**Recursive Function**

A [function](https://www.datamentor.io/r-programming/function) that calls itself is called a recursive function and this technique is known as recursion.

This special programming technique can be used to solve problems by breaking them into smaller and simpler sub-problems.

# Recursive function to find factorial

recursive.factorial <- function(x) {

if (x == 0) return (1)

else return (x \* recursive.factorial(x-1))

}

> recursive.factorial(5)

# String functions

* [**toupper/tolower**](http://stat.ethz.ch/R-manual/R-patched/library/base/html/chartr.html)(*x*)
* [**strsplit**](http://stat.ethz.ch/R-manual/R-patched/library/base/html/strsplit.html)(*x*, *split*)
* [**paste**](http://stat.ethz.ch/R-manual/R-patched/library/base/html/paste.html)(…, **sep**=*""*, **collapse**=*NULL*)
* [**gsub**](http://stat.ethz.ch/R-manual/R-patched/library/base/html/grep.html)(*pattern*, *replacement*, *x*, **ignore.case**=*FALSE*, **fixed**=*FALSE*)
* [**substr**](http://stat.ethz.ch/R-manual/R-patched/library/base/html/substr.html)(*x*, **start**=*n1*, **stop**=*n2*)
* paste(..., sep = " ", collapse = NULL)
* print(paste(a,b,c))
* print(paste(a,b,c, sep = "-"))
* print(paste(a,b,c, sep = "", collapse = ""))
* result <- nchar("Count the number of characters")
* print(result)
* substring(x,first,last)

# Vector

x <- c(1, 5, 4, 9, 0)

seq(1, 3, by=0.2) # specify step size

Accessing MySQL through R

Connecting to MySQL is made very easy with the RMySQL package. To connect to a MySQL database simply install the package and load the library.  
  
install.packages("RMySQL")  
library(RMySQL)

**Connecting to MySQL:**  
Once the RMySQL library is installed create a database connection object.  
  
mydb = dbConnect(MySQL(), user='user', password='password', dbname='database\_name', host='host')

**Listing Tables and Fields:**  
Now that a connection has been made we list the tables and fields in the database we connected to.  
  
dbListTables(mydb)  
  
This will return a list of the tables in our connection.   
  
dbListFields(mydb, 'some\_table')  
  
This will return a list of the fields in some\_table.

**Running Queries:**  
Queries can be run using the dbSendQuery function.  
  
dbSendQuery(mydb, 'drop table if exists some\_table, some\_other\_table')  
  
In my experience with this package any SQL query that will run on MySQL will run using this method.

**Making tables:**  
We can create tables in the database using R dataframes.  
  
dbWriteTable(mydb, name='table\_name', value=data.frame.name)

**Retrieving data from MySQL:**  
To retrieve data from the database we need to save a results set object.  
  
rs = dbSendQuery(mydb, "select \* from some\_table")  
  
I believe that the results of this query remain on the MySQL server, to access the results in R we need to use the fetch function.  
  
data = fetch(rs, n=-1)  
  
This saves the results of the query as a data frame object. The n in the function specifies the number of records to retrieve, using n=-1 retrieves all pending records.

Web Data

## Using data.table’s fread()

I love the data.table [package](https://github.com/Rdatatable/data.table/wiki). I use it every day, for almost every project I do. It’s an extension of the data.frame object class in R that makes many improvements. One of those improvements is in the function fread(). It’s data.table’s answer to base R’s read.csv(). It does many things better, but here I’m only going to address its ability to read data right from the web. As a primer, its typical use on a data file residing on your computer would look something like this:

**library**(data.table)  
mydat <- fread('C://Some/File/Path.csv')

Reading data from a source on the web is no different. The example the package authors give in the help file (?fread) is this:

**library**(data.table)  
mydat <- fread('<http://www.stats.ox.ac.uk/pub/datasets/csb/ch11b.dat>')  
head(mydat)

**library**(RCurl)  
myfile <- getURL('<https://sakai.unc.edu/access/content/group/3d1eb92e-7848-4f55-90c3-7c72a54e7e43/public/data/bycatch.csv>', ssl.verifyhost=FALSE, ssl.verifypeer=FALSE)