Yet Another R FAQ, or How I Learned to Stop Worrying and Love Computing [1]

Roger Koenker
CEMMAP and University of Illinois, Urbana-Champaign

"It was a splendid mind. For if thought is like the keyboard of a piano, divided into so many notes, or like the alphabet is ranged in twenty-six letters all in order, then his splendid mind had no sort of difficulty in running over those letters one by one, firmly and accurately, until it had reached the letter Q. He reached Q. Very few people in the whole of England reach the letter Q.... But after Q? What comes next?... Still, if he could reach R it would be something. Here at least was Q. He dug his heels in at Q. Q he was sure of. Q he could demonstrate. If Q then is Q–R–.... Then R... He braced himself. He clenched himself.... In that flash of darkness he heard people saying–he was a failure–that R was beyond him. He would never reach R. On to R, once more. R—.... ...He had not genius; he had no claim to that: but he had, or he might have had, the power to repeat every letter of the alphabet from A to Z accurately in order. Meanwhile, he stuck at Q. On then, on to R."

Virginia Woolf (To the Lighthouse)

1. How to get it? Google CRAN, click on your OS, and download. Buy a case of wine with what you've saved.
2. How to start? Click on the R icon if you are mousey, type R in a terminal window if you are penguinesque.
3. What next? At the prompt, > type `2 + 2`
4. What next? At the prompt, > type `1:9/10`
5. What next? At the prompt, > type `x <- 1:99/100`

[1] Version: April 19, 2016. Revised for an Short Course on Quantile Regression: 18-20 May 2016, in Copenhagen. More official R FAQs are available from the CRAN website. A FAQ for the quantile regression package **quantreg** can be found by the invoking the command `FAQ()` from within R after loading the package.

6. What next? At the prompt, > type `plot(x,sin(1/x))`

7. What next? At the prompt, > type `lines(x,sin(1/x),col = "red")`

8. How to stop? Click on the Stop sign if you are mousey, type `q()` if you are penguinesque.

9. Isn't there more to R? Yes, try downloading some packages: using the menu in the GUI if you are mousey, or typing install.packages("pname") if you are penguinesque.

10. What's a package? A package is a collection of R software that augments in some way the basic functionality of R, that is it is a way of going "beyond R." For example, the **quantreg** package is a collection of functions to do quantile regression. There were 6789 packages on CRAN as of June 2015.

11. How to use a package? Downloading and installing a package isn't enough, you need to tell R that you would like to use it, for this you can either type: `require(pname)` or `library(pname)`. I prefer the former.

12. How to read data files? For simple files with values separated by white space you can use `read.table`, or `read.csv` for data separated by commas, or some other mark. For more exotic files, there is `scan`. And for data files from other statistical environments, there is the package **foreign** which facilitates the reading of Stata, SAS and other data. There are also very useful packages to read html and other files from the web, but this takes us beyond our introductory objective.

13. What is a `data.frame`? A `data.frame` is a collection of related variables; in the simplest case it is simply a data matrix with each row indexing an observation. However, unlike conventional matrices, the columns of a `data.frame` can be non-numeric, e.g. logical or character or in R parlance, "factors." In many R functions one can specify a `data = "dframe"` argument that specifies where to find the variables mentioned elsewhere in the call.

14. How to get help? If you know what command you want to use, but need further details about how to use it, you can get help by typing `?fname`, if you don't know the function name, then you might try `apropos("concept")`. If this fails then a good strategy is to search `http://finzi.psych.upenn.edu/search.html` with some relevant keywords; here you can specify that you would like to search through the R-help newsgroup, which is a rich source of advice about all things R.

15. Are there manuals? Yes, of course there are manuals, but only to be read as a last resort, but when things get desparate you can always RTFM. The left side of the CRAN website has links to manuals , FAQs and contributed documentation. Some of the latter category is quite good, and is also available in a variety of natural languages. There is also an extensive shelf of published material about R, but indulging in this tends to put a crimp in one's wine budget.

16. What about illustrative examples? A strength of R is the fact that most of the documentation files for R functions have example code that can be easily executed. Thus, for example if you would like to see an example of how to use the command `rq` in the **quantreg** package, you can type `example(rq)` and you will see some examples of its use. Alternatively, you can cut and paste bits of the documentation into the R window; in the OSX GUI you can simply highlight code in a help document, or other window and then press Command-Enter to execute. Similarly, many packages have demo files that act as auxiliary documentation. To see what demos are available for currently loaded packages, just try `demos()`. Finally, many packages have vignettes, short overviews of various aspects of the functionality of the package usually with explicit examples of how to do things. For example, the **quantreg** package has three vignettes: one basic, one about survival modeling, and one about additive nonparametric models. Vignettes can be accessed from R by simply typing `vignette(''vname'')`. The names of the various package vignettes can be found by typing `vignette()`.

17. What's in a name? Objects in R can be of various types called classes. You can create objects by assignment, typically as above with a command like `f <- function(x,y,z)`. A list of the objects currently in your private environment can be viewed with `ls()`, objects in lower level environments like those of the packages that you have loaded can be viewed with `ls(k)` where `k` designates the number of the environment. A list of these environments can be seen with `search()`. Objects can be viewed by simply typing their name, but sometimes objects can be very complicated so a useful abbreviated summary can be obtained with `str(object)`.

18. What about my beloved least squares? Fitting linear models in R is like taking a breath of fresh air after inhaling the smog of other industrial environments. To do so, you specify a model `formula` like this: `lm(y ~ x1 + x2 + x3, data = "dframe")`, if one or more of the x's are factor variables, that is take discrete, qualitative values, then they are automatically exanded into several indicator variables. Interactions plus main effects can be specified by replacing the "+" in the formula by "*". Generalized linear models can be

specified in much the same way, as can quantile regression models using the **quantreg** package.

19. What about class conflict? Class analysis can get complicated, but you can generally expect that classes behave themselves in accordance with their material conditions. Thus, for example, suppose you have fitted a linear regression model by least squares using the command `f <- lm(y ~ x1 + x2 + x3)`, thereby assigning the fitted object to the symbol `f`. The object `f` will have class `lm`, and when you invoke the command `summary(f)`, R will try to find a summary method appropriate to objects of class `lm`. In the simplest case this will entail finding the command `summary.lm` which will produce a conventional table of coefficients, standard errors, t-statistics, p-values and other descriptive statistics. Invoking `summary` on a different type of object, say a data.frame, will produce a different type of summary object. Methods for prediction, testing, plotting and other functionalities are also provided on a class specific basis.

20. What about graphics? R has a very extensive graphics capability. Interactive graphics of the type illustrated already above is quite simple and easy to use. For publication quality graphics, there are device drivers for various graphical formats, generally I find that `pdf` is satisfactory. Dynamic and 3D graphics can be accessed from the package **rgl**.

21. Latex tables? The package **Hmisc** has very convenient functions to convert R matrices into latex tables.

22. Random numbers? There is an extensive capability for generating pseudo random numbers from R. Reproducibility of random sequences is ensured by using the `set.seed` command. Various distributions are accessible with families of functions using the prefixes pdqr, thus for example `pnorm`, `dnorm`, `qnorm` and `rnorm` can be used to evaluate the distribution function, density function, quantile function, or to generate random normals, respectively. See `?Distributions` for a complete list of standard distributions available in base R in this form. Special packages provide additional scope, although it is sometimes tricky to find them.

23. Programming and simulation? The usual language constructs for looping, switching and data management are available, as are recent developments for exploiting multicore parallel processing. Particularly convenient are the family of `apply` functions that facilitate summarizing matrix and list objects. A good way to learn the R language is to look at the code for existing functions. Most of this code is easily accessible from the R command line. If

you simply type the name of an R function, you will usually be able to see its code on the screen. Sometimes of course, this code will involve calls to lower level languages, and this code would have to be examined in the source files of the system. But everything is eventually accessible. If you don't like the way a function works you can define a modified version of it for your private use. If you are inspired to write lower level code this is also easily incorporated into the language as explained in the manual called "Writing R Extensions."