

www.testarchiv.eu

Open Test Archive

Repositorium für Open-Access-Tests

Auswertungssyntax:

GHB-56

General Health Behavior Scale – (Gesundheitseinstellungsskala)

Kaiser, F. G. & Kibbe, A. (2021)

Kaiser, F. G. & Kibbe, A. (2021). GHB-56. General Health Behavior Scale – (Gesundheitseinstellungsskala) [Verfahrensdokumentation, Fragebogen deutsch und englisch, Auswertungssyntax]. In Leibniz-Institut für Psychologie (ZPID) (Hrsg.), Open Test Archive. Trier: ZPID.
<https://doi.org/10.23668/psycharchives.4543>

Alle Informationen und Materialien zu dem Verfahren finden Sie unter:

<https://www.testarchiv.eu/de/test/9008198>

Verpflichtungserklärung

Bei dem Testverfahren handelt es sich um ein Forschungsinstrument, das der Forschung, Lehre und Praxis dient. Es wird vom Testarchiv online und kostenlos zur Verfügung gestellt und ist urheberrechtlich geschützt, d. h. das Urheberrecht liegt weiterhin bei den AutorInnen.

Mit der Nutzung des Verfahrens verpflichte ich mich, die Bedingungen der [Creative Commons Lizenz CC BY-SA 4.0](#) zu beachten. Ich werde nach Abschluss meiner mit dem Verfahren zusammenhängenden Arbeiten mittels des [Rückmeldeformulars](#) die TestautorInnen über den Einsatz des Verfahrens und den damit erzielten Ergebnissen informieren.

Terms of use

The test instrument is a research instrument that serves research, teaching and practice. It is made available online and free of charge by the test archive and is protected by copyright, i.e. the copyright remains with the author(s). By using this test, I agree to abide by the terms of the [Creative Commons License CC BY-SA 4.0](#). After completion of my work with the measure, I will inform the test authors about the use of the measure and the results I have obtained by means of the [feedback form](#).

```
#####
```

```
##### reading dataset from an SPSS file #####
```

```
#####
```

```
if (!require("rstudioapi")) install.packages("rstudioapi")
```

```
library(rstudioapi)
```

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path)) # sets working directory to path of this script
```

```
if (!require("foreign")) install.packages("foreign")
```

```
library(foreign)
```

```
data <- read.spss("gesamtsample_n349_(2).sav", to.data.frame = T) # replace dataset name if you analyse your own data.
```

```
if (!require("varhandle")) install.packages("varhandle")
```

```
library(varhandle)
```

```
items <- unfactor(data[ c(5:32,55:82) ]) # please select the columns of your items in the dataset by changing the expression --- data[...] ---
```

```
#####
```

```
##### recode responses in word-form into numeric values #####
```

```
#####
```

```
# Please be aware that this script handles response options in numeric form. Response labels
```

```
# of polytomous items should be numeric values ranging from 1 to 5, or "NA" for missing values.
```

```
# Dichotomous items should be coded 0 for "no" and 1 for "yes".
```

```
# If your dataset contains response values as words (e.g. "very often") instead of using numeric values (e.g. 5) please
```

```
# uncommment and use command rows 46 and 48 according to your language and the structure of your dataset to
```

recode your response values into numeric form.

To check in the Console of R-Studio if your response values are coded as numerics please execute the following command row.

```
str(items)
```

```
if (!require("car")) install.packages("car")
```

```
library(car)
```

```
### recoding of all polytomous items (columns 29 to 56 in demonstration dataset) ###
```

```
items[c(29:56)] <- as.data.frame(sapply(items[c(29:56)], recode, "'nie'=1; 'selten'=2; 'gelegentlich'=3; 'oft'=4; 'sehr oft/ immer'=5; NA=NA"))
```

```
### recoding of alldichotomous items (columns 1 to 28 in demonstration dataset) ###
```

```
items[c(1:28)] <- as.data.frame(sapply(items[c(1:28)], recode, "'nein'=0; 'ja'=1; NA=NA"))
```

```
#####
```

```
##### recode all expressions for missing values into "NA" #####
```

```
#####
```

Please check in the Console of R-Studio if your dataset now consists of variables either having 2 or 5 different values (excluding "NA")

Polytomous items should have 5 levels and dichotomous items should have 2 levels.

If items have more levels this could indicate, that missing values are not coded as NA, but instead as -99/9/etc.

If this is the case, please adapt and run the command row 58 to recode e.g. -99 to NA.

```
itemsTest <- as.data.frame(items)
```

```
str(itemsTest)
```

```
rm(itemsTest)
```

```
items[c(13,14,19,20,33,35)] <- as.data.frame(sapply(items[c(13,14,19,20,33,35)], recode, "'kann ich nicht beantworten'=NA"))
```

```
#####
```

```
##### dichotomize response values #####
```

```
#####
```

```
# All reponses are dichotomized before Rasch analysis.
```

```
#####
```

```
# In standard polytomous items, the two highest response options (representing the response options  
"often" and "very often")
```

```
# are coded as 1 representing health-benefital behavior.
```

```
# In inverse polytomous items the two lowest response options (representing the response options "never"  
and "seldom") are
```

```
# coded as 1 representing health-benefital behavior.
```

```
#####
```

```
# Standard dichotomous items are not recoded and remain as 0 for disagreement with the item and 1 for  
agreement with
```

```
# the item (representing health-benefital reponses).
```

```
# Inverse dichotomous items are recoded into 0 for agreement with item and 1 for disagreement with
```

```
# the item (representing health-benefital responses).
```

```
#####
```

```
# Inverse polytomous items in our survey are: P4, P6, P7, P10, P11, P16 and P19. Inverse dichotomous  
items in our survey are: D9, D12 and D13.
```

```
# If you are using your own dataset please adapt the command rows which dichotomize the response  
values according to the structure
```

```
# of your own dataset by changing the numbers which represent the columns of your items in command  
rows 85 and 89. --- items[c(...)] ---
```

```
#####
```

```
# Please note, in some cases, the values of your dataset may already be inverted. If this is the case, please  
dichotomize
```

```
# all variables according to the standard items and dont run the command rows for inverse items.
```

```
### inverse dichotomous items ###
```

```
items[c(37,40,41)] <- as.data.frame(sapply(items[c(37,40,41)], recode, "0=1; 1=0; NA=NA"))
```

```
### inverse polytomous items ###
```

```
items[c(4,6,7,10,11,16,19)] <- as.data.frame(sapply(items[c(4,6,7,10,11,16,19)], recode, "1=5; 2=4; 3=3; 4=2; 5=1; NA=NA"))
```

```
### standard dichotomous items ###
```

```
items[c(29:56)] <- as.data.frame(sapply(items[c(29:56)], recode, "0=0; 1=1; NA=NA"))
```

```
### standard polytomous items ###
```

```
items[c(1:28)] <- as.data.frame(sapply(items[c(1:28)], recode, "1=0; 2=0; 3=0; 4=1; 5=1; NA=NA"))
```

```
#####
```

```
##### Rasch analysis #####
```

```
#####
```

```
if (!require("eRm")) install.packages("eRm")
```

```
library(eRm)
```

```
items <- unfactor(items)
```

```
RaschModel <- RM(items) # this might take a while
```

```
pp <- person.parameter(RaschModel)
```

```
#####
```

```
##### Obtain item parameters #####
```

```
#####
```

```
### Create item output dataset ###
```

```
item_estimates <- data.frame(colnames(items))
```

```
names(item_estimates) <- "item"
```

```
item_estimates$item <- as.character(item_estimates$item)
```

```
### find items with zero or full score responses ###
```

```
items_perfect <- colnames(items[colSums(items, na.rm=T) == colSums(!is.na(items))])
```

```
items_zero <- colnames(items[colSums(items, na.rm=T) == 0])
```

```

items_excluded <- c(items_zero, items_perfect)

if(length(items_excluded)==0){items_excluded<-"nothing"} #if there are no items with zero or perfect
score, the output list remains complete

item_estimates <- as.data.frame(item_estimates[!item_estimates$item == items_excluded,]) #this line
removes items with full or zero score from output list

names(item_estimates) <- "item"

item_estimates$item <- as.character(item_estimates$item)

### Obtain item difficulty ###

item_estimates['GEB.diff'] <- round(RaschModel$betapar*(-1),3) # betapar is the item easiness; inverting it
by *(-1) results in item difficulty.

### item difficulty standard error ###

item_estimates['Std.Error'] <- round(RaschModel$se.beta,3)

### Obtain item fit values ###

ifit <- itemfit(pp)

item_estimates['infit MS'] <- round(ifit$i.infitMSQ, 3) # meansquare
item_estimates['outfit MS'] <- round(ifit$i.outfitMSQ, 3)
item_estimates['infit t'] <- round(ifit$i.infitZ, 3) # z-standardized
item_estimates['outfit t'] <- round(ifit$i.outfitZ, 3)

### Obtain additional item parameters ###

item_adds <- data.frame()

item_adds[1,1] <- "mean of item estimates"
item_adds[1,2] <- round(mean(item_estimates$GEB.diff),3)
item_adds[2,1] <- "standard diviation of item estimates"
item_adds[2,2] <- round(sd(item_estimates$GEB.diff),3)
item_adds[4,1] <- "mean of infit MS"
item_adds[4,2] <- round(mean(item_estimates$`infit MS`),3)
item_adds[5,1] <- "standard diviation of infit MS"
item_adds[5,2] <- round(sd(item_estimates$`infit MS`),3)
item_adds[7,1] <- "mean of infit t"

```

```

item_adds[7,2] <- round(mean(item_estimates$`infit t`),3)
item_adds[8,1] <- "standard diviation of infit t"
item_adds[8,2] <- round(sd(item_estimates$`infit t`),3)
item_adds[10,1] <- "min of infit MS"
item_adds[10,2] <- min(item_estimates$`infit MS`)
item_adds[11,1] <- "max of infit MS"
item_adds[11,2] <- max(item_estimates$`infit MS`)
item_adds[13,1] <- "items with perfect score"
item_adds[13,2] <- paste(items_perfect, collapse = ' , ')
item_adds[14,1] <- "items with zero score"
item_adds[14,2] <- paste(items_zero, collapse = ' , ')
item_adds[16,1] <- "item seperation reliability"
item_adds[16,2] <- round( (var(item_estimates$GEB.diff, na.rm=T) - sum((item_estimates$Std.Error)^2,
na.rm=T) / sum(!is.na(item_estimates$Std.Error)))/var(item_estimates$GEB.diff, na.rm=T) ,3)

### add removed items to output file ###
items_excluded_bind <- data.frame(items_excluded,NA,NA,NA,NA,NA,NA)
names(items_excluded_bind) <- colnames(item_estimates)
if (!items_excluded_bind[1,1] == "nothing") item_estimates <- rbind(as.data.frame(item_estimates),
as.data.frame(items_excluded_bind))

#####

##### Obtain person parameters #####

#####

### Create person output dataset ###
person_estimates <- data[2] #please select the column of the subjects' IDs in the orginial dataset
names(person_estimates) <- "ID"
person_estimates$ID <- as.character(person_estimates$ID)

### find persons with zero or perfect score ###
person_perfect <- rownames(items[rowSums(items, na.rm=T) == rowSums(!is.na(items)),])

```

```

person_perfect <- person_estimates[rownames(person_estimates) %in% person_perfect,]
person_zero <- rownames(items[rowSums(items, na.rm=T) == 0,])
person_zero <- person_estimates[rownames(person_estimates) %in% person_zero,]
persons_excluded <- c(person_zero, person_perfect)

if(length(persons_excluded)==0){persons_excluded<-"noone"} #if there are no persons with zero or perfect
score, the output list remains complete

```

Obtain person estimates (i.e. logit value representing individual environmental attitude)

```

person_estimates['GEB.est'] <- person.parameter(RaschModel)$theta.table
person_estimates['GEB.est'] <- round(person_estimates['GEB.est'], 3)

person_estimates <- as.data.frame(person_estimates[!person_estimates$ID %in% persons_excluded,]) #
remove persons from output list

names(person_estimates[1]) <- "ID"

person_estimates$ID <- as.character(person_estimates$ID)

```

person estimate standard error

```

error <- pp$se.theta
error <- unlist(error, use.names=T) # create a list of all Std.errors and their namelabels
names(error) <- sub('.*\\P', "", names(error))
names(error) # only keep number of datarow from original data set from the namelabels of Std.errors
error.frame <- data.frame(error)

error.frame['person'] <- as.numeric(names(error)) # create a data frame with Std.errors and extracted
datarow numbers

error.frame <- error.frame[order(error.frame$person),] # sort Std.errors by datarow number
person_estimates['Std.Error'] <- round(error.frame, 3) # match Std.errors to the Output file

```

Obtain person fit values

```

pfit <- personfit(pp)

person_estimates['infit MS'] <- round(pfit$p.infitMSQ, 3)
person_estimates['outfit MS'] <- round(pfit$p.outfitMSQ, 3)
person_estimates['infit t'] <- round(pfit$p.infitZ, 3)
person_estimates['outfit t'] <- round(pfit$p.outfitZ, 3)

```



```
### Obtain additional person parameters ###
```

```
person_adds <- data.frame()
person_adds[1,1] <- "mean of person estimates"
person_adds[1,2] <- round(mean(person_estimates$GEB.est),3)
person_adds[2,1] <- "standard diviation of person estimates"
person_adds[2,2] <- round(sd(person_estimates$GEB.est),3)
person_adds[4,1] <- "mean of infit MS"
person_adds[4,2] <- round(mean(person_estimates$`infit MS`),3)
person_adds[5,1] <- "standard diviation of infit MS"
person_adds[5,2] <- round(sd(person_estimates$`infit MS`),3)
person_adds[7,1] <- "mean of infit t"
person_adds[7,2] <- round(mean(person_estimates$`infit t`),3)
person_adds[8,1] <- "standard diviation of infit t"
person_adds[8,2] <- round(sd(person_estimates$`infit t`),3)
person_adds[10,1] <- "% Misfit (z-value > 1.96) of subjects"
person_adds[10,2] <- PersonMisfit(pp)
person_adds[10,2] <- round(as.numeric(person_adds[10,2]),3)
person_adds[12,1] <- "persons with perfect score (ID)"
person_adds[12,2] <- paste(person_perfect, collapse = ' , ')
person_adds[13,1] <- "persons with zero score (ID)"
person_adds[13,2] <- paste(person_zero, collapse = ' , ')
person_adds[15,1] <- "person seperation reliability"
person_adds[15,2] <- round(SepRel(person.parameter(RaschModel))$sep.rel, 3)
```

```
### add removed persons to output file ###
```

```
persons_excluded_bind <- data.frame(persons_excluded,NA,NA,NA,NA,NA,NA)
names(persons_excluded_bind) <- colnames(person_estimates)
if (!persons_excluded_bind[1,1] == "noone") person_estimates <- rbind(as.data.frame(person_estimates),
as.data.frame(persons_excluded_bind))
```

```
#####
```

```
##### create Plot #####
```

```
#####
```

```
### item-person-map ###
```

```
if (!require("WrightMap")) install.packages("WrightMap")
```

```
library(WrightMap)
```

```
wrightMap(person_estimates$GEB.est, item_estimates$GEB.diff,item.side = itemHist,  
  person.side = personHist, main.title = "Wright Map GEB", min.logit.pad = 0.25,  
  max.logit.pad = 0.25, min.l = NULL, max.l = 4, item.prop = 0.5, return.thresholds = F,  
  new.quartz = T)
```

```
#####
```

```
##### save analysis outputs #####
```

```
#####
```

```
write.csv2(person_estimates, "person_estimates1.csv")
```

```
write.csv2(person_adds, "person_estimates2.csv")
```

```
write.csv2(item_estimates, "item_estimates1.csv")
```

```
write.csv2(item_adds, "item_estimates2.csv")
```