c(1,4,5),scale2 = c(2,3,6))  #etec
my.scales <- scoreItems(my.keys,my.data)
my.scales
```

The resulting output will be both graphical and textual.

This guide helps the naive R user to issue those three lines. Be careful, for once you start using R, you will want to do more.

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

2 Overview for the impatient

Remember, before using *psych* and the *psychTools* packages you must make them active:

```
library(psych)
library(psychTools)
```

Then

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) or save it as .txt or .csv file.
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. Or, if you have a data file already that end in .sav, .text, .txt, .csv, .xpt, .rds, .Rds, .rda, or .RDATA, then just read it in directly using `read.file` (from *psychTools*)..
5. Construct a set of scoring keys for the scales you want to score using the `make.keys` function. This is simply the item names or numbers that go into each scale. A negative sign implies that the item will be reverse scored.
6. Use the `scoreItems` function to score the scales.
7. Use the output for `scoreItems` for further analysis.

3 An example

Suppose we have 12 items for 200 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.

3.1 Getting the data

There are, of course, many ways to enter data into R.

3.1.1 Reading from a local file

Reading from a local file using `read.table` is perhaps the most common procedure. You first need to find the file and then read it. This can be done with the `file.choose` and `read.table` functions. `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.

R code

```
file.name <- file.choose()
my.data <- read.table(file.name)
```

Even easier is to use the `read.file` function which combines the `file.chooose` and `read.table` functions into one function. In addition, `read.file` will read normal text (txt) files, comma separated files (csv), SPSS (sav) files as well as files saved by R(rds) files. By default, it assumes that the first line of the file has header information (variable names). `read.file` is included in the *psychTools* package which needs to be installed (once) and made active using the `library` function.

R code

```
my.data <- read.file()  #locate the file to read using your normal system.
```

3.2 Reading from a remote file

To read from a file saved on a remote server, you just need to specify the file name and then read it. By using the `read.file` function, you can read a variety of file types (e.g., text, txt, csv, sav, rds) from the remote server. e.g., to read the file `http://personality-project.org/r/psych/HowTo/sco` we just

```
file.name <- "http://personality-project.org/r/psych/HowTo/scoring.tutorial/small.m
my.data <- read.file(file.name)

## Data from the .txt file http://personality-project.org/r/psych/HowTo/scoring.tut
has been loaded.
```

3.2.1 Read from the clipboard

Many users find it more convenient to enter their data in a text editor or spreadsheet program and then just copy and paste into R. The `read.clipboard` set of functions are perhaps more user friendly. These functions are included in the *psychTools* package:

`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.

R code
```
my.data <- read.clipboard()   #note the parentheses
```

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.

R code
```
my.tab.data <- read.clipboard.tab()   #This is for data from a spreadsheet
```

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).

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

3.2.2 An example data set

Consider the data stored at a remote data location. (This is the same file that we read directly above.) Open the file in your browser and then select all. Read them into the clipboard and

go. (These data are selected variables from the first 200 cases from the msqR data set in the *psychTools* package). Once you have read the data, it useful to see how many cases and how many variables you have dim and to find some basic descriptive statistics.

R code

```
my.data <- read.clipboard()
headTail(my.data)
dim(my.data)
describe(my.data)
```

```
> headTail(my.data)
    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
...    ...   ...     ...    ...   ...    ...     ...     ...     ...  ...     ...     ...
197      1     1       0      1     2      1       0       0       0    1       1       1
198      2     2       0      0     1      0       1       0       0    2       3       3
199      1     3       0      1     0      1       0       1       0    3       3       3
200      1     2       0      1     1      0       0       0       0    2       3       3
> dim(my.data)
[1] 200  12
```

```
fn <- "http://personality-project.org/r/psych/HowTo/scoring.tutorial/small.msq"
my.data <- read.file(fn, filetype="txt")  #because the suffix is not a standard one

## Data from the .txt file http://personality-project.org/r/psych/HowTo/scoring.tut
has been loaded.

 dim(my.data)  #same as before

## [1] 200  12

 headTail(my.data) #same as before

##     active alert aroused sleepy tired drowsy anxious jittery nervous calm
## 1        1     1       1      1     1      1       1       1       1    1
## 2        1     1       0      1     1      1       0       0       0    1
## 3        1     0       0      0     1      0       0       0       0    1
## 4        1     1       1      1     1      1       1       3       2    1
## ...    ...   ...     ...    ...   ...    ...     ...     ...     ...  ...
## 197      1     1       0      1     2      1       0       0       0    1
## 198      2     2       0      0     1      0       1       0       0    2
## 199      1     3       0      1     0      1       0       1       0    3
## 200      1     2       0      1     1      0       0       0       0    2
##     relaxed at.ease
## 1         1       1
## 2         1       1
## 3         2       2
## 4         2       1
```

```
## ...     ...     ...
## 197       1       1
## 198       3       3
## 199       3       3
## 200       3       3
```

```
describe(my.data)
```

```
##         vars   n mean   sd median trimmed  mad min max range  skew kurtosis
## active     1 199 0.70 0.82      1    0.59 1.48   0   3     3  0.97     0.20
## alert      2 197 0.83 0.75      1    0.77 1.48   0   3     3  0.51    -0.41
## aroused    3 199 0.38 0.63      0    0.25 0.00   0   3     3  1.55     1.68
## sleepy     4 199 1.77 1.04      2    1.84 1.48   0   3     3 -0.21    -1.22
## tired      5 198 1.87 1.00      2    1.96 1.48   0   3     3 -0.43    -0.95
## drowsy     6 198 1.61 1.05      2    1.63 1.48   0   3     3 -0.07    -1.21
## anxious    7 101 0.42 0.71      0    0.27 0.00   0   3     3  1.87     3.36
## jittery    8 198 0.37 0.61      0    0.27 0.00   0   3     3  1.66     2.74
## nervous    9 199 0.30 0.63      0    0.16 0.00   0   3     3  2.39     5.79
## calm      10 199 1.66 0.91      2    1.70 1.48   0   3     3 -0.01    -0.91
## relaxed   11 199 1.67 0.89      2    1.70 1.48   0   3     3 -0.05    -0.81
## at.ease   12 199 1.53 0.93      1    1.53 1.48   0   3     3  0.07    -0.87
##             se
## active    0.06
## alert     0.05
## aroused   0.04
## sleepy    0.07
## tired     0.07
## drowsy    0.07
## anxious   0.07
## jittery   0.04
## nervous   0.04
## calm      0.06
## relaxed   0.06
## at.ease   0.07
```

3.3 Reading data from a Qualtrics data set

If you have used Qualtrics to collect your data, you can export the data as a csv data file. Unfortunately, this file is poorly organized and has one too many header lines. You can open the file using a spreadsheet program (e.g. Excel) and then change the line above the data to be item labels (e.g., Q1, Q2, ...). Then select that line and all the lines of data that you want to read, and use the `read.clipboard.tab` function (see above).

4 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. There are several ways of creating keys. Probably the most intuitive is just to make up a list of keys. For example, make up keys to score Energetic Arousal, Activated Arousal, Deactivated Arousal ... We can do this by specifying the names of the items or their location. For demonstration purposes, we do this both ways.

```
my.keys <-  list(EA= c("active","alert","aroused", "-sleepy","-tired", "-drowsy"),
                           TA = c("anxious","jittery","nervous","-calm", "-relaxed
                           EAp = c("active","alert","aroused"),
                           EAn = c("sleepy","tired", "drowsy"),
                           TAp = c("anxious","jittery","nervous"),
                           TAn = c("calm", "relaxed", "at.ease")
                           )

 another.keys.list <- 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)
```

Earlier versions of `scoreItems` required a `keys` matrix created by `make.keys`. This is no longer necessary.

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.

What is actually done internally in the `scoreItem` function is the keys.list are converted to a matrix of -1s, 0s, and 1s to represent the scoring keys and this multiplied by the data.

Now, we use those keys to score the data using `scoreItems`:

These three commands allow you to score the scales, find various descriptive statistics (such as coefficient α), the scale intercorrelations, and also to get the scale scores.

```
my.scales <- scoreItems(my.keys,my.data)
my.scales   #show the output
```

```
## Call: scoreItems(keys = my.keys, items = my.data)
##
## (Unstandardized) Alpha:
##         EA   TA  EAp  EAn  TAp TAn
## alpha 0.84 0.74 0.72 0.91 0.66 0.8
##
## Standard errors of unstandardized Alpha:
##          EA    TA   EAp   EAn   TAp   TAn
## ASE   0.035 0.044 0.071 0.052 0.077 0.063
##
## Average item correlation:
##             EA   TA  EAp  EAn  TAp  TAn
## average.r 0.46 0.33 0.46 0.76 0.39 0.58
##
## Median item correlation:
```

```
##   EA   TA  EAp  EAn  TAp  TAn
## 0.42 0.26 0.53 0.78 0.37 0.55
##
##  Guttman 6* reliability:
##            EA   TA  EAp  EAn  TAp  TAn
## Lambda.6 0.88 0.79 0.71 0.88 0.61 0.77
##
## Signal/Noise based upon av.r :
##               EA  TA EAp EAn TAp TAn
## Signal/Noise 5.2 2.9 2.6 9.7 1.9 4.1
##
## 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   0.8387 -0.30  1.00 -1.06 -0.0045  0.38
## TA  -0.2357  0.74 -0.26  0.26  1.0011 -1.17
## EAp  0.7819 -0.19  0.72 -0.59  0.2350  0.45
## EAn -0.9208  0.21 -0.48  0.91  0.1373 -0.26
## TAp -0.0033  0.70  0.16  0.11  0.6574 -0.45
## TAn  0.3100 -0.90  0.35 -0.22 -0.3272  0.80
##
##  In order to see the item by scale loadings and frequency counts of the data
##  print with the short option = FALSE
```

```
my.scores <- my.scales$scores    #the actual scores are saved in the scores object
```

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 $|1.0|$.

4.1 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: scoreItems(keys = my.keys, items = my.data)
##
## (Unstandardized) Alpha:
##         EA   TA  EAp  EAn  TAp TAn
## alpha 0.84 0.74 0.72 0.91 0.66 0.8
```

```
##
## Standard errors of unstandardized Alpha:
##          EA    TA   EAp   EAn   TAp   TAn
## ASE   0.035 0.044 0.071 0.052 0.077 0.063
##
## Average item correlation:
##             EA   TA  EAp  EAn  TAp  TAn
## average.r 0.46 0.33 0.46 0.76 0.39 0.58
##
## Median item correlation:
##   EA   TA  EAp  EAn  TAp  TAn
## 0.42 0.26 0.53 0.78 0.37 0.55
##
##  Guttman 6* reliability:
##            EA   TA  EAp  EAn  TAp  TAn
## Lambda.6 0.88 0.79 0.71 0.88 0.61 0.77
##
## Signal/Noise based upon av.r :
##                EA  TA EAp EAn TAp TAn
## Signal/Noise 5.2 2.9 2.6 9.7 1.9 4.1
##
## 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   0.8387 -0.30  1.00 -1.06 -0.0045  0.38
## TA  -0.2357  0.74 -0.26  0.26  1.0011 -1.17
## EAp  0.7819 -0.19  0.72 -0.59  0.2350  0.45
## EAn -0.9208  0.21 -0.48  0.91  0.1373 -0.26
## TAp -0.0033  0.70  0.16  0.11  0.6574 -0.45
## TAn  0.3100 -0.90  0.35 -0.22 -0.3272  0.80
##
## Item by scale correlations:
##  corrected for item overlap and scale reliability
##            EA    TA   EAp   EAn   TAp   TAn
## active   0.64 -0.25  0.78 -0.47  0.10  0.39
## alert    0.58 -0.26  0.63 -0.47  0.08  0.39
## aroused  0.44  0.04  0.62 -0.27  0.36  0.14
## sleepy  -0.85  0.22 -0.55  0.89  0.12 -0.23
## tired   -0.82  0.29 -0.55  0.85  0.16 -0.30
## drowsy  -0.78  0.16 -0.46  0.84  0.09 -0.17
## anxious -0.11  0.34  0.00  0.15  0.50 -0.19
## jittery  0.03  0.48  0.21  0.07  0.62 -0.31
## nervous  0.05  0.53  0.22  0.05  0.68 -0.35
## calm     0.11 -0.65  0.24 -0.02 -0.35  0.68
```

```
## relaxed  0.33 -0.71  0.34 -0.28 -0.37  0.76
## at.ease  0.39 -0.72  0.47 -0.29 -0.34  0.79
##
## Non missing response frequency for each item
##             0    1    2    3 miss
## active   0.49 0.35 0.13 0.04 0.01
## alert    0.37 0.46 0.16 0.02 0.02
## aroused  0.70 0.23 0.07 0.01 0.01
## sleepy   0.13 0.30 0.25 0.33 0.01
## tired    0.12 0.23 0.33 0.33 0.01
## drowsy   0.17 0.30 0.27 0.25 0.01
## anxious  0.68 0.25 0.04 0.03 0.50
## jittery  0.69 0.26 0.04 0.01 0.01
## nervous  0.78 0.17 0.04 0.02 0.01
## calm     0.09 0.37 0.33 0.21 0.01
## relaxed  0.09 0.35 0.37 0.20 0.01
## at.ease  0.13 0.38 0.32 0.17 0.01
```

4.2 Correcting for overlapping items across scales

The `scoreOverlap` function will correct for item overlap. In the case of overlapping keys, (items being scored on multiple scales), scoreOverlap will adjust for this overlap by replacing the overlapping covariances (which are variances when overlapping) with the corresponding best estimate of an item's "true" variance using either the average correlation or the smc estimate for that item. This parallels the operation done when finding alpha reliability. This is similar to ideas suggested by Cureton (1966) and Bashaw and Anderson Jr (1967) but uses the smc or the average interitem correlation (default).

R code

```
scales.ov <- scoreOverlap(my.keys,my.data)
scales.ov
```

```
 scales
Call: scoreOverlap(keys = my.keys, r = my.data)

(Standardized) Alpha:
  EA   TA  EAp  EAn  TAp  TAn
0.83 0.76 0.72 0.91 0.73 0.81

(Standardized) G6*:
  EA   TA  EAp  EAn  TAp  TAn
0.66 0.61 0.72 0.88 0.69 0.77

Average item correlation:
  EA   TA  EAp  EAn  TAp  TAn
0.45 0.34 0.47 0.76 0.48 0.58

Number of items:
 EA  TA EAp EAn TAp TAn
  6   6   3   3   3   3

Signal to Noise ratio based upon average r and n
```

```
 EA  TA EAp EAn TAp TAn
5.0 3.1 2.6 9.8 2.7 4.1

Scale intercorrelations corrected for item overlap and attenuation
 adjusted for overlap correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
       EA     TA   EAp    EAn    TAp   TAn
EA   0.833 -0.211  0.88 -0.893  0.073  0.39
TA  -0.168  0.758 -0.12  0.234  0.831 -0.85
EAp  0.684 -0.086  0.72 -0.579  0.286  0.44
EAn -0.776  0.194 -0.47  0.907  0.111 -0.26
TAp  0.057  0.619  0.21  0.091  0.733 -0.42
TAn  0.320 -0.661  0.34 -0.222 -0.320  0.81

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

4.3 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. We just look at first 10 cases.

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

```
##       EA   TA  EAp  EAn  TAp  TAn
## 1    1.5  1.5    1    1    1    1
## 2   1.33    1 0.67    1    0    1
## 3    1.5 0.67 0.33 0.33    0 1.67
## 4    1.5 1.83    1    1    2 1.33
## ...  ...  ...  ...  ...  ...  ...
## 197 1.17    1 0.67 1.33    0    1
## 198    2 0.33 1.33 0.33 0.33 2.67
## 199 1.83 0.17 1.33 0.67 0.33    3
## 200 1.67 0.17    1 0.67    0 2.67
```

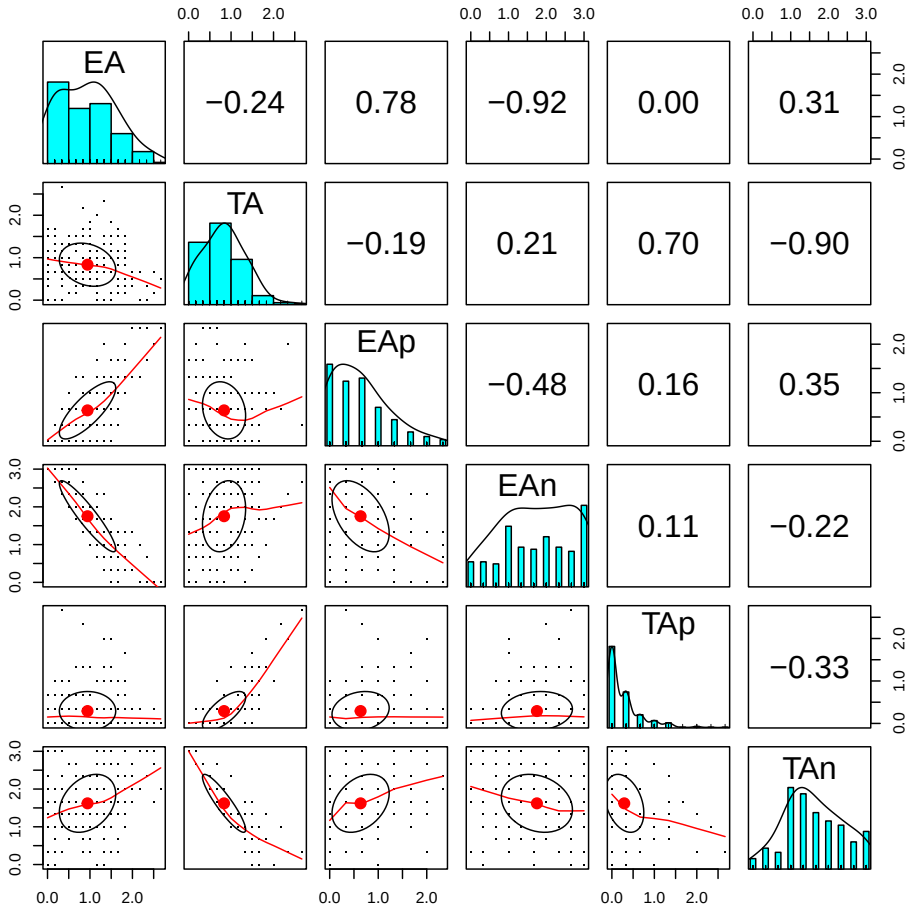
4.4 The example, continued

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). (Note that for the figure, we set the plot character to be '.' so that it makes a cleaner plot.)

```
describe(my.scores)
```

```
##      vars   n mean   sd median trimmed  mad min  max range  skew kurtosis   se
## EA      1 200 0.94 0.66   1.00    0.91 0.74   0 2.67  2.67  0.34    -0.66 0.05
## TA      2 200 0.84 0.51   0.83    0.82 0.49   0 2.67  2.67  0.40     0.15 0.04
## EAp     3 200 0.64 0.59   0.67    0.56 0.49   0 2.33  2.33  0.90     0.21 0.04
## EAn     4 200 1.75 0.94   1.67    1.80 0.99   0 3.00  3.00 -0.20    -1.13 0.07
## TAp     5 200 0.29 0.46   0.00    0.19 0.00   0 2.67  2.67  2.26     6.08 0.03
## TAn     6 200 1.62 0.77   1.67    1.61 0.99   0 3.00  3.00  0.12    -0.69 0.05
```

```
pairs.panels(my.scores,pch='.')
```




```
## pdf
##   2
```

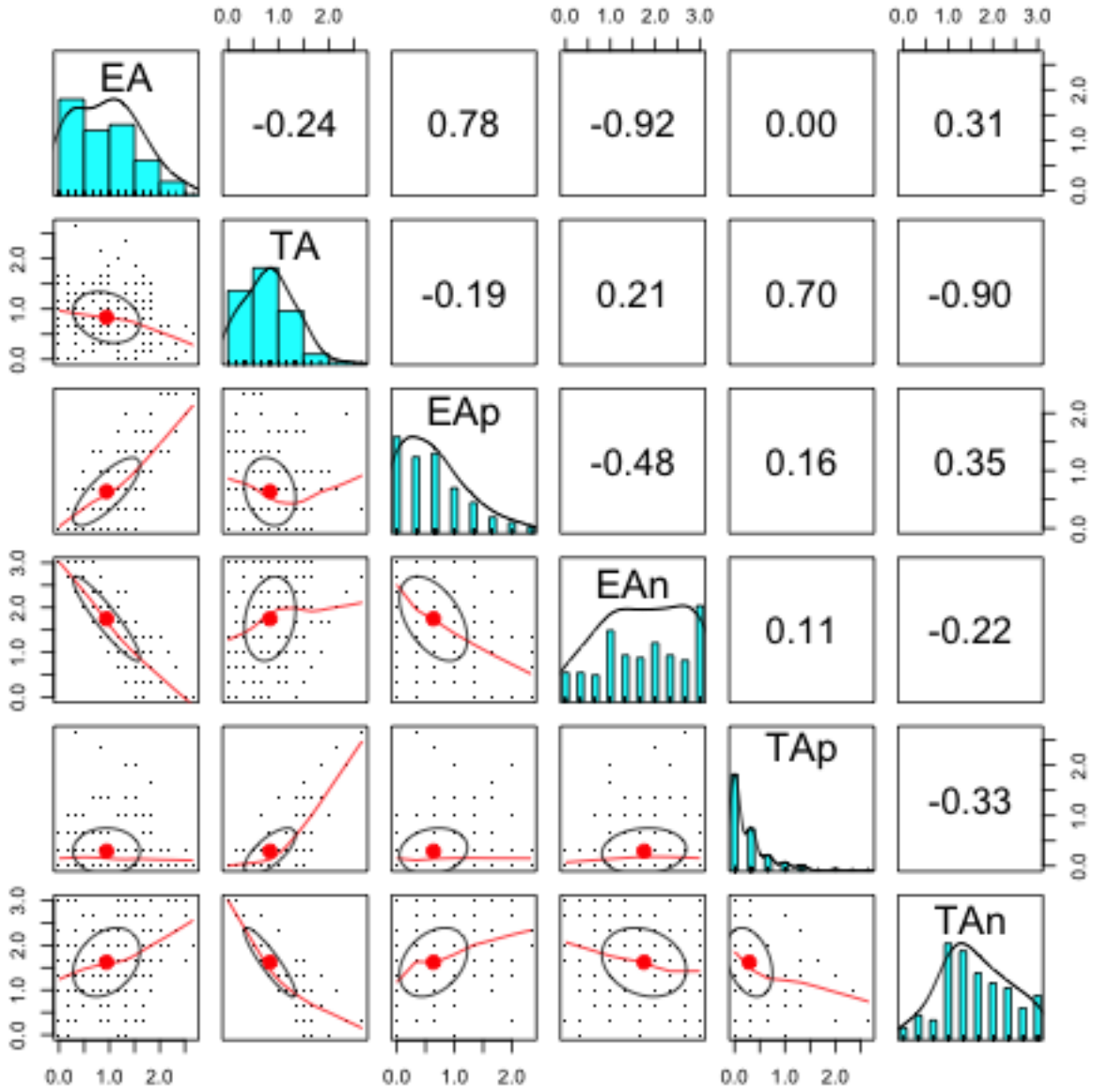


Figure 1: A simple scatter plot matrix shows the histograms for the variables on the diagonal, the correlations above the diagonal, and the scatter plots below the diagonal. The best fitting loess regression is shown as the red line.

5 Exploring a real data set

The 12 mood items for 200 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 our analysis of EA and TA on that data set. This is a data set collected over about 10 years at the Personality, Motivation and Cognition laboratory at Northwestern and described by Revelle and Anderson (1997) and Rafaeli and Revelle (2006).

5.1 Conventional reliability and scoring

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. Note that the colnames of our small sample are the items we want to pick from the larger set. Another way to choose some items is to use the `selectFromKeys` function.

```
select <- colnames(my.data)
#or
select <- selectFromKeys(my.keys)

small.msq <- msq[select]
describe(small.msq)
```

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

```
## relaxed 0.01
## at.ease 0.01
```

```
msq.scales <- scoreItems(my.keys,small.msq)
msq.scales   #show the output
```

```
## Call: scoreItems(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
##
## Standard errors of unstandardized Alpha:
##           EA     TA   EAp   EAn   TAp   TAn
## ASE   0.0071 0.0099 0.014 0.011 0.018 0.014
##
## Average item correlation:
##             EA   TA  EAp  EAn  TAp  TAn
## average.r 0.54 0.34 0.58 0.81 0.37 0.57
##
## Median item correlation:
##   EA   TA  EAp  EAn  TAp  TAn
## 0.53 0.28 0.60 0.80 0.36 0.58
##
##  Guttman 6* reliability:
##           EA   TA  EAp EAn  TAp  TAn
## Lambda.6 0.9 0.77 0.76 0.9 0.59 0.74
##
## Signal/Noise based upon av.r :
##               EA TA EAp EAn TAp TAn
## Signal/Noise 6.9  3 4.1  13 1.8   4
##
## 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
##
##  In order to see the item by scale loadings and frequency counts of the data
##  print with the short option = FALSE
```

5.2 Show the correction for overlap

```
msq.scales.ov <- scoreOverlap(my.keys,small.msq)
msq.scales.ov   #show the output
```

```
## Call: scoreOverlap(keys = my.keys, r = small.msq)
##
## (Standardized) Alpha:
##   EA   TA  EAp  EAn  TAp  TAn
## 0.87 0.78 0.81 0.93 0.73 0.80
##
## (Standardized) G6*:
##   EA   TA  EAp  EAn  TAp  TAn
## 0.69 0.63 0.76 0.90 0.67 0.75
##
## Average item correlation:
##   EA   TA  EAp  EAn  TAp  TAn
## 0.53 0.37 0.58 0.81 0.48 0.58
##
## Median item correlation:
##   EA   TA  EAp  EAn  TAp  TAn
## 0.53 0.28 0.60 0.81 0.47 0.58
##
## Number of items:
##  EA  TA EAp EAn TAp TAn
##   6   6   3   3   3   3
##
## Signal to Noise ratio based upon average r and n
##   EA   TA  EAp  EAn  TAp  TAn
##  6.8  3.5  4.2 12.9  2.7  4.1
##
## Scale intercorrelations corrected for item overlap and attenuation
##  adjusted for overlap correlations below the diagonal, alpha on the diagonal
##  corrected correlations above the diagonal:
##         EA      TA    EAp     EAn    TAp    TAn
## EA   0.872  0.0264  0.895 -0.8999  0.240  0.176
## TA   0.022  0.7753  0.057  0.0053  0.856 -0.866
## EAp  0.750  0.0450  0.806 -0.6153  0.369  0.244
## EAn -0.810  0.0045 -0.532  0.9283 -0.077 -0.078
## TAp  0.192  0.6447  0.283 -0.0632  0.732 -0.490
## TAn  0.147 -0.6837  0.197 -0.0675 -0.376  0.804
##
##  In order to see the item by scale loadings and frequency counts of the data
##  print with the short option = FALSE
```

6 Even more analysis: Factoring, clustering, and more tutorials

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. 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 `scoreItems`.

For examples and tutorials in how to do *factor analysis* or to find *coefficient omega* see the factor analysis tutorial or the omega tutorial.

7 Session Information

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dyli
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dy
##
## locale:
## [1] C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] psychTools_2.1.5 psych_2.1.6
##
## loaded via a namespace (and not attached):
##  [1] lattice_0.20-44 grid_4.1.0      nlme_3.1-152    magrittr_2.0.1
##  [5] evaluate_0.14   highr_0.9       stringi_1.6.2   tools_4.1.0
##  [9] stringr_1.4.0   foreign_0.8-81  xfun_0.24       parallel_4.1.0
## [13] compiler_4.1.0  mnormt_2.0.1    tmvnsim_1.0-2   knitr_1.33
```

References

Bashaw, W. and Anderson Jr, H. E. (1967). A correction for replicated error in correlation coefficients. *Psychometrika*, 32(4):435–441.

Cureton, E. (1966). Corrected item-test correlations. *Psychometrika*, 31(1):93–96.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rafaeli, E. and Revelle, W. (2006). A premature consensus: Are happiness and sadness truly opposite affects? *Motivation and Emotion*, 30(1):1–12.

Revelle, W. (2020). *psych: Procedures for Personality and Psychological Research*. Northwestern University, Evanston, https://CRAN.r-project.org/package=psych. R package version 2.0.8.

Revelle, W. and Anderson, K. J. (1997). Personality, motivation and cognitive performance. final report to the Army Research Institute on contract MDA 903-93-K-0008. Technical report, Northwestern University:.

Index