Center for Tropical Forest Science R Package Manual
Pamela Hall, Suzanne Lao, Ellen Connell and Marie Massa
Version 1.00 March 29, 2006

1. Installing and Customizing the R environment

This manual is written for use by CTFS colleagues using R in a Windows (version 95 or later) operating system. Some information is also provided for use of R with Mac OSX. The vast majority of the users experience of the R environment will be identical across platforms.

There are a number of books an online manuals for learning to use R. We suggest that you access one or more of these and along with the CTFS manuals you should, with some attention to detail ! be able to run R programs and analyse the CTFS data. It takes time to learn to work in a new programming environment, so give yourself a break, follow the examples in the online manuals and just keep experimenting. You can't damage anything beyond repair as long as you have copies of your dataset safely squirreled away. You can always install a new version of R if things really appear to be boloxed up.

1.1 Installing R

All CTFS partners working on analyzing CTFS datasets should have the most recent version of R installed on their computer. The R installation file can be downloaded from the CRAN homepage http://cran.r-project.org/. Users should return to this web page to keep track of changes and new releases of R.

From this website R users can easily update their R program. The R program can either be installed from a binary (precompiled form) or from source files (compiled locally on a users machine). If you have previously installed R as a binary, continue to update it in the same form. If you have installed R by compiling source files locally, then do this for updating. Do not mix locally compiled R with binary precompiled R updates or vice versa.

The easiest and fastest method of installing R is to locate the link for the operating system that you are using under the *"Precompiled Binary Distributions"* section of CRAN's homepage. The user should click on *Windows (95 or later)* and will then be prompted to choose between two subdirectories. *Base* provides access to latest R files and standard packages that go with the base. Installing R for MAC OSX is much the same. If using a precompiled binary form, chose the *"Precompiled Binary Distributions"* section on CRAN's homepage. Chose the most recent version of R for the Mac and check to see if you need to download a separate update of the R GUI (graphical user interface). Shortcut or aliases for R and R-GUI (Mac) are created during installation. Use these to make life easier!

When R is installed on your machine, the R directory will contain folders with the files needed to run R. The folder named library contains sub-folders for each package that automatically came with this version of R. Any package that is installed in the future, including the **CTFS package**, will be installed into its own subfolder in the library. More instructions on how to download and install R are available on the CRAN homepage.

Installing other packages can be done using the *Package Installer* in the Mac. Install the package and then load it using *Package Manager*. For Windows, packages can be downloaded from the CRAN site by clicking on *contrib.* This is where the CTFS package will appear as well as its updates.

1.2 Running R and Quitting R

Run R. A window called the *R Console* will open. R is interactive mode such that lines of code are evaluated immediately upon input. Commands are entered at the R prompt below the bar and the results of the command appear above. In R, a **session** begins each time R Console opens and ends each time R is exited. The **R workspace** is all of the R objects (discussed in Chapter 2,) that were created in the current, and possibly R sessions. This is also known as the **GlobalEnv**. All R commands are in the form of *command()*. Various options are placed inside the () which are unique to each function.

To quit R, select Exit from the File menu or type '*q()'* into the command line. R will then ask the user whether he/she wants to save the workspace image. If you answer yes, all of the R objects are saved into the file ".RData" in the default directory. These will be available the next time R is started and you load in this save workspace. If you answer no, all objects that have not been explicitly saved to a folder on disk will be deleted. (See below for more details).

In most cases, the user should answer NO since starting a new session with an old workspace means that much of the memory available for R is already used. Note that R generally requires 512 RAM to run analyses of full CTFS datasets.

1.3 Important R Terms:

Ambiguity and uncertainty about the definitions of many R related terms is a source of confusion and frustration for many CTFS colleagues. This confusion is also evident in the R documentation produced by the R project; many concepts are given several names and the same names are given to several different concepts. This section will try to define some of these terms in a straightforward and clear manner. Let us know if these definitions work for you and if they are used consistently in this manual and other documentation.

The **R Environment** is meant to encompass everything that is currently accessible to the user when R is opened. This includes the R Console, the R Window, the objects, packages and help pages available to the user. It also includes files that are read into the current R session using the appropriate R functions.

The **R Environment** is divided into **search paths** which contain all of the objects (functions, variables, datasets, etc.) that the **R Environment** has available at any given time.

Search Path 1, which is also referred to as the **Global Environment** and the **Workspace Image**, is the location of all objects that are created at the R command line. Objects that are

"sourced" and "loaded" into R are also put into this search path. (See Section 1.5 Useful Functions for explanation of *source(), load()* and *attach()*.)

Search Path 2, 3, 4, **etc.** are the location of all packages that are "installed" or "loaded" into the **R Environment** as well as R objects that are attached to the to it. (See Section X.x Useful Functions for explanation of *attach()*.)

1.4 R Commands:

R is an expression language with a very simple syntax. It is case sensitive, so "A" and "a" are considered different statements in R. For example,

```
> A = 2
> a = 1
> A
[1] 2
> a
[1] 1
```

The *[1]* is the R way of indicating the position of the returned value. It is the index of the value. In this case the value is an atomic value.

```
> B=c(1,2,3,4,5)
> B
[1] 1 2 3 4 5
```

B was created by using the *c()* or concatenate function. *B* is a vector, a series of numbers. It has a single dimension. The numbers inside the [] are the index or location of the value. R only prints the index of the first value until it runs out of display area. To whit when *B* is long:

```
> B
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[17] 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

Below *B* is created as a matrix, a 2 dimensions object. The index is provided by the number inside the [] which indicate the row and column position of the values in the matrix.

```
> B=matrix(c(1,2,3,4,5,6),nrow=2,ncol=3)
> B
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

All R commands are expressions or assignments. An **expression** is a command that is evaluated and printed but does not change the value of an object.

```
> 2 + 2
> 4
```

An **assignment** is a command that is also evaluated and the result is saved into an object rather than displayed. The result can be displayed by properly "addressing" an object, which means typing its name and dimension you want displayed.

```
> a <- 2 + 2
> a
> 4
```

If a command is not complete at the end of a line, R will give a different prompt, by default + on second and subsequent lines and continue to read input until the command is syntactically complete.

```
>a <- 2 +
> +
```

In R, the user can use two assignment operators (almost) interchangeably. Above, the assignment operator used was <- . The = is another assignment operator, could have been used instead. The standard is to use <- as the assignment operator and to use = for actual arithmetic.

1.5 Useful Functions:

The following functions are used to control the contents of the search paths, allow objects in the workspace to be saved to disk and provide summary information on the structure of an object.

search(): displays the names all attached packages and R objects in all of the search paths.

```
> search()
 [1] ".GlobalEnv"              "file:bci9095.full.rdata"
 [3] "file:bci95.mult.rdata"   "file:tst.bci9095.spp.rdata"
 [5] "package:CTFSAUG"         "package:methods"
 [7] "package:stats"           "package:graphics"
 [9] "package:utils"           "Autoloads"
[11] "package:base"
```

The **GlobalEnv** is the first search path. There are 3 datafiles in the next search paths (2 to 4). A test package of CTFS functions has been loaded into the 5th path. Then the standard packages that are loaded when R is run as in the remaining paths (6 to 11).

ls(): displays the names of the objects in the search path specified by the argument, a number, passed to the function. When used without an argument (default) the objects in the **GlobalEnv** are listed. When used with a number, the objects in that search path are listed.

```
> ls()
[1] "mort.out"
> ls(2)
[1] "bci9095.full"

> ls(5)
 [1] "abundanceperquad"      "assemble.demography"   "fill.dimension"
 [4] "find.climits"          "growth"                "growth.dbh"
 [7] "growth.eachspp"        "growth.indiv"          "gxgy.to.hectindex"
[10] "gxgy.to.index"         "gxgy.to.rowcol"        "index.to.gxgy"
[13] "index.to.rowcol"       "mergecensus"           "mortality"
[16] "mortality.calculation" "mortality.dbh"         "mortality.eachspp"
[19] "pop.change"            "recruitment"           "recruitment.eachspp"
[22] "rndown5"               "rowcol.to.index"       "select.dbhrange"
[25] "sep.data"              "sep.dbh"               "sep.grform"
[28] "sep.habitat"           "texttordata"           "totalabund"
[31] "totalabund.spp"        "trim.growth"           "unwind.matrix"
```

load(): searches (in the working directory unless otherwise specified by the user) for a file matching the name (in quotation marks) and then copies the file into search path 1 (**GlobalEnv**). If the desired file is in the working directory then no path name is needed. Otherwise, the full path and filename must be provided.

```
> load("bci.spp.info.rdata")
> ls()
[1] "bci.spp.info" "mort.out"
```

After a file (*.rdata) is loaded, it can be addressed by its name and the "rdata" extension is not a part of its name. For more information about how to organizing saved files see chapter 4.

attach():searches (in the working directory unless otherwise specified by the user) for a file matching the name (in quotation marks) and then copies the file into search path 2. All objects in search path 2 and higher are shifted one search path higher. After a file or database (*.rdata) is attached, it can be referenced without its ".rdata" extension.

```
> attach("bci.spp.info.rdata")
> search()
 [1] ".GlobalEnv"            "file:bci.spp.info.rdata"   "file:bci9095.full.rdata"
 [4] "file:bci95.mult.rdata" "file:tst.bci9095.spp.rdata" "package:CTFSAUG"
 [7] "package:methods"       " "package:stats"           "package:graphics"
[10] "package:utils"         "Autoloads"                 "package:base"
```

Note that *bci.spp.info.rdata* is now is search path 2 and all of the preexisiting objects have been "pushed down" one more search path. Compare with the use of *search()* above. After a file

(*.rdata) is loaded, it can be addressed by its name and the "rdata" extension is not a part of its name. For more information about how to organizing saved files see chapter 4.

source(): This function reads R code from a file. In this way, r can automatically get large chunks of code or functions at once. The code is saved into search path 1.

```
>source("basal.area.r")
> ls()
[1] "ba"              "ba.ind"          "ba.perquad"      "ba.tot"
[5] "ba.tot.spp"      "bci.spp.info"    "mort.out"        "pop.change.ba"
```

Note that the file *basal.area.r* contained 6 functions. These functions are now all available in the GlobalEnv along with the preexisting objects. (See use of *ls()* above).

str(): displays the structure (i.e. its type and associate information about the type: list, data frame, vector, etc.) of an R object as well as parts of the object itself to allow the user to address and use the object properly. The name of the object that you want to see the structure of must be within the R environment and must be provided to the function as an argument.

```
> str(bci.spp.info)
`data.frame':   318 obs. of  8 variables:
 $ sp      : chr  "acacme" "acaldi" "acalma" "ade1tr" ...
 $ genus   : chr  "Acacia" "Acalypha" "Acalypha" "Adelia" ...
 $ species : chr  "melanoceras" "diversifolia" "macrostachya" "triloba" ...
 $ family  : chr  "Fabaceae:Mimos." "Euphorbiaceae" "Euphorbiaceae"
"Euphorbiaceae" ...
 $ grform  : chr  "U" "S" "U" "U" ...
 $ repsize : int  4 2 2 10 4 20 16 4 4 20 ...
 $ breedsys: chr  "B" "M" "M" "D" ...
 $ maxht   : chr  "6" "6" "5" "5" ...
```

save(): writes an R object(s) to an external file on disk. The argument(s) are the names of object(s) in the R environment to be saved as well as a specification of the file to which to save. The full path must be specified if the current directory is not where the file should be placed. The file name must be in quotes.

```
> save(bci.spp.info, tst.bci95.spp.rdata, myspp, file = "SppAug04.rdata")
> save(mort.out, file="mort.out.rdata")
```

rm(): removes R object(s) from the search path. By default, it removes from search path 1, but by passing the function a value for its parameter "pos", one can use the function to remove objects from the other search paths.

```
> rm(bci.spp.info, tst.bci95.spp.rdata)          #removes objects from search path 1
>rm(myspp,  pos=3)              #removes myspp from search path 3
```

detach(): detaches a database, i.e., removes it from the search() path of available R objects. Usually, this is either a data frame which has been attached or a package which was required previously. It does not detach objects from search path 1 nor can it be used to detach the package called base. By default the object in search path 2 is removed. Provide the search path number to remove an object from a specific location. When an object if removed, all remaining object advance one search path.

> *detach()*

history(): is used to display a history of the previous commands in the workspace. By default, the function displays the last 25 commands in a separate window. For example,

> *history()*

If the user wants to see a specific number of commands, he/she must pass the number of commands as an argument in the function call. For example, the following will display the last 10 commands:

> *history(10)*

Previous commands can be displayed directly in the R console box with the use of the back arrow key. The previous commands can be edited and reentered.

gc(): causes a garbage collection, or memory reallocation, to take place.
Although memory allocation and reallocation takes place automatically, sometimes things can bet "boloxed" up in memory by accidental overwriting or pointers to memory locations being lost. This is not often obvious to a user until file contents change unexpectantly or an object that was accessible becomes inaccessible without cause

> gc()

1.6 **Using a Text Editor for Programming**

R Console is very convenient when you wish to immediately see the results of a single command line. However, oftentimes the user may wish to write several lines of code before seeing any results. In this situation, the user could write a series of commands in a text editor, save the commands as a simple text file in $R_Home with the extension ".r" and then *source()* this file to make its contents available in R. This is how all R functions are written.

Windows comes with two simple text editors: WordPad and Notepad. There are also numerous additional freeware and shareware text editors that can be downloaded off of the internet, such as Alphatk (http://www.santafe.edu/~vince/Alphatk.html), Winedt (http://www.winedt.com/) and EMacs (http://www.gnu.org/software/emacs/emacs.html#Obtaining). For the MacOS X, AlphaX or Emacs work are both powerful text editors and have capacity to keep track of complex function development projects.

DO NOT use a word processor.

1.7 Installing and Loading Extra Packages

A long-term goal of CTFS is to develop a package distribution of its functions, datasets and documentation. The CTFS package is ready for distribution and this document is part of that package distribution. Keep a look out for updates as this is a package in rapid development and bug fixing.

Packages that do not come pre-installed into R are generally distributed as compressed files for Windows users. In order to use one such package, a user should uncompress, or extract, the file into $R_HOME\library. This should create a folder under library named after the package name.

Occasionally, the extractive process creates an extra folder bearing the packages name into which the actual package folder and all of its files are placed. It is necessary to eliminate this extra folder so that the path to the package is: *$R_HOME\library\package name* rather than *$R_HOME\library\package name\package name* in order for R to recognize the package as installed.

Once the package is installed in the *$R_HOME\library,* the package must be attached to the current R session. Do this selecting the *Packages* menu. And then select the "*Load Package…"* option. A box will appear that lists each package contained in the user's $_Home\library folder. The user should highlight the name of the package (varies based on version) and select OK.

For MacOSX, there is the *Packages&Data* menu that provides options for installing and loading packages and datasets. Packages can be downloaded, installed and loaded using these menus. Just remember that after installing the package still needs to be loaded into the current R session. This can be done in the same set of menus.

library() loads a package from *$R_HOME/library* into the current R Environment.

> *> library("CTFSAUG")*

This loads the entire contents of the CTFSAUG package into the GlobalEnv. *library()* is a *load()* command not an *attach()* command.

Additional packages are maintained and distributed by CRAN (http://cran.us.r-project.org/src/contrib/PACKAGES.html)

1.8 Customizing your R Environment

The various characteristics of the R Environment are set in R preferences which is used each time R is run. This file can contain many user specific preferences including commands to *load* certain package(s) into one's workspace upon initiation of a new R session. A Using a text editor, the user should open the file $R_HOME\R\rw1091\etc\Rprofile. At the bottom of the file, he/she should add the following line: "*library("Package Name")*". We strongly recommend that users do not change anything else in *Rprofile*.

Alternatively, one can make a *mystartup.r* file that contains commands you would like to have executed for each new session of R. This can be the first file you source upon running R. Here is an example of some of useful things to do to make the R environment easy to work in. The lines in such a file can be easily changed to accommodate the changing needs of an ongoing analysis or function development project. All lines that begin with "#" are comments and are not executed.

File: *CTFS.development.setup.R*

```
# install and load most recent version of CTFS package AND survival
library(survival)
library(CTFSAUG)

# attach datasets by changing working directory and attaching desired files
setwd("/Users/phall/Documents/ /CTFS R&D/CTFS datasets 2003/BCI")
attach("bci.quad.info.rdata")
attach("bci.spp.info.rdata")
attach("bci9095.full.rdata")
attach("bci95.mult.rdata")
attach("bci95.full.rdata")
attach("tst.bci95.full.rdata")
attach("tst.bci9095.full.rdata")

# run help for html files
help.start()

# set working directory where functions under development are being kept.
setwd("/Users/phall/Documents/CTFS_Package_Development/ R programs dev")
```