

# One-Page Intro to R

Sara E. Burke — Updated 2015-09-07

**To download R**, visit [www.r-project.org](http://www.r-project.org), click "download R," and then choose any mirror site. Follow the directions for your operating system.

## Executing R scripts

- Type commands into the command line to see immediate output.
- Type out lists of commands in a script file. From the R program, execute selected commands from a script using `ctrl+R` or `command+enter`.

## If you need help

- For help with a specific function or command, type `?`  followed by the name, e.g., `?aov`
- Search Google. There are many free online tutorials for specific tasks in R.
- Try looking through this introductory manual: <http://cran.r-project.org/doc/manuals/R-intro.html>

## Important symbols

<code>* + - / ^ ( )</code>	Basic arithmetic symbols work intuitively.
<code>x = 9</code> or <code>x &lt;- 9</code>	variable assignment (these examples set <code>x</code> to the numeric value <code>9</code> )
<code>x="aa"</code> or <code>x='aa'</code>	Quotation marks enclose character data rather than numeric data.
<code>sum(2,3,5)</code>	Parentheses contain function input; commas separate multiple input values.
<code>x[2]</code>	Square brackets allow you to specify items within a vector.
<code>1:10</code>	A colon generates a list of integers (in this example, from <code>1</code> to <code>10</code> ).
<code>T TRUE F FALSE</code>	<code>TRUE</code> and <code>FALSE</code> are the Boolean values. <code>T</code> and <code>F</code> stand for them.
<code>&lt; &gt; &lt;= &gt;= != ==</code>	These symbols all compare two values, returning <code>True</code> or <code>False</code> .
<code>x == 9</code>	returns <code>True</code> if <code>x</code> is <code>9</code> . Don't mix this up with the assignment operator!
<code>T F T&amp;F !F</code>	<code> </code> means "or", <code>&amp;</code> means "and", and <code>!</code> means "not"
<code>NA</code>	missing value
<code># comment text</code>	Text that follows <code>#</code> on a line will be ignored by R.

## Functions of vectors

<code>x = c(1,2,3)</code>	concatenates a list of values to form a vector, then calls that vector <code>x</code>
<code>length(x)</code>	returns the number of items in vector <code>x</code>
<code>sum(x) mean(x) var(x) sd(x) min(x) max(x) median(x)</code>	basic statistics of vectors
<code>x * 2</code>	returns a vector containing each item in vector <code>x</code> multiplied by <code>2</code>
<code>log(x)</code>	returns the natural log of each item in <code>x</code>
<code>x == 9</code>	returns a vector of Boolean values indicating whether each item in <code>x</code> is equal to <code>9</code>
<code>sum(x==9)</code>	The sum of a Boolean vector treats <code>True</code> as <code>1</code> and <code>False</code> as <code>0</code> , so this syntax gives the number of values in <code>x</code> that are equal to <code>9</code> .
<code>is.na(x)</code>	returns a vector of Boolean values indicating whether each item in <code>x</code> is missing
<code>x[y==9]</code>	If <code>x</code> and <code>y</code> are vectors of equal length, this syntax returns all values of <code>x</code> for which the corresponding value of <code>y</code> is equal to <code>9</code> .
<code>which(y==9)</code>	returns the list of indices for which <code>y</code> is equal to <code>9</code>

## Organizational functions

<code>ls()</code>	returns a list of all variables assigned so far
<code>rm(x)</code>	removes or deletes the variable <code>x</code>
<code>library()</code>	attaches a package. Useful packages include <code>car</code> and <code>psych</code>
<code>search()</code>	identifies data frames and packages which are attached

## Working with data frames

`read.table()` creates a data frame from a text file. Remember to specify `header=T`  
`read.csv()` creates a data frame from a comma separated text file  
`names(data)` lists all the column/variable names in the data frame called `data`  
`attach(data)` attaches the data frame called `data` so variables can be easily referenced

## Formula syntax

`y~x1` predict the variable `y` using the variable `x1`  
`y~x1+x2` predict the variable `y` using the main effects of `x1` and `x2`  
`y~x1*x2` predict the variable `y` using the main effects of `x1` and `x2` and their interaction  
`factor()` transforms a variable into a factor so `lm()` or `aov()` will treat it as such

## Common statistical tests

Use `?` at the command line to learn how to specify the inputs for these functions.

`lm()` linear model – regression or ANOVA  
`summary()` Input your `lm()` object and this function will give you a regression summary.  
`summary.aov()` Input your `lm()` object and this function will give you an ANOVA summary.  
`aov()` linear model – ANOVA only  
`leveneTest()` Input your `lm()` or `aov()` object and this function will perform Levene's test.  
This function requires the `car` package. Don't forget the capital T in Test.  
`TukeyHSD()` Input your `aov()` object and this function will perform a Tukey test.  
`t.test()` t test – input can be a single variable, two variables, or a formula  
`var.test()` F test to compare two variances  
`chisq.test()` Pearson chi-squared test – input can be one or two variables  
`cor.test()` correlation significance test

## Scale analysis

`cbind()` Sticks a series of vectors next to each other as columns of a matrix.  
`rowMeans()` Computes row means of a matrix – use this for averaging variables into a scale.  
Specify `na.rm=T` to compute means ignoring missing values.  
`cronbach()` Cronbach's alpha of matrix columns – requires the `multilevel` package.  
`principal()` principal component analysis of matrix columns – requires the `psych` package.  
Use the `GPARotation` package for extra rotations.

## Plotting

Checking `?par` and `?plot` is very helpful when plotting.

`par()` set graphics parameters before plotting – see `?par` for details  
`plot()` scatter plot by default – input can be two variables or a formula  
`barplot()` basic bar plot – see me for a more advanced custom version  
`hist()` histogram  
`qqnorm()` normal Q-Q plot  
`abline()` draw a line on the current plot – if the input is a `lm` object, it draws the best fit line  
`legend()` add a legend to the existing plot – see `?legend` for details