R Quick Start Guide

Amanda White, Tony Bladek, Landon Sego

14 July 2014

R Language Tips:

- Expressions and commands are CaSe-SenSiTivE
- Anything following the pound character (#) R ignores as a comment.
- Assignments to variables are done with either an '=' sign or the '<-' character sequence.
- Assignment to parameters for function calls are done using the equal (=) sign.
- An object name must start with an alphabetical character, but may contain numeric characters thereafter.
- A period may also form part of the name of an object. For example, x.1 is a valid name for an object in R.
  Spaces, 'x 1' for example, are not valid in an object name.
- **List:** To list the objects you have created in a session enter the commands `objects()` or `ls()` (both do the same thing).
- **Delete:** To remove all the objects you have created, enter the command `rm(list = ls(all = TRUE))`.
- **Save:** To save all the objects you have created in your current session, use the "File" menu (`Save As()`) or use the `save.image()` command.

R Basic Commands

- **Help**
  - `Help.search('search term')`: get help when you are not sure of the name of the command or package. (e.g. `help.search('graphics')`)
  - ??'search term': get help when you are not sure what the name of the command or package. R will return results related to your search term.
  - `help(command)`: get help on using a certain command when you know the name of the command. (e.g. `help(print)`)
  - `?'command'`: get the help content on a command (e.g. `?print`);
  - `args(command)`: gets help with the command arguments when you know the name of the command.
- **Demo**
  - `demo('graphics')`: demo the graphics capabilities of R
  - `demo(lm.glm)`: demo the linear modeling capability of R.
  - `example(command)`: Will execute all the example code from the 'command'
- **Data**
  - `X<-5` or `X=5`: Assign a single value to an object.

- `c()`: To create a vector of numbers (e.g. `my.vector <-c(1,2,3,4,5)`) Can also be used to create a vector of character strings (e.g. `c("cat", "dog", "fish")`) or True/False values (e.g. `c(T,F,T,T)`)
- `:` or `seq()`: To create a sequence of number use the colon : or the seq() command (e.g. `my.sequence<-0:10` or `my.sequence2<-seq(0,10, by = 2)`
- **Named Vector:** Create a named vector (e.g. `aNamedVector <- c(type =1, count =7 ,max =10)` or `bNamed Vector <- c("a type" = 1, "b%type" = 7)`
  - The values can be integer, numeric, character, logical (T/F) or complex values
- `[]`: To reference specific elements use square brackets (`blob<=foo[4]`)
  - Element numbering starts with 1 not 0. (`my.vector[3]=3`)
  - Use the name of the element (e.g. `aNamedVector[["type"]]` or `aNamedVector[c("type", "max")]`
- `indicator`: create a vector of TRUE, FALSE values based upon a conditional (e.g. `indicator <- y <4;` Which values in the vector y are less than 4?)
- `rep(value, number of repeats)`: repeat function (e.g. `rep(1:5, 2)` repeat the vector 1 to 5 twice)
- `%in%` : Find a specific value (substring) in another string (e.g. y %in% c("cat", "dog"))
- `!` (Negation) : Negate a T/F vector (e.g. `!z`)
- `& (AND)`: 'And' the values of at T/F vector (e.g. `x & y`)
- | (OR): 'OR' the values of a T/F vector (e.g. x | y)
- **`Quotes:`** R will accept both double and single quotes for all character strings. It is a way of embedding quoted values within another string. (e.g. `x<- "A String with 'inner quotes'"`)
- `print(y)`: print the contents of the variable y to the screen
- `cat()`: prints text to the screen (e.g. cat ("Here is the value of ", x,"\n")
- `load('filename')`: Load a data file from local directory.
- `attach(dataset)`: load named data set (`attach(mtcars)`)
- `str(y)`: show the structure of a variable y
- `length(y)`: show the number of elements in y
- `head(y)`: show the first few members of the variable y
- `tail(y)`: show the last few members of the variable y
- `which(y)`: returns the indices of a T/F vector which are TRUE
- `any(y)`: returns TRUE if any of the values of the T/F vector are TRUE
- `all(y)`: returns TRUE if all of the values of a T/F vector are TRUE
- **Dataframe:** A basic unit of data in R is the data frame. It is typically a matrix of data with named columns much like an Excel spreadsheet
  - Example: The mtCars data set. It has the following columns:
    - `mpg,cyl,disp: num,hp,drat,wt,qsec,vs.am,gear,carb`
- **Column Reference:** To reference a named member of a data frame use the dollar sign ($)

        - e.g. `mtCars$mpg, mtCar$qsec`.
    - `nrow(dataset)`: Display the number of rows in a named data frame (e.g. `nrow(mtcars)`)
    - `ncol(dataset)`: Display the number of rows in a named data frame (e.g. `ncol(mtcars)`)
    - `nchar(x)` : Count the number of characters in the character vector x
    - **Matrix:** A matrix is similar to a dataframe except that all the elements in a matrix must be the *same* type but the type does not matter (integer, numeric, character, logical, or complex)
        - Matrices are created using the matrix() command. (e.g. `m1<- matrix(1:15, nrow = 5, ncol =3)`. They are loaded column-wise by default
        - All the standard matrix operations are available in R including transpose (`t(m1)`), Inner product, outer product and multiplication (all use '`%*%`'; R returns the correct value depending upon the data type.), inversion (`solve(m1)`), eigenvectors (`eigen(m1)`) and determinant (`deter(m1)`)
    - It is possible to convert from a formal matrix to a dataframe (and back) as long as there is only one datatype.
    - `rownames(m1)`: Extract the row names from a matrix
    - `colnames(m1)`: Extract the column names from a matrix
    - **Lists:** A list is a collection of R objects which do not have to be of any particular type or size and can even be other lists.
        - Lists are created using the list() command. (e.g. `aList <- list(a = 1:5, b = rep(TRUE, 2), c = letters[1:3])`
    - `names(l1)`: Extract the names from a list
- **Basic Calculator Commands**
    - Addition: '+' sign (e.g. `7+3`)
    - Subtraction: '−' sign (e.g. `7−3`)
    - Multiplication: '*' sign (e.g. `7*3`)
    - Division: '/' sign (e.g. `7/3`)
    - Integer Division: '`%/%`' (e.g. `7%/% 3`)
    - Division Remainder: '`%%`' (e.g. `7%%3`)
    - Exponentiation: `exp(#)` (e.g. `exp(7)` = the constant e = 2.718282.. raised to the power of 7)
    - Natural Logarithm: l`og(#)` (e.g. `log(7))`
    - Base 10 Log: `log10(#) (e.g. log10(1000))`
    - Square Root: `sqrt(#) (e.g. sqrt(16))`
    - Cosine: `cos(radian)` (e.g. cos(pi) where pi is a defined constant = 3.141519…radians)
    - Large Numbers: `#e# (e.g.1.7e+05)`
    - Small Numbers: `#e# (e.g. 1.7e-03)`
    - Division by 0: `#/0 (e.g. 1/0 = NaN)`

- **Basic Statistical Commands**
  - `^`: raise to the power (e.g. `y<-x^2` (y = x squared))
  - `mean(y)`: calculate and show the mean of the vector y
  - `var(y)`: calculate and show the variance of the vector y
  - `sd(y)`: calculate and show the standard deviation for a vector y
  - `sum(y)`: calculate and show the sum of the vector(y) for a numerical vector or the number of true values in a T/F vector y
  - `min(y)`: calculate and show the minimum of a vector y
  - `max(y)`: calculate and show the maximum of vector y
  - `quantile(y,.25)`:calculate the 25th percentile of a vector y
  - `quantile(y,0.75)`:calculate and show the 75th percentile of the vector y
  - `summary(y)`: calculate and show basic statistical measures of the variable y
  - `lm()`: fit a linear regression model to the data (e.g. `lm_1<- lm(y ~ x)` y = f(x))
- **Input/Output**
  - `read.csv()`: read a comma-separated-value (csv) file (e.g. `read.csv("nf-week2-sample.csv");`
  - `save()`: Save an R binary object to disk for use later in the session or a new session. Used primarily for objects which take some time to compute. (e.g `x <- runif(20); y <- list(a = 1, b = TRUE, c = "oops");save(x, y, file = "xy.Rdata"`)
  - `unlink()`: Remove a previously saved R binary data object from disk (e.g. `unlink("xy.Rdata")`)
- **Library Commands**
  - `install.packages("package")`: install for use in the working directory a new R package (e.g. `install.packages("ggplot2");`). To install a new R package for use in specified location use the argument *lib* to specify such as `install.packages("package",lib="location")` (e.g. `install.packages("ggplot2",lib="/data/Rpackages/")`);
  - `library(library)` :loads library for use (e.g. `library(lattice);`
  - `view(package)`: Show contents of package (e.g. `view(mtcars)`)
  - `attach(package)`: Load package (e.g. `load(mtcars);`
- **Directory Commands**
  - `ls()` : Show all variables in the current environment
  - `dir.create('location')`: Create a local directory (e.g. `dir.create("data/artifacts")`)
  - `dir('directory')`: look at the contents of a directory
  - `dir()`: look at the contents of the current working directory
  - `setwd('directory')`:set working directory
  - `getwd()`: get working directory
  - `source('script_file.R', echo = TRUE)`: run an entire R script file and show the commands and the results in the console window

- **Plotting**
  - `hist()`:make a histogram plot (e.g. `hist(mtCars$mpg)`)
  - `plot()`:plot an x vs y plot (e.g. (`plot(mtCars$mpg, mtCars$cyl)`)
  - `ggplot2()`: plotting command from the ggplot package (see `??gplot` for details).

- **Functions**
  - Functions in R have inputs (arguments) and outputs(values)
  - Arguments can be named or un-named
  - Examples:
    - Simple: `simple<- function(x,y) { return x+y-7) }`
    - Named argument: `simp1 <- function(x, method = "sum") {if (method == "sum") {out <- sum(x)}else if (method == "prod") {out <- prod(x)}else {stop("'method' must be 'sum' or 'prod'")}return(out)}`

Web Sites:

- **Introduction to R, CRAN:** http://cran.r-project.org/doc/manuals/R-intro.html
- **R for Beginners:** http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
- **R Tutorial:** http://www.r-tutor.com/
- **RStudio:** http://www.rstudio.com
- **CRAN:** http://cran.us.r-project.org
- **DataDR:** http://tesseradata.org/datadr/
- **Trelliscope:** http://tesseradata.org/trelliscope/
- **Bootcamp code and description:** http://tesseradata.org/docs-r-intro-bootcamp/
- **VAST Challenge Data exploration** http://tesseradata.org/example-vast-challenge/