1-1-1988

Brief Mood Introspection Scale (BMIS): Open-Source Code

John D. Mayer
University of New Hampshire, Durham, jack.mayer@unh.edu

Comments

For manuals, data, and other publications visit the UNH Personality Lab Collection.

Recommended Citation

The authors give their permission for general research use. Please, though, credit the original article as the source for the scale. The proper APA citation is: Mayer, J. D., & Gaschke, Y. N. (1988). The experience and meta-experience of mood. Journal of Personality and Social Psychology, 55, 102-111.

```
######################################## PART 1 Basics and
Preliminaries
######################################## PART 1 Basics and
Preliminaries
######################################## PART 1 Basics and
Preliminaries
######################################## PART 1 Basics and
Preliminaries

#               SOME BEGINNING FILES, DOCUMENTATION, STARTING COMMANDS
#               SOME BEGINNING FILES, DOCUMENTATION, STARTING COMMANDS
#               SOME BEGINNING FILES, DOCUMENTATION, STARTING COMMANDS
#               SOME BEGINNING FILES, DOCUMENTATION, STARTING COMMANDS

# NOTE BENE: You will need to change the commented line below that
begins "# setwd" to indicate
# a valid directory in which you would like to work (if you have not
done so already).
# To do so, remove the number sign and insert the revised computer
address of the working directory
# of your choice.
setwd ("C:/Users/jdmayer/Box Sync/SFW/BMIS/2018-07-20-BMISAnly")

######################## We'll redirect this week's work to an output
file called "outputfile.txt"
######################## You'll still see output in the console, but
it will also go to that output file.

#the output file ensures all the output will be preserved
sink("outputfile.txt")

#               TABLE OF DATA FILES FOR THIS CODE
#               TABLE OF DATA FILES FOR THIS CODE
#               TABLE OF DATA FILES FOR THIS CODE
#               TABLE OF DATA FILES FOR THIS CODE

# Throughout this code I will be manipulating data and creating new
data "objects"
# That said, a few files will be employed repeatedly. These are:

# Data file name      Data file description
# BMIS1998R.txt       starting file from Mayer & Gaschke 1999, 465
cases
# dtorig              the object (aka, data.frame), or (in this case)
matrix that BMIS1998R.txt is read into
# dtfactr             the dtitms file with missing data removed (i.e.,
all participants with
```

```
#                     complete responses for the 16 items), and ready
for correlational/factor analysis

#               LOAD PACKAGES USED IN THIS CODE
#               LOAD PACKAGES USED IN THIS CODE
#               LOAD PACKAGES USED IN THIS CODE
#               LOAD PACKAGES USED IN THIS CODE

# A list of the packages to find, copy and load in
# order to carry out what we need to do follows

# "psych"       The psych package allows us a convenient means of
carrying out basic exploratory factor analyses
# "lavaan"      Lavaan allows for confirmatory factor analysis

install.packages("SparseM")
library(SparseM)
install.packages("car")
library(car)
install.packages("psych")
library(psych)
install.packages("lavaan")
library(lavaan)
install.packages("GPArotation")
library(GPArotation)

#                READ IN DATA FILE AND LOOK AT IT
#                READ IN DATA FILE AND LOOK AT IT
#                READ IN DATA FILE AND LOOK AT IT
#                READ IN DATA FILE AND LOOK AT IT
#                READ IN DATA FILE AND LOOK AT IT


# read in data file
dtorig <- read.table("BMIS1998R.txt", header = TRUE, sep="")

#see the beginning of the file
head (dtorig)

#see the end of the file
tail (dtorig)

#                SUMMARIZE THE DATA
#                SUMMARIZE THE DATA
#                SUMMARIZE THE DATA
#                SUMMARIZE THE DATA

#find some descriptive statistics for the data
```

```
summary (dtorig)

#                BUILD SOME VARIATIONS OF THE DATA FILE
#                BUILD SOME VARIATIONS OF THE DATA FILE
#                BUILD SOME VARIATIONS OF THE DATA FILE
#                BUILD SOME VARIATIONS OF THE DATA FILE

# set up the "dtritems" file--a copy of dtorig that will hold re-coded
items
dtritems <-(dtorig)

#                CREATE REVERSE-SCORED VERSIONS OF THE 16 MOOD ITEMS
#                CREATE REVERSE-SCORED VERSIONS OF THE 16 MOOD ITEMS
#                CREATE REVERSE-SCORED VERSIONS OF THE 16 MOOD ITEMS
#                CREATE REVERSE-SCORED VERSIONS OF THE 16 MOOD ITEMS
#                CREATE REVERSE-SCORED VERSIONS OF THE 16 MOOD ITEMS
#                CREATE REVERSE-SCORED VERSIONS OF THE 16 MOOD ITEMS


# start by copying lively into its own vector (a column)
vlively <-dtorig[,'lively']

# look at the result
head (vlively)

# reverse code it and look at it to make sure it is right
rlively = recode(vlively, '1=4; 2=3; 3=2; 4=1')
head (rlively)

# now add the new vector into a new datafile
# the dollar sign in the next command is shorthand for a column in dt2
dtritems$rlively <- rlively

# repeat process for remaining 15 adjectives
vhappy <-dtorig[,'happy']
vsad <-dtorig[,'sad']
vtired <-dtorig[,'tired']
vcaring <-dtorig[,'caring']
vcontent <-dtorig[,'content']
vgloomy <-dtorig[,'gloomy']
vjittery <-dtorig[,'jittery']
vdrowsy <-dtorig[,'drowsy']
vgrouchy <-dtorig[,'grouchy']
vpeppy <-dtorig[,'peppy']
vnervous <-dtorig[,'nervous']
vcalm <-dtorig[,'calm']
vloving <-dtorig[,'loving']
vfedup <-dtorig[,'fedup']
```

```
vactive <-dtorig[,'active']

rhappy = recode(vhappy, '1=4; 2=3; 3=2; 4=1')
rsad = recode(vsad, '1=4; 2=3; 3=2; 4=1')
rtired = recode(vtired, '1=4; 2=3; 3=2; 4=1')
rcaring = recode(vcaring, '1=4; 2=3; 3=2; 4=1')
rcontent = recode(vcontent, '1=4; 2=3; 3=2; 4=1')
rgloomy = recode(vgloomy, '1=4; 2=3; 3=2; 4=1')
rjittery = recode(vjittery, '1=4; 2=3; 3=2; 4=1')
rdrowsy = recode(vdrowsy, '1=4; 2=3; 3=2; 4=1')
rgrouchy = recode(vgrouchy, '1=4; 2=3; 3=2; 4=1')
rpeppy = recode(vpeppy, '1=4; 2=3; 3=2; 4=1')
rnervous = recode(vnervous, '1=4; 2=3; 3=2; 4=1')
rcalm = recode(vcalm, '1=4; 2=3; 3=2; 4=1')
rloving = recode(vloving, '1=4; 2=3; 3=2; 4=1')
rfedup = recode(vfedup, '1=4; 2=3; 3=2; 4=1')
ractive = recode(vactive, '1=4; 2=3; 3=2; 4=1')

# now add the columns into the new data.frame (object) dtritems
# and check the result
dtritems$rhappy <- rhappy
dtritems$rsad <- rsad
dtritems$rtired <- rtired
dtritems$rcaring <- rcaring
dtritems$rcontent <- rcontent
dtritems$rgloomy <- rgloomy
dtritems$rjittery <- rjittery
dtritems$rdrowsy <- rdrowsy
dtritems$rgrouchy <- rgrouchy
dtritems$rpeppy <- rpeppy
dtritems$rnervous <- rnervous
dtritems$rcalm <- rcalm
dtritems$rloving <- rloving
dtritems$rfedup <- rfedup
dtritems$ractive <- ractive

head (dtritems)
tail (dtritems)

#                                          HANDLE MISSING DATA
#                                          HANDLE MISSING DATA
#                                          HANDLE MISSING DATA
#                                          HANDLE MISSING DATA
#                                          HANDLE MISSING DATA
#                                          HANDLE MISSING DATA

# This creates a new version of the datafile, dtfactr in which only
the test items are included
```

```
# and only those with no missing data
# Omit subjno, age, sex, and overall (latter because all the data is
missing)
dttemp1 <-subset(dtorig, select=-c(group, subjno, age, sex, overall))
head (dttemp1)

# Now ready to create dtfactr--basically, a file of complete items, by
removing the remaining cases with missing data and describe it
dtfactr <- na.omit(dttemp1)
describe(dtfactr)

ct <-(fa(dtfactr,2))

# If I use "summary" to summarize the data, I notice it lacks an
overall N, so I'll switch to "describe" from psych
summary (dtfactr)
describe(dtfactr)

# The describe function (from "psych" has a specific argument for
handling missing data. So, I could
# use the "describe" function itself to see what the original (and
reverse-coded) data looks like
# when excluding the missing data.
describe (dtritems, na.rm = TRUE, check=TRUE)

############################################################# PART 2
Correlations and Reliabilities
############################################################# PART 2
Correlations and Reliabilities
############################################################# PART 2
Correlations and Reliabilities
#                        CORRELATIONAL ANALYSES
#                        CORRELATIONAL ANALYSES
#                        CORRELATIONAL ANALYSES
#                        CORRELATIONAL ANALYSES

# Using what we have learned, let's conduct some correlational
analyses
# First, I'll copy four variables--the targeted items--into a
data.frame using the select command
# I make sure one is reverse-scored. I want to double-check that it
correlates r = -1.o with its original version
# the new data object will be "dt4corrs" or, data "for" correlations

dt4corrs <-subset(dtfactr, select=c(happy, lively, active, rlively))
describe (dt4corrs)
cor (dt4corrs)
```

```
# Or, you can use the function from the psych package--"use" is a
missing values option
# Remember that R is case sensitive
lowerCor(dt4corrs, digits=2, use="pairwise")

#                        RELIABILITY ANALYSES
#                        RELIABILITY ANALYSES
#                        RELIABILITY ANALYSES
#                        RELIABILITY ANALYSES

#Now, let's find the reliability of the pleasant-unpleasant scale.
# According to Mayer & Gashke (1988), the scale is composed of
# active, calm caring, content, happy, lively, loving and peppy, and
(reversed)
# drowsy, fedup, gloomy, grouchy, jittery, nervous, sad, and tired.
The reliability is supposed
# to be .83 according to the article. Is it?
# To find out, we'll first set up a file with the targeted items.

plsunp <-subset(dtritems, select=c(active, calm, caring, content,
happy, lively, loving,
                                   peppy, rdrowsy, rfedup, rgloomy,
rgrouchy, rjittery, rnervous, rsad, rtired))
head (plsunp)

#and now, the alpha

alpha(plsunp)

# Note 1: The default is to use pairwise correlations when missing
data is present
# Note 2: Although we created reverse-scored versions of the items
ourselves, in the alpha
# procedure you can use the "keys" argument to reverse-key items like
this:
# First, you set up a new data file with the 16 original items.
plsunp2 <-subset(dtritems, select=c('lively', 'happy', 'sad', 'tired',
'caring',
                                    'content', 'gloomy', 'jittery',
'drowsy', 'grouchy',
                                    'peppy', 'nervous', 'calm',
'loving', 'fedup', 'active'))

# Second, set up a vector of columns to be reversed:
reversethese <- c('drowsy', 'fedup', 'gloomy', 'grouchy', 'jittery',
'nervous', 'sad', 'tired')

# Third, run alpha with the additional argument as shown:
```

```
alpha(plsunp2, keys=reversethese)

######################################################################
# PART 3 Saving a handy file
######################################################################
# PART 3 Saving a handy file
######################################################################
# PART 3 Saving a handy file

#                        OUTPUT FILES WE WOULD LIKE TO KEEP
#                        OUTPUT FILES WE WOULD LIKE TO KEEP
#                        OUTPUT FILES WE WOULD LIKE TO KEEP
#                        OUTPUT FILES WE WOULD LIKE TO KEEP

# direct output to a file in the working directory, then read it back
in again and check it
head (dtritems)
write.table(dtritems, "dtritems_BMIS_N=465.txt", append=FALSE)
write.table(dtritems, "dtritems_BMIS_N=465.txt", append=FALSE)
checkfile <- read.table("dtritems_BMIS_N=465.txt", header = TRUE,
sep="")
head (checkfile)
################################################################# PART
2 Factor Analysis
################################################################# PART
2 Factor Analysis
################################################################# PART
2 Factor Analysis
################################################################# PART
2 Factor Analysis

# Here, I'll demonstrate a few prelimnaries recommended by William
Revelle, the author of "Psych"
# that definitely have a fun aspect to them.
# 1. pairs.panels creates a rather weird talbe with the correlations
in the upper triangular portion
# of the matrix and thumbnail scatterplots of each correlation in the
lower triangular portion
# 2. lowerCor is just the correlation matrix (I would generally have
looked at this early-on)
# 3. corPLot creates a heat-map believe it or not of the correlations
(the most fun)
# Revelle also suggests iclust, a cluster analysis, but we are doing
factor analysis here, and that is
# a different approach so I don't include it here, although if you are
interested it is iclust(r.mat=dtfactr)
# Honorable mention, but more complex to implement/explain right away,
is a bifactor model via the
```

```
# coefficient omega approach "> omega(dtfactr)" that checks for a
hierarchical model.

pairs.panels(dtfactr)
lowerCor(dtfactr)
corPlot(dtfactr)
# finally, a parallel plot to illustrate the possible number of
factors
fa.parallel(dtfactr[1:6], main="Parallel analysis of BMIS")


#                                                   EXPLORATORY FACTOR
ANALYSIS
#                                                   EXPLORATORY FACTOR
ANALYSIS
#                                                   EXPLORATORY FACTOR
ANALYSIS
#                                                   EXPLORATORY FACTOR
ANALYSIS
#                                                   EXPLORATORY FACTOR
ANALYSIS
#                                                   EXPLORATORY FACTOR
ANALYSIS


# The basic factor analysis program for R is called factanal
# and produces a maximum likelihood factor analysis

# First, we'll need a file of the original items (no reverse scoring,
and nothing but the items themselves)
# We have created it above: dtfactr

#                            ATTEMPT AT ORIGINAL 1998 PRINCIPLE AXIS
EXTRACTION
#                            ATTEMPT AT ORIGINAL 1998 PRINCIPLE AXIS
EXTRACTION
#                            ATTEMPT AT ORIGINAL 1998 PRINCIPLE AXIS
EXTRACTION
#                            ATTEMPT AT ORIGINAL 1998 PRINCIPLE AXIS
EXTRACTION
#                            ATTEMPT AT ORIGINAL 1998 PRINCIPLE AXIS
EXTRACTION

# Checking the original report (and the recovered data)
# Checking the original report (and the recovered data)
# Checking the original report (and the recovered data)
```

```
# To match the original 1988 paper, we need a principle axis
extraction
#fa is the factor analysis program developed by Bill Revelle
#check out this principal axis factoring and compare to SPSS

pa <- fa(dtfactr, 2, fm="pa", rotate="none")
print (pa)
plot(pa)
# Is it similar to the original report?

# To check the eigenvalues (helpful for a scree plot),
# we call on the nFactors program (loaded as a part of psych),
# and ask for the eigenvalues

ev <-eigen(cor(dtfactr))
ev

#the first set correspond to the eigenvalues of the factors

# A circumplex representation
# A circumplex representation
# A circumplex representation
# A circumplex representation


# For fun, let's take the loadings and see if they fit a circumplex
structure--
# a simple structure around a circle
circfact <-(fa(dtfactr,2))
circfact
plot(circfact, title="circumplex Structure")
ct <-circ.tests(circfact, loading = TRUE)
ct
# check out the plot that appears (click on the plot tab in R Studio)

#                            PRINCIPLE AXIS EXTRACTIONS
#                            PRINCIPLE AXIS EXTRACTIONS
#                            PRINCIPLE AXIS EXTRACTIONS
#                            PRINCIPLE AXIS EXTRACTIONS

# The 2-factor shows the JPSP solution. The 3 and 4 factor shows a
couple other solutions
# we looked at at the time
pa1 <- fa(dtfactr, 1, fm="pa", rotate="none")
print (pa1)

pa2 <- fa(dtfactr, 2, fm="pa", rotate="none")
print (pa2)
```

```
pa3 <- fa(dtfactr, 3, fm="pa", rotate="none")
print (pa3)

pa4 <- fa(dtfactr, 4, fm="pa", rotate="none")
print (pa4)

#                             MAXIMUM LIKELIHOOD
#                             MAXIMUM LIKELIHOOD
#                             MAXIMUM LIKELIHOOD
#                             MAXIMUM LIKELIHOOD
#                             MAXIMUM LIKELIHOOD

#nowadays, maximum likelihood often is conducted. Here is an ML
analysis for 2, 3, and 4 factors


ml1 <- fa(dtfactr, 1, fm="ml", rotate="none")
print (ml1)

ml2 <- fa(dtfactr, 2, fm="ml", rotate="none")
print (ml2)

ml3 <- fa(dtfactr, 3, fm="ml", rotate="none")
print (ml3)

ml4 <- fa(dtfactr, 4, fm="ml", rotate="none")
print (ml4)

ml5 <- fa(dtfactr, 5, fm="ml", rotate="none")
print (ml5)

# End of code for exploratory fa

#                              CATEGORICAL DATA VERSION--MAXIMUM
LIKELIHOOD
#                              CATEGORICAL DATA VERSION--MAXIMUM
LIKELIHOOD
#                              CATEGORICAL DATA VERSION--MAXIMUM
LIKELIHOOD
#                              CATEGORICAL DATA VERSION--MAXIMUM
LIKELIHOOD
#                              OPTIONAL

facat1 = fa.poly(dtfactr, fm="wls", nfactors=1, rotate="oblimin")
facat1

facat2 = fa.poly(dtfactr, fm="wls", nfactors=2, rotate="oblimin")
```

```
facat2

facat3 = fa.poly(dtfactr, fm="wls", nfactors=3, rotate="oblimin")
facat3

facat4 = fa.poly(dtfactr, fm="wls", nfactors=4, rotate="oblimin")
facat4

facat5 = fa.poly(dtfactr, fm="wls", nfactors=5, rotate="oblimin")
facat5
#                     A GOOD "EXPRESS" SOLUTION IN LAVAAN BEGINNING
WITH THE SCALES IN JPSP 1998
#                     A GOOD "EXPRESS" SOLUTION IN LAVAAN BEGINNING
WITH THE SCALES IN JPSP 1998
#                     A GOOD "EXPRESS" SOLUTION IN LAVAAN BEGINNING
WITH THE SCALES IN JPSP 1998

# If we use a simplified model, keeping items loading on one scale,
this works
# These assignments are based on placing items on the factor on which
they load highest,
# and that meet the criterion of > .35.
# It omits items that load near-equally on both scales. For details
see the accompanying
# "Technical Lab Documentation" document, Table 4.2


####  STEP 1
####  STEP 1
####  STEP 1
####  STEP 1

twofact.model <- 'f1 =~ lively + happy + content + peppy + active
f2 =~ gloomy + jittery + grouchy + fedup + sad'
#then you fit your model in a particular way (using confirmatory
factor analysis)
fit <- cfa(twofact.model, data = dtfactr)
#then you take a look at the results
summary (fit, fit.measures=TRUE, modindices = TRUE)

# For a better fit, I'll try a three-factor model based on the three-
factor ML solution
# All items on a scale (a) load highest on the scale (b) > |.35|, (c)
< |.40| on any other
# scale (See Technical-Lab-Documentation-BMIS-Analyses-2016-09-05-
1631)

####  STEP 2
```

```
####  STEP 2
####  STEP 2
####  STEP 2


threefact.model <- 'f1 =~ lively + happy + tired + drowsy + peppy +
active
f2 =~ sad + content + gloomy + jittery + grouchy + nervous + calm +
fedup
f3 =~ caring + loving'
#then you fit your model in a particular way (using confirmatory
factor analysis)
fit <- cfa(threefact.model, data = dtfactr)
#then you take a look at the results
summary (fit, fit.measures=TRUE, modindices = TRUE)

# Next approach: Drop problematic items


# two factor model--dropping the third factor--without "content"
(identified through MIs) and happy on both factors
twofact.model <- 'f1 =~ lively + happy + peppy + active
f2 =~ gloomy + jittery + grouchy + fedup + sad + happy'
#then you fit your model in a particular way (using confirmatory
factor analysis)
fit <- cfa(twofact.model, data = dtfactr)
#then you take a look at the results
summary (fit, fit.measures=TRUE, modindices = TRUE)

# Much better fit: Do we still have a good alpha?

f1 <-subset(dtritems, select=c('lively', 'happy', 'peppy','active'))
f2 <-subset(dtritems, select=c('sad', 'gloomy', 'jittery', 'grouchy',
'fedup', 'rhappy'))

alpha(f1)
alpha(f2)

sink()
```