Psychology 350: Advanced statistics and programming in R

William Revelle

Department of Psychology
Northwestern University
Evanston, Illinois USA

Spring, 2023

Outline

Core R may be extended by the use of *packages*

1. R is very powerful, but what makes it even more useful is the large set of *packages* that may be installed.
2. Each package contains at least one (and perhaps hundreds) of `functions`.
3. Each `function` comes with a help menu with (usually) examples of how to use it.
4. We will discuss one such *package*, the *psych* package which contains many functions specifically written for psychological research.
5. It may be installed with the normal menu option. Once installed, you need to make it, and any other packages active by using the `library` command:

R code

```
install.packages(c("psych","psychTools"))  #just do this once
#or, to get the cutting edge version
install.packages(c("psych","psychTools"),
      repos="http://personality-project.org/r",
       type="source")
#in which case, you will need to restart R
library(psych)  #need to do this at the beginning of every session
```

Getting help

1. For some packages, ? the name of the package will give preliminary help
2. For many packages, 'vignettes' will give detailed descriptions of how to use it
3. These may be found in the 'vignettes' menu or, for *psych* at `https://personality-project.org/r/psych/intro.pdf`

A small subset of very useful packages

- General use
  - core R
  - MASS
  - lattice
  - lme4 (core)
  - psych
  - Zelig
- Special use
  - ltm/eRm/mirt
  - sem
  - lavaan/OpenMx
  - GPArotation
  - mvtnorm
  - > 20,000 known
  - + ?
- General applications
  - most descriptive and inferential stats
  - Modern Applied Statistics with S
  - Lattice or Trellis graphics
  - Linear mixed-effects models
  - Personality/psychometrics/general purpose
  - General purpose toolkit
- More specialized packages
  - Latent Trait Model (IRT)
  - SEM and CFA (RAM path notation)
  - SEM and CFA (multiple groups)
  - Jennrich rotations
  - Multivariate distributions
  - Thousands of more packages on CRAN
  - Code on GitHub/ webpages/journal articles

Objects and Functions

1. R is a collection of Functions that act upon and return Objects
2. Although most functions can act on an object and return an object (a =f(b)), some are binary operators
   - primitive arithmetic functions +, -, * , /, %*%, ^
   - logical functions <, > ,==, !=
3. Some functions return “invisible" values
   - e.g., p <- `print(x,digits=3)` will print out x to 3 digits but also returns a value to p.
   - Similarly, s <- `summary(some object)` will return the value of the `summary` function.
4. But most useful functions act on an object and return a resulting object
   - This allows for extraordinary power because you can combine functions by making the output of one the input of the next.
   - The number of R functions is very large, for each package has introduced more functions, but for any one task, not many functions need to be learned. Keep a list of the ones you use.

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
principal	(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

Show all the functions in the psych package

`objects("package:psych")`

```
 objects("package:psych")
  [1] "%+%"                      "ability"                  "affect"                   "all.income"
  [5] "alpha"                    "anova.psych"              "autoR"                    "Bechtoldt"
  ...
[49] "cohen.kappa"              "comorbidity"              "con2cat"                  "congeneric.s
 [53] "cor.ci"                  "cor.plot"                 "cor.plot.upperLowerCi"    "cor.smooth"
 ...
 [81] "cushny"                  "d2r"                      "densityBy"                "describe"
...
[109] "epi.dictionary"          "equamax"                  "error.bars"               "error.bars.
...
[177] "ICC2latex"               "iclust"                   "ICLUST"                   "ICLUST.clus
...
[201] "irt.fa"                  "irt.item.diff.rasch"      "irt.person.rasch"         "irt.respons
...
[241] "mixed.cor"               "mixedCor"                 "mlArrange"                "mlPlot"
...
[253] "omega"                   "omega.diagram"            "omega.graph"              "omega2latex
...
[309] "read.clipboard.upper"    "read.file"                "read.file.csv"            "read.https"
...
[329] "score.alpha"             "score.irt"                "score.irt.2"              "score.irt.p
[333] "score.items"             "score.multiple.choice"    "scoreFast"                "scoreIrt"
...
[405] "Thurstone"               "Thurstone.33"             "topBottom"                "tr"
[409] "Tucker"                  "unidim"                   "varimin"                  "veg"
...
```

Functions

1. All functions
   1.1 Take input (usually as a set of parameters)
   1.2 Process the input
   1.3 Return the results (either explicitly, or as a list)
2. The help option will explain what the inputs are, what it does, and what it returns
3. The examples will show various forms of input, various processes, and various outputs
4. The vignettes will walk through more complicated examples and are meant to be more helpful than just the help pages

Using functions by scripting

Most statistical procedures require three or more steps. You can do this a series of calls to various functions

1. Getting the data (e.g. `read.file`)
2. Describing and cleaning the data (e.g., `describe`, `scrub` `pairs.panels`)
3. Do some statistical operation (e.g., `t-test` `lm`, `aov`, `cor`
4. Organize the results into helpful tables.

These steps should be documented so that you can do them again You can make up your own set of 'useful R' commands which you can then use again and again

Writing functions

1. Eventually, the set of scripts that you have written become cumbersome and you can write little functions to save you time.
2. For instance, if you frequently want to convert a correlation to the z score equivalent using the Fisher transformation, you could write a function similar to `fisherz`.
3. Or, if you want to know the effect size equivalent of a correlation `r2d`
4. Each function needs to define what the inputs will be, what default values they have (if any)
5. Each function then needs to process those inputs in some meaningful way
6. The value returned may be a single item (or vector) or a list of objects.
7. How to write functions? Read someone else's functions and then modify them.

Programming Style and advice

1. Everyone has their own style, but there are some recommended styles
2. Document your scripts, document your functions.
3. You thought long and hard about how to make something work. Once it does, say what you did so that when you need to do this again later, you know what to do.
4. Try to keep all scripts and functions to be less than a screen long. This makes it easier to follow your own code.
5. Break long scripts into shorter chunks which you can test as you go along.