Psychology 350: Advanced statistics and programming in R
UserRs versus ProgrammeRs

William Revelle

Department of Psychology
Northwestern University
Evanston, Illinois USA

April, 2024

Outline

UserRs and ProgrammeRs

1. There are many ways to use R
2. It is important to become a useR before becoming a programmeR
3. This requires learning how to work through various examples
4. Using functions
5. Writing short scripts which combine functions
6. Only then should we try to develop programs
7. Lets work through various scripts for doing basic descriptions
8. And then various scripts for doing reliability analysis.

R as a useful tool

1. For simple data analysis and graphics just a few lines of code
2. For programmatic analyses, use and keep scripts
   - This might involve a set of analysis on one set of data
   - Typically just step through the script and write more code as we go along
   - This is a way to document what you are doing in your research,
   - You are documenting the specific data set and the specific analyses you do.
3. For frequent analysis of different sets of data, write (program) a function
   - A way of organizing repeated analyses
   - Want to share a method with others.
   - Documentation of what the function does, not the specific application

Common structure of scripts and programs

1. Defining the packages you need to use (e.g.,)
   - library(psych) #for some useful tools
   - library(psychTools) #for large example data sets
2. Data entry (accessing the data)
   - Checking for correct entry
   - Basic characteristics of the data (e.g., `describe`, `pairs.panels`, `corPlot`)
3. Data processing
   - Apply some particular operation on the data
   - This might be an iterative operation or merely some closed form calculation
4. Reporting the results
   - Return all results (in a list of results)
   - But perhaps print out just the important results

Example scripts: taken from the Reliability appendix

R code

```
install.packages("psych",dependencies = TRUE)  #Just need to do this once
install.packages("psychTools")  #Just do this once as well
library(psych) #make the psych package active-- need to do this everytime you start R
library(psychTools) #if you want to use the example data sets and some convenient tools

   #Getting help
help("psych") #opens a help window overview of the packagr
help{"psychTools") #opens a help window listing the various data sets in psychTools
vignette(topic="intro",package="psych") #opens an extensive  pdf document
vignette(topic="overview",package=psychTools") #opens the second part of this vignette
?omega #opens the specific help page for e.g., the omega function
```

From Online supplement to Reliability from α to ω: A Tutorial

Data input

Several *psych* or *psychTools* fiunctions

```
my.data <- read.file()#opens an OS dependent search window and
              reads data according to the suffix
#or first copy your data to the clipboard and then
 my.data <- read.clipboard() #assumes that header information is
                   on  the first line of the file
  my.data <- read.clipboard(header=FALSE) #no header information:
                   the data start on the first line
  # choose a remote file and read it
  remote <- "https://personality-project.org/courses/350/dataets/
       simulation.txt"
  my.data <- read.file(remote)
```

Preliminary processing – checking the data

R code

```
dim(sai)   #how many rows (subjects) and columns (variables) in the built in data set sai
dim(msqR) #how many rows (subjects) and columns  variables) in the msqR data set
headTail(sai) # show the first and last 3 lines of the sai data set
```

```
dim(sai) #how many rows (subjects) and columns (variables) in the data set sai
[1] 5378   23
>  dim(msqR) #how many rows (subjects) and columns  variables) in msqR data set
[1] 6411   88
> headTail(sai) # show the first and last 3 lines of the sai data set
     study time  id calm secure tense regretful at.ease upset worrying rested anxious comfort
1     AGES    1   1    3      3     2         1       2     1         1      3       2
2     AGES    1   2    3      3     2         2       3     2         1      1       2
3     AGES    1   3    3      3     2         1       3     1         2      1       1
4     AGES    1   4    3      3     1         1       3     1         1      2       1
...   <NA>  ... ...  ...    ...   ...       ...     ...   ...       ...    ...     ...
5375  XRAY    2 197    2      2     3      <NA>    <NA>  <NA>      <NA>   <NA>    <NA>
5376  XRAY    2 198    4      4     1         1       4     1         1      1       1
5377  XRAY    2 199    4      4     1         1       4     1         1      1       1
5378  XRAY    2 200    3      3     2         2       3     2         2      2       2
     nervous jittery high.strung relaxed content worried rattled joyful pleasant
1          2       2           2       2       3       1       1      3        3
2          2       1           1       2       3       2       1      1        2
3          2       2           1       2       3       1       1      3        3
4          1       2           1       3       4       1       1      2        3
...      ...     ...         ...     ...     ...     ...     ...    ...      ...
5375    <NA>    <NA>        <NA>    <NA>    <NA>    <NA>    <NA>   <NA>     <NA>
5376       1       1           1       4       3       2       1      3        4
5377       1       1           1       3       4       1       1      3        4
5378       2       2           2       2       3       4       1      3        3
>
```

Basic structure of the msqR and sai data

```
?msqR   #ask for information about the sai data set
  Help files are meant to guide you to the data or function

  table(msqR$study,msqR$time)   #show the study names and  sample sizes


  table(msqR$study,msqR$time)

           1   2   3   4
AGES      68  68   0   0
Cart      63  63   0   0
CITY     157 157   0   0
EMIT      71  71   0   0
Fast      94  94   0   0
FIAT      70  70   0   0
FILM      95  95  95   0
FLAT     170 170 170   0
GRAY     107 107   0   0
HOME      67  67   0   0
IMPS     102 102   0   0
ITEM      49  49   0   0
Maps     160 160   0   0
MITE      49  49   0   0
MIXX      71  71   0   0
PAT       65  65  65  65
PATS     132   0   0   0
...
SWAM.one  94   0   0   0
SWAM.two  54   0   0   0
VALE      77  77  70  70
XRAY     200 200   0   0
```

SAI and MSQR are from 10 years of studies

1. Basic project was to examine the effect of emotional and physiological state on cognitive performance.
2. Many studies administered caffeine (4 mg/kg body weight) (or a placebo).
3. Some studies manipulated mood with movies (fear, sadness, happiness, control)
4. Mood data (SAI and MSQ) collected before manipulation
5. We can choose studies representing various conditions using the `subset` command.

Scripts – continued

Select some subsets for analysis using the `subset` function. Use the %in% or `is.element` functions.

R code

```
#Now, select some subsets for analysis using the subset function.
 #the short term consistency sets
 sai.control <- subset(sai,is.element(sai$study,c("Cart", "Fast", "SHED",  "SHOP")) )

 #pre and post drug studies
 sai.drug <- subset(sai,is.element(sai$study, c("AGES","SALT","VALE","XRAY")))

 #pre and post film studies
 sai.film <- subset(sai,is.element(sai$study, c("FIAT","FLAT", "XRAY") ))
 msq.control <- subset(msqR,is.element(msqR$study,c("Cart", "Fast", "SHED",  "SHOP")) )

 #pre and post drug studies
 msq.drug <- subset(msqR,is.element(msqR$study, c("AGES","CITY","EMIT","SALT",
           "VALE","XRAY")))

#pre and post film studies
 msq.film <- subset(msqR,is.element(msqR$study, c("FIAT","FLAT", "MAPS", "MIXX","XRAY") ))
 msq.films4 <- subset(msqR, is.element(msqR$study, c("FLAT", "MAPS", "XRAY") ))
 msq1 <- subset(msqR,msqR$time == 1)  #just the first day measures
 sai1 <- subset(sai,sai$time==1)  #just the first set of observations for the SAI

 #do some fancy subsetting

 sam.rim <- subset(sai,(sai$study %in% c("SAM" ,"RIM")))#choose SAM and RIM for 2 day test
 vale <- subset(sai,sai$study=="VALE") #choose the VALE study for multilevel analysis
```

Scripts: continued

R code

```
dim(msq.control)  #how many subjects and how many items?
dim(sai.control)  #show the number of subjects and items for the seco
table(sam.rim$time) #how many were in each time point
table(vale$time) #how many were repeated twice on day 1 and then on
```

```
> dim(msq1)  #how many subjects and how many items?
   [1] 3032   88
   > dim(sai1)  #show the number of subjects and items for the second subset
   [1] 3032   23
   > table(rim$time) #how many were in each time point
  1   3
666 666

> table(vale$time) #how many were repeated twice on day 1 and then on day 2
 1  2  3  4
77 77 70 70
```

Reading and writing code

1. Just as we learn to write by reading books
2. So we learn to code by reading code
3. Almost all R functions are available to read just by typing in the name of the function
4. All functions have a help page and usually examples.
5. You can run them in “debug” mode to step through the function to see what it does

The parts of a function

1. Function name and parameter list
2. Preliminaries (making sense of the parameters, getting ready to process)
3. The body of the function: do something useful
4. The results: package them into useful output, perhaps by creating lists of the results
5. return(results) and end the function.

When reading a function, see if you can identify these parts.

Reading code

1. To read the code of almost any function, just type in the name of the function without ()
2. However, this does not include the comments that have been written for that function by the programmer
3. Source Code for *psych* is in the class folder (this has the comments)
4. To see a list of all of the functions in *psych*
5. To see a particular function find out where it probably is by ? the function
6. e.g., `fisherz` is in the `fisherz` help page, so open and view that one
7. the `isCorrelation` function is part of a larger set of miscellaneous functions. (This is a collection of minor functions that are documented either in the `psych.misc` help page or separately.) Worth examing for demonstration of documenting code.

The `fisherz` functions

As set of one liners that don't really need to be documented by comments. The names speak for themselves.

```
"fisherz" <-
function(rho)  {0.5*log((1+rho)/(1-rho)) }   #converts r to z

"fisherz2r" <-
  function(z) {(exp(2*z)-1)/(1+exp(2*z)) }   #converts back again

"r2d" <-
function(rho) {2*rho/sqrt(1-rho^2)}

"d2r" <-
function(d) {d/sqrt(d^2+4)}

#added sign correction October 8, 2018
"t2r" <- function(t,df) {sign(t) * sqrt(t^2/(t^2 + df))}  #fixed April 27, 2017

"g2r" <- function(g,df,n) {sign(g) * g/sqrt(g^2 + 4*df/n)}

"chi2r" <- function(chi2,n) {sqrt(chi2/n)}

"r2chi" <- function(rho,n) { chi2 <-( rho^2 *n)}

"cor2cov" <- "r2c" <- function(rho,sigma) { sigma <- diag(sigma)
cov <- sigma %*% rho %*% sigma
colnames(cov) <- rownames(cov) <- colnames(rho)
return(cov)}
```

The `isCorrelation` function

Part of the misc.R functions.

Note how if it is non obvious, we explain what the function does and perhaps when it was developed or when it was patched.

Uses several core-R functions

```
#this just shows if it is a matrix is symmetric and has diagonals of 1
#Added the unclass to  handle a problem with class partial.r  4/10/21
"isCorrelation" <-  function(x,na.rm=FALSE) {value <- FALSE
  if(NROW(x) == NCOL(x)) {
  if( is.data.frame(x)) {if(isSymmetric(unclass(unname(as.matrix(x))))){value <- TRUE}}else {
           if(isSymmetric(unclass(unname(x)))) {value <- TRUE}}
  value <- value && isTRUE(all.equal(prod(diag(as.matrix(x))),1) )
  if(!value  && (na.rm) && any(is.na(diag(as.matrix(x))))) stop(
         "Although the matrix is symmetric, one of the elements of the  diagonal is NA.
           Check your data.")
  value <- value && isTRUE((min(x,na.rm=TRUE)>= -1) & (max(x,na.rm=TRUE) <= 1))
  }
  return(value)}

#this just shows if it is a symmetric matrix
"isCovariance" <-  function(x) {value <- FALSE
  if(NROW(x) == NCOL(x)) {
  if( is.data.frame(x)) {if(isSymmetric(unclass(unname(as.matrix(x))))) {
          value <- TRUE}} else {
        if(isSymmetric(unclass(unname(x)))) {value <- TRUE}}}
 # value <- value && isTRUE(all.equal(prod(diag(as.matrix(x))),1) )  #don't check for diagona
  return(value)}
```

The many types of reliability

1. As discussed in Revelle and Condon (2019) there are many measures of reliability:
   - Internal consistency as measured by α or ω_h or ω_t.
   - Alternate forms as found from the correlation of two forms.
   - Split half reliability
   - Measures of *unidimensionality* Revelle and Condon (2023).
   - Stability over time by measuring test-retest correlations
2. Each of these may use a different function.
3. With a little bit of code, we can stitch these together.
4. This is an example of scripting
5. Which is then followed by an example of small function to do it.

Simple script

R code

```
 my.data <- ability #  change this to match your data
alp<- alpha(my.data)
sp <- splitHalf(my.data)
om <- omega(my.data,4)
uni <- unidim(my.data)
summary(alp)
summary(sp)
uni
summary(om)
```

```
summary(alp)

Reliability analysis
 raw_alpha std.alpha G6(smc) average_r S/N    ase mean   sd median_r
      0.83      0.83    0.84      0.23 4.9 0.0064 0.51 0.25     0.21
> summary(sp)
Warning message:
In summary.psych(sp) :
  I am sorry, I do not have a summary function for this object
> uni

A measure of unidimensionality
 Call: unidim(keys = my.data)

Unidimensionality index =
       u      tau      con    alpha     av.r median.r      CFI      ECV
    0.84     0.91     0.93     0.83     0.23     0.21     0.80     0.75
```

Better: use a function which does the stitching

R code

```
reliability(my.data)
```

```
keys not specified, all items will be scored
Measures of reliability
reliability(keys = my.data)
          omega_h alpha omega.tot  Uni  tau cong max.split min.split mean.r med.r n.items CFI
All_items    0.56  0.83      0.85 0.84 0.91 0.93      0.87      0.73   0.23  0.21      16 0.8
```

Using a keys list

1. Some functions can be done on multiple subscales
2. Specify a keys list which includes direction for scoring
3. Functions taking `keys.list` include `alpha`,`reliability`, `scoreItems`
4. Functionally, this takes our script file and runs it for each of a subset of the data.

The `reliability` function

R code

```
rel <- reliability(bfi.keys,bfi)
rel
```

```
Measures of reliability
reliability(keys = bfi.keys, items = bfi)
             omega_h alpha omega.tot  Uni  tau cong max.split min.split mean.r med.r n.items
agree           0.64  0.71      0.75 0.89 0.90 0.99      0.75      0.62   0.33  0.34       5
conscientious   0.53  0.73      0.77 0.95 0.97 0.98      0.74      0.64   0.35  0.34       5
extraversion    0.55  0.76      0.82 0.97 0.97 0.99      0.78      0.66   0.39  0.38       5
neuroticism     0.71  0.81      0.85 0.93 0.95 0.98      0.83      0.69   0.47  0.41       5
openness        0.39  0.61      0.70 0.85 0.88 0.97      0.66      0.52   0.24  0.23       5
>
```

An example of using a function which combines other functions

R code

```
reliability(ability) #just one scale
reliability(bfi.keys, bfi)  #multiple scales with keying information
```

```
reliability(ability) #just one scale
keys not specified, all items will be scored
Measures of reliability
reliability(keys = ability)
          omega_h alpha omega.tot  Uni r.fit fa.fit max.split min.split mean.r med.r n.items
All_items    0.56  0.83      0.85 0.84  0.91   0.93      0.87      0.73   0.23  0.21      16
```

```
 reliability(bfi.keys, bfi)  #multiple scales with keying information
Measures of reliability
reliability(keys = bfi.keys, items = bfi)
             omega_h alpha omega.tot  Uni r.fit fa.fit max.split min.split mean.r med.r n.it
agree           0.64  0.71      0.75 0.89  0.90   0.99      0.75      0.62   0.33  0.34
conscientious   0.53  0.73      0.77 0.95  0.97   0.98      0.73      0.64   0.35  0.34
extraversion    0.55  0.76      0.82 0.97  0.97   0.99      0.78      0.71   0.39  0.38
neuroticism     0.71  0.81      0.85 0.93  0.95   0.98      0.83      0.73   0.47  0.41
openness        0.39  0.61      0.70 0.85  0.88   0.97      0.66      0.53   0.24  0.23
>
```

Reliability: Preliminaries

```
#Developed 6/15/21
#Fixed 6/20/21 to avoid the problem of scales of one item or too many to find splits
#Modified 7/10/21 to allow it to function with just a set of items, no keys specified
#Modified 7/11/21 to include the unidimensional estimates from unidim  and
     # to use the R object for speed

reliability <- function(keys=NULL,items,nfactors=2,split=TRUE,raw=TRUE,plot=FALSE,
hist=FALSE,  n.sample=10000) {
 cl <- match.call()
  result <- list()
  splits <- list()
  best <- worst <- list()
   res.name <- list()
  if(hist) raw <-TRUE
  if(raw) split <- TRUE

  #check to see if the first parameter is a list of keys, if not, then we want
     #   to do reliability on just one scale

  if(!is.null(keys)){ if(NCOL(keys)==1) { n.scales <- length(keys) } else {
      items <- keys  #this is the case where the user did not specify keys but just the data
       message("keys not specified, all items will be scored")
       n.scales  <- 1
       keys <- list()
       keys[["All_items"]] <- colnames(items)
      }} else {n.scales <- 1
   keys[["All_items"]] <- colnames(items)
     }
      if(isCorrelation(items)) {cors<- TRUE} else {cors<- FALSE}
            #is the input a correlation matrix
```

Reliability: the main loop

```
for (scales in 1:n.scales) {  #the general case
 scale.key <- keys[[scales]]   #
select <- selectFromKeys(scale.key)  #the items for this scale
if(length(select)>1 )  {
 if(cors) {om <- omegah(items[select,select], nfactors=nfactors,plot=plot,two.ok=TRUE)
         } else {
   om <- omegah(items[,select], nfactors=nfactors,plot=plot,two.ok=TRUE)}

   uni <- unidim(om$R)  #use the R object rather than redoing the factoring


 if(split){temp.keys <- colnames(om$R) <-  gsub("-","",colnames(om$R))
           sign.key <- rep("",length(select))
           sign.key[which(colnames(om$R)!= rownames(om$R))] <- "-"
            temp.keys <- paste0(sign.key,temp.keys)
  #this next line was dropped, but I keep to show my thinking
   # split.half <- suppressWarnings(splitHalf(om$R,raw=raw,brute=FALSE,n.sample=n.sample,
    #                key=temp.keys))
     #don't use temp.keys
   split.half <- suppressWarnings(splitHalf(om$R,raw=raw,brute=FALSE,n.sample=n.sample))
         best[[scales]] <- list(max=split.half$maxAB)
         worst[[scales]] <- list(min=split.half$minAB)

         result[[scales]] <- list(omega_h = om$omega_h, alpha = split.half$alpha,
                omega.tot = om$omega.tot,u=uni$u[1],av.r.fit=uni$u[2],fa.fit=uni$u[3],
                 maxrb=split.half$maxrb,minrb=split.half$minrb,
                mean.r=split.half$av.r, med.r <- split.half$med.r, n.items=length(select) )
         if(raw) splits[[scales]] <- split.half$raw} else {
  result[[scales]] <- list(omega_h = om$omega_h, alpha = om$alpha, omega.tot = om$omega.tot,
             u=uni$u[1],av.r.fit=uni$u[2],fa.fit=uni$u[3],n.items=length(select)) }
 res.name[scales] <- names(keys)[scales]
  }
 }
```

Reliability: wrapping it up

```
 #Creates a list that keeps the names
best <- unlist(unlist(best,recursive=FALSE),recursive=FALSE)
worst <- unlist(unlist(worst,recursive=FALSE),recursive=FALSE)
 # names(result) <- res.name
 if(split) {ncol <- 11} else {ncol <- 7}
  result.df <- matrix(unlist(result[!is.null(result)]), ncol=ncol,byrow=TRUE)
if(split) {   colnames(result.df) <- c("omega_h", "alpha", "omega.tot", "Uni","r.fit",
          "fa.fit","max.split","min.split","mean.r", "med.r", "n.items") } else {
colnames(result.df) <- c("omega.h", "alpha", "omega.tot", "Uni","r.fit",
              "fa.fit","n.items")}
rownames(result.df) <- unlist(res.name)
 if(raw) {
 lx <- unlist(lapply(splits,length))
 splits <- splits[lx>0]
 names(splits) <- rownames(result.df)

# splits.mat <- matrix(unlist(splits),ncol=length(keys))
# colnames(splits.mat) <- names(keys)
  class(result.df) <- c("psych","reliability", "matrix")
  names(best)<- paste(rep(res.name,each=2),names(best))
  names(worst) <- paste(rep(res.name,each=2),names(worst))
 result <- list(result.df = result.df,splits= splits,max=best,min=worst, Call = cl)
 if(hist) {multi.hist(splits)}
 class(result) <-  c("psych","reliability")
 return(result) } else {

  class(result.df) <- c("psych","reliability", "matrix")
 return(result.df)
 }
 }
 Created June 11-17, 2021
   #fixed 6/20/21 to avoid the problem of null cases
```

An example of using the function (again)

R code

```
reliability(ability) #just one scale
reliability(bfi.keys, bfi)  #multiple scales with keying information
```

```
reliability(ability) #just one scale
keys not specified, all items will be scored
Measures of reliability
reliability(keys = ability)
          omega_h alpha omega.tot  Uni r.fit fa.fit max.split min.split mean.r med.r n.items
All_items    0.56  0.83      0.85 0.84  0.91   0.93      0.87      0.73   0.23  0.21      16
```

```
 reliability(bfi.keys, bfi)  #multiple scales with keying information
Measures of reliability
reliability(keys = bfi.keys, items = bfi)
             omega_h alpha omega.tot  Uni r.fit fa.fit max.split min.split mean.r med.r n.it
agree           0.64  0.71      0.75 0.89  0.90   0.99      0.75      0.62   0.33  0.34
conscientious   0.53  0.73      0.77 0.95  0.97   0.98      0.73      0.64   0.35  0.34
extraversion    0.55  0.76      0.82 0.97  0.97   0.99      0.78      0.71   0.39  0.38
neuroticism     0.71  0.81      0.85 0.93  0.95   0.98      0.83      0.73   0.47  0.41
openness        0.39  0.61      0.70 0.85  0.88   0.97      0.66      0.53   0.24  0.23
>
```

The parts of a function

Functions can automate parts of scripts that are repeated multiple times.
Each function has:

1. function name and parameter list
2. Preliminaries (making sense of the parameters, getting ready to process)
3. The body of the function: do something useful
4. The results: package them into useful output, perhaps by creating lists of the results
5. return(results) and end the function.

When reading a function, see if you can identify these parts.

A baby function

R code

```
first <- function(input, parameters) {
  #some checks on this
  #now do something (the body
  output  <- input + 1

  #now return what you have done
  return(output)
  }
```

```
first <- function(input, parameters) {
+   #some checks on this
+   #now do something (the body
+   output  <- input + 1
+
+   #now return what you have done
+   return(output)
+   }
>
 first(10)
[1] 11
```

Make the function more complicated (and useful)

R code

```
second <- function(input, par1, par2) {
  #some checks on this
  #now do something (the body)
  for(i in 1: par2) {
  input  <- input + par1
  print(input)
  }
  result <- input
  #now return what you have done
  return(result)
  }
```

```
second <- function(input, par1, par2) {
+   #some checks on this
+   #now do something (the body)
+   for(i in 1: par2) {
+   input  <- input + par1
+   print(input)
+   }
+   result <- input
+   #now return what you have done
+   return(result)
+   }
> second(10,2,4)
[1] 12
[1] 14
[1] 16
[1] 18
[1] 18
```

A few of the most useful data manipulations functions (adapted from Rpad-refcard). Use ? for details

file.choose	() find a file
file.choose	(new=TRUE) create a new file
read.table	(filename)
read.csv	(filename) reads a comma separated file
read.delim	(filename) reads a tab delimited file
c	(...) combine arguments
from:to	e.g., 4:8
seq	(from,to, by)
rep	(x,times,each) repeat x
gl	(n,k,...) generate factor levels
matrix	(x,nrow=,ncol=) create a matrix
data.frame	(...) create a data frame
dim	(x) dimensions of x
str	(x) Structure of an object
list	(...) create a list
colnames	(x) set or find column names
rownames	(x) set or find row names
ncol(x), nrow(x)	number of row, columns
rbind	(...) combine by rows
cbind	(...) combine by columns
is.na	(x) also is.null(x), is...
na.omit	(x) ignore missing data
table	(x)
merge	(x,y)
apply	(x,rc,FUNCTION)
ls	() show workspace
rm	() remove variables from workspace

More useful statistical functions. Use ? for details

mean	(x)
is.na	(x) also is.null(x), is...
na.omit	(x) ignore missing data
sum	(x)
rowSums	(x) see also colSums(x)
min	(x)
max	(x)
range	(x)
table	(x)
summary	(x) depends upon x
sd	(x) standard deviation
cor	(x) correlation
cov	(x) covariance
solve	(x) inverse of x
lm	`(y~x) linear model`
aov	`(y~x) ANOVA`

Selected functions from *psych* package

describe	(x) descriptive stats
describeBy	(x,y) descriptives by group
pairs.panels	(x) SPLOM
error.bars	(x) means + error bars
error.bars.by	(x) Error bars by groups
fa	(x,n) Factor analysis
pca	(x,n) Principal components
iclust	(x) Item cluster analysis
scoreItems	(x) score multiple scales
score.multiple.choice	(x) score multiple choice scales
alpha	(x) Cronbach's alpha
omega	(x) MacDonald's omega
irt.fa	(x) Item response theory through factor analysis
lmCor	`(y~x)` linear model for correlations
bestScales	empirical scale construction

See the R studio analysis for reliability for detailed examples of reliability

1. http://personality-project.org/courses/350/350.wk.4.html
2. A simple script to show the distinction between test-restest and α reliability uses 474 participants on the EPI

R code

```
epi1 <- subset(epiR,epiR$time==1);
epi2 <- subset(epiR, epiR$time==2) #select time 1 and2 Ss
scores1 <- scoreItems(epi.keys,epi1) #find the scores and alpha
scores2 <- scoreItems(epi.keys,epi2) #find the scores and alpha
summary(scores1);  summary(scores2)  #show the summaries for time 2
r12 <- cor2(scores1$scores,scores2$scores) #correlate time 1 and time
round(scores1$alpha,2) ;round(scores2$alpha,2);round(diag(r12),2)
```

```
round(scores1$alpha,2)
         E    N    L  Imp  Soc
alpha 0.77 0.81 0.39 0.52 0.76
> round(scores2$alpha,2)
         E   N   L  Imp  Soc
alpha 0.74 0.8 0.4 0.49 0.75
> round(diag(r12),2)
   E    N    L  Imp  Soc
0.81 0.80 0.65 0.70 0.81
```

For a more complete list of the functions in *psych*

1. *psych* as a Swiss Army knife
   - Data entry and descriptives
   - Multivariate analysis
   - Scale Construction and reliability
   - Regression
   - Machine Learning
   - Simulation
2. This probably makes more sense now after 4 weeks of R

Revelle, W. and Condon, D. (2023). Using unidim rather than omega in estimating undimensionality. *submitted*.

Revelle, W. and Condon, D. M. (2019). Reliability from α to ω: A tutorial. *Psychological Assessment.*, 31(12):1395–1411.