Amazon-PIRE mini course in:

Data Management and Analysis in R: Creating Repeatable Workflows

By Scott C Stark, Brad Christoffersen, Steve Wofsy

One of the key challenges in science is data management and analysis.

Unfortunately, there is often little formal education offered in universities about how to manage data that you may collect, receive from others, or download.

Furthermore, formal training in statistics often focuses on statistical theory, or the details of applying particular statistical tests, with little consideration about how the overall process—from data collection through data analysis and publication—should be executed and archived.

A key goal of the Amazon-PIRE is to provide a strong introduction to this process. As researchers draw on increasingly detailed and diverse sources of data to confront the questions of global change in tropical forests, having strong data management and reliable data analysis skills is becoming a must for all researchers. The Amazon-PIRE course is here to help. We will start, before the field course, with this short tutorial that all students are required to complete. We have tried to make this tutorial a useful reference and we hope that you will enjoy working through it.

Uma nota para quem fala português: Tem links em português embaixo desse documento com informação sobre o uso do programa computacional e estatística R que a gente vai introduzir aqui. Podem ser utilizados para aumentar e clarificar o nosso documento.

Part 1, Repeatable Workflows

After collecting or receiving data our first action is often to take a look in a spreadsheet program, usually Microsoft Excel. We often notice errors, missing values, unnecessary place-holder and counts columns etc. In short, we find that the data set needs to be cleaned. Here is where we can get in trouble; if we are not careful, before we know it we have lost or altered the data in an irreproducible way. Down the line we might realize that we deleted a column that was essential or we might find unexpected or nonsensical results and wonder if it is because of a data clean up error that we made in Excel. Or, we might submit a paper and have a reader or reviewer find an error and bring it to our (or the journal editors') attention. The problem is that the work cannot be repeated!

Here we propose a scheme as a solution to this problem. You will likely develop your own scheme in time but we think that we have laid out the major elements that any good scheme should include. One major element of our scheme that may be new to some students is the use of an object oriented programming language to create a document of 'code,' also called a 'script,' that takes the data from a raw format to a final product,

including analysis, whenever we want it to. More on this in a moment but first we present the basic scheme:

1. Create a Folder (Directory) for the project with an informative name.

2. Convert your raw data file to a text format. The easiest is a comma separated, ".csv," format.

3. Save the original file in a read-only format with an informative name such as EXAMPLE_FILE.data.archive.csv

4. Open the computer program "R: A Language and Environment for Statistical Computing" and create an "R Document ('.R') file" with an informative name. This document is known as a ***script*** or ***R script***.

5. Import the text data file to R, making sure to type all commands into the R Document file.

6. Clean up and manipulate your data with a progression of 'code' and 'comments' documenting your steps in your R document.

7. OPTIONAL: Create multiple associated R Doc files if the project is complex or synthesizes many data sources. List the associated R docs at the top of each file and collect these in your File Folder.

8. Save output files, if appropriate, to your File Folder with an informative name that designates the file as output and contains the date, Clean_Amazon_DATA.csv.

The beauty of this scheme is that years after the original work is done it will be possible to open your R Document (or Documents) and immediately recreate all your work with a few clicks or keystrokes! It should be noted that the with the increasing politicization of issues surrounding global change and ecosystem and land use change in the Amazon it is particularly important to follow these sorts of 'transparent' practices. In the future, it may even become common to publish some or all of these workflows as accessory documents with scientific articles, best to get the jump on this now!

Note for Windows and Mac users: R comes with a Graphical Users Interface for Mac and Windows obtained by clicking on the icon (Windows) or opening R.app on the Mac. These include the capability for writing, saving and running scripts. However, there are some better scripting applications that help you with "syntax highlighting", showing when commands are being used, balancing parentheses, etc. For Windows these can be installed from this tutorial website (Tinn_R or Notepad++), and for Mac (or any system) use the java installation of Jedit.app http://www.jedit.org/index.php?page=download .

Part 2: Implementing the Repeatable Workflow Scheme

Here we describe how to follow steps 1 – 8. When we get to the parts that require the R programming language we will ask you to follow R tutorial documents (that you open in R) that are integrated with this tutorial. As a learning aid, we require that you complete a simple assignment after going through this mini-course. Much of this will be implementing and organizing the examples we give in your own R document. For this reason, we introduce the assignment before you start on the next page; you may wish to print it out for reference.

Assignment: Create a Simple Workflow

For our assignment and as an example, we are going to look at data on direct and diffuse radiation through time in the Tapajós National Forest near Santarém Pará. Diffuse radiation is comprised of light incident form many angles that has been scattered (most often) by clouds while direct light arrives from the direction of the sun in clear conditions. Diffuse light may permit a greater quantity of light radiation to penetrate deeper in the canopy because light incident from many angles can pass through more holes in the canopy. Thus the ratio of diffuse to direct light, not just the total quantity of light, has consequences for photosynthesis (Gu *et al*. 2002). Lets say we are modeling photosynthesis in the Tapajós. We would like to know what the distribution of direct and diffuse light is through time. What factors predict this ratio? Does it change with time of day or with the total intensity of light? Lets take a raw dataset and ask these questions, creating a workflow R script while we do. For convenience we have provided the raw data in a '.csv' file "direct.diffuse.data.csv". Everything you need to complete the assignment can be found in our tutorial but we ask you to go beyond 'cutting and pasting' to start—if this is new to you—building your understanding of how R 'syntax' (the grammar of the language) is related to R output.

Specifically:

a) Create an R document with an informative name that begins by importing the data and removing all rows that do not have numeric data as demonstrated in the R-Tutorial R Document files we work through below.
b) Calculate sun angle.
c) Plot diffuse fraction vs. solar zenith angle and time of day (*Two figures, required features are numbered*). Label (*i*) the axes and (*ii*) title both plots. Also, (*iii*) put both plots in the same R window. Make the points different colors in the two plots (*iv*). Lastly, (*v*) make the points solid and smaller than normal in one plot. Save these figures.
d) Plot diffuse fraction vs. total radiation. Label the axes and 'main' and save the figure. Test for a relationship using a regression method of your choice from the examples. Add a label to the plot with the R^2 and P-value from your regression and save the figure.
e) Save your complete R script '.R' file. Save the cleaned-up data object as a '.csv' file and as a '.Rdata' file.
f) Hand in the final version of your R Document file. (No need to hand in the '.pdf' figures, or the '.csv' or '.Rdata' data files because, remember, we will be able to see if you created these successfully by looking at the code in the R script.) Email your file as an attachment to amazonpire@arizona.edu with the subject line "YOUR NAME; R Assignment."

Gu, L., D. Baldocchi, S. B. Verma, T. A. Black, T. Vesala, E. M. Falge, and P. R. Dowty (2002), Advantages of diffuse radiation for terrestrial ecosystem productivity, J. Geophys. Res., 107(D6)

Steps 1 – 3 of creating repeatable work flows: Basic File Management.

Step 1: create a project folder: You will need a convenient place to put your data files and the scripts that will analyze them. Since the path to this folder will have to be specified in your scripts, keep the location easy to find. Do not include spaces, or any symbols other than letters, numbers, and "_" (which will not work in all operating systems). For example:

c:\pire [for windows systems]
c:\Users\Saleska\AppData\pire [e.g. of application data directory in windows 7]
$HOME/pire [for (Linux or Mac)]

Note that you will need subfolders within each of these for data, scripts, etc.

Step 2-3: Create a text file (if necessary), saving as read-only: Once your folder is set-up and you have copied in your raw data file the next step may be to convert your raw file into a more useful format. We suggest that you convert your raw data file to a text format, preferably '.csv,' **without altering it in any other way**. Save this raw data file in a read-only format with an informative name such as "example.file.data.archive.csv". To save as read-only you will need to access details and permissions of the file:

Windows: 'right click', select "properties, and check "read-only"
Mac: 'right click'/'control click' then change permissions to "read-only".

Using the Command Line

Other common methods to convert text files between formats can be found in the **Command Line** interface of the **Operating System (OS)** running on your computer. For Windows, the command line OS is MS Dos, accessed from a Dos Prompt executable file. Mac OS X runs on Unix, accessed through "Terminal," while there are several common varieties of the Linux OS (very similar to Unix) such as Ubuntu. Linux can coexist on a PC with Windows as a second or **Duel Boot** option. To interconvert text file formats take a look at the Unix/Linux functions '*tr*' and '*awk*;' access their manuals by typing, e.g., "*man tr*" in the terminal window. One of the few things that, in our experiences, require you to use the command line is splitting large text files into smaller ones. Check out '*split*' in Unix/Linux. Files over a gigabyte or so in size (e.g., LiDAR remote sensing data) need to be split into smaller pieces to be efficiently processed in R. Finally, note that the command line does not like files with spaces in them, name files with '_' or '.' instead. Use quotations in commands when there are spaces in the file name, e.g., *split "file 1.txt"*

Unix Tutorial: www.ee.surrey.ac.uk/Teaching/Unix/unixintro.html
MS Dos Tutorial: www.best-of-computing.com/command-prompt.html
Windows Linux Dual Boot: help.ubuntu.com/community/WindowsDualBoot

Most are familiar with creating nested file folder systems, however, converting the initial data to a comma separated text file ('csv.') may require a less familiar procedure. This is the only step in which some use of Microsoft Excel is necessary to convert an Excel format into a text format by selecting 'save as' and "CSV (comma separated)". Care

must be taken though, Excel often auto-recognizes date and other special 'cell' formats. These special formats are often corrupted when using 'save as' to covert to the '.csv' format. Text files saved in other formats can usually easily be read into R (Excel can handle text conversion too, though we advise against taking this rout as much as possible). While we focus on the importing '.csv' to R in our tutorial we will show you where to look to learn about importing other formats.

Note: Use an informative file name: First, months and years after file creation you will appreciate your informative file name when you attempt to reconstruct your work. Second, a key technique of efficient computation is to be able to **rapidly search through your files using a search command**. Mac has an excellent search function built into finder, Windows versions from Vista on have had good built in searching (sorry XP users, your search puppy is terribly slow), while Unix/Linux operating systems offer the '*find*' command. You should take some time to lean how to search your files and folders; logical, complete, file names will help you do this efficiently. Note that searches can look *inside* files too. So, try to include search terms in text at the beginning of files like R scripts.

Step 4: R (A Language and Environment for Statistical Computing) and Your R Script

Lets get R installed on your machine if it is not already there. Thanks to the 'binary' files available on the R Project homepage this is very easy. Go to http://www.r-project.org/ and click on the CRAN link, then choose your geographical location. There are 'mirror' websites in the US, Brazil and many other countries—select the one closest to you. You will arrive at a page where the top links take you to the most recent executable files for your given operating system, e.g., Windows. Download the one for your system and install it like you would any other software package. Find the R application in your programs or applications file folder or in your startup menu and open R. Find the window called the consol and type in verbatim or paste the following text:

```
for (i in 1:100) {print(rep("I am R, I am running!", 3))}
```

Something should happen; output should appear in the consol. If it does, you are up and running. If not, you are likely either not in the consol or the text has not been entered correctly (try typing if you pasted; if that does not work let us know). Note that this code is an arbitrary test of the system; do not worry about understanding what it is doing yet. To see something simpler, try using the consol as a calculator, e.g., (1 + 6)*5.

The next step is to create a script file. The **Graphical User Interface (GUI)** for R includes a source code editor with highlighting built in, just open a new document under 'file' in the menu. Once you have a script document open, we suggest putting a header at the top, for example:

```
############################################################
####  Script to Analyze Diffuse and Direct Radiation and Diffuse Fraction
####  THE DATE
```

```
####  YOUR NAME
#############################################################

#### Note: Diffuse Fraction = Diffuse Radiation/ Total Radiation
```

Understanding R

R has many uses—it can manipulate and manage data, conduct classical statistical analyses such as linear modeling, ANOVA, and boot strapping analysis. It can also be used simulate the real world by combining environmental **process models** with probability distributions and random number generators. It is a complete computing language with a focus on statistics and data that is **object oriented**. Object orientation means that a wide variety of computational packages can be grouped into named 'objects' that have certain properties and attributes. Here is an example: say we use the linear model function to do a regression, we would enter something like "lin.mod <- lm(y~x)" into the R console; "lin.mod" is now an object that contains the outputs of our statistical test including information or '**attributes**' that indicate what test was performed. This object can now be used with a variety of tools (functions) in R that 'know what to do' with this class ('lm') of object; predict() calculates points on the regression, summary() calculates statistics including p-values, plot.lm() represents the results graphically etc. In some instances the *attributes* of R objects can be changed. Maybe the best example of this is changing the names of rows and columns in a spreadsheet like data frame objects. In this case we see that the attributes are a sort of data about the data or metadata. The R object definition is very flexible; R objects need not include actual data. Functions that you define to carry tasks—even very complex ones—are R objects. This flexibility is one of the strengths of the R.

The # symbol tells R to ignore this line, so the symbol can be used to divide your document and create headers as the example demonstrates. Save your R script document in the project folder you created in Step 1, above. The next thing we suggest is to copy-and-paste or type the piece of R code from above into the script document. Do NOT place a '#' in front of it as it is *code* not a *comment*. Now highlight this line with the mouse or shift and arrow keys. To run this code segment:

Mac: *press and hold the 'command' (apple) key and hit 'enter' at the same time;*
Windows: *press "Cntrl-r"* (for run)

The code should run in the consol—this is how to run code form the R script. You can run lines of code individually—as long as the syntax is complete (i.e. "as long as it is a full sentence of code") —or as larger chunks by highlighting multiple lines at once. Now, you can delete this test line of code if you wish. (Note: Use shortcut keys to cut, copy and past text. In windows these are 'control' + 'x', 'c', or 'v' and on macs they are 'command' + 'x', 'c', or 'v')

Steps 5 – 8 Are Covered by the ***R Tutorial*** **:**
(http://www.seas.harvard.edu/~swofsy/PIRE/PIRE_2010_R_Tutorial.htm)

The tutorial is based in R script documents. You may wish to move these documents to the same folder that you created for the diffuse and direct radiation. Also, move the data file "direct.diffuse.data.csv" to this folder. Open the first document and complete it by entering the examples line by line, as we have shown you how to do with the first 'test' example above. Try your own modifications to the code. Use online resources to help better understand—these are listed at the end of this document. After you have gone through the first document, review your assignment and modify the examples into your assignment R script document. Repeat the process with the second tutorial document. Finally, return here for some more tips on advanced use that will hopefully make more sense after you are better acquainted with R.

Managing More Complex Projects in R (Step 7 of Our 'Scheme')

If you rely on R (or a similar computing language like MATLAB or IDL) to manage much of your research by creating workflows, you will soon find yourself dealing with many R scripts. Often multiple sources of data must be synthesized for analysis and each data source may require its own separate processing step. It is best to keep each step in its own R script file. The output of these separate script files can then be collected in another R script that is aimed at synthesizing the data and often conducting a statistical analysis. You can think about this as a sort of hierarchy of files. How do you get the output of separate R scripts into another analysis? The simplest way is to open all of the scripts and run them sequentially; this will add all of the associated R objects into the **R Workspace**. The problem with this is that if you are not careful you might accidentally name objects the same thing and inadvertently write over one important object with another. A better method is to either (*i*) turn each script into an R function that only outputs key data that can be turned into an object by calling the function or (*ii*) save key output in '.Rdata' or '.csv' files. When an R script runs quickly it is often better to turn it into a function which can then be entered into the R workspace by using the function 'source().' On the other hand, when the script takes a looong time to run and you wish to use multiple R objects it is often best to save the output as a '.Rdata' with 'save().' Use 'load()' to import these files. When the script is slow and the output is in a single data table you may wish to save as a '.csv' text file with 'write.csv().' This format is good for data sharing.

External Resources:

Basic Data Import, Entry, and Manipulation. Also, reviews how to do stats 101-type basic probability calculations in R. We strongly suggest that all students read through "1. Basic Input," "2. Data Types," and "3. Basic Operations."

http://www.cyclismo.org/tutorial/R/

This is an advanced tutorial. Is concise and informative, but assumes a high level of familiarity with statistical computation. A wider range of advanced statistical techniques are covered.

http://www.statmethods.net/index.html

Using the R-Project Web Site to Search R Archives

The r-project web site, http://www.r-project.org/, has links to several R search functions. Search can be found under the R-Project header in the column of text on the left of the screen. The first link for the 'R Search Site' is a good option—take a look. There is a text box in which to enter your query and a series of checkable boxes below that allow you to specify what you resources you want to query. We suggest that you check only 'Functions,' 'R-help 2008-2009' and 'R-help 2010' to start with (add older help files only if needed). Functions gives you access to the same documentation as the built in R-manuals *plus* documentation for functions in packages that are not loaded on your computer (allowing you to find things you might want to load). The R-help folders are user forums in which people post questions. These can be really helpful but also frustrating at times. You may need to develop your R vocabulary a little to use the R search site. Try to use the sort of vocabulary that you might find to describe R functions in the help documentation and try entering search words (in multiple searches) corresponding to different ways that others may have tried to ask the same question. With a little patience this approach to getting help can really pay off.

When you find a package that you want to load on your machine to access one or more of its functions you can find a pdf manual in the 'packages' link in the R-Project web site too. The pdf manuals are usually very useful as they describe all the functions in the package. You may find more that one function that will be useful. Packages are alphabetically listed on the web page. Jump to the package you want (there are hundreds) by using find in your web browser; this can be accessed by 'control' + 'f' in Windows and 'command' + 'f' on macs.

R Books We Recommend:

Dowloadable Introduction(!):

> *An introduction to R* by W. N. Venables, D. M. Smith and the R Development Core Team. http://cran.r-project.org/doc/manuals/R-intro.pdf

In the Use R! series:

> *A Beginner's Guide to R* by Alain F. Zuur
>
> *Data Manipulation with R* by Phil Spector.

Em Português:

Tem várias introduções à linguagem estatística e de programação completa R no web. Encontramos essas, por exemplo, que são talvez boas. Procure mais em Google usando "linguagem estatística R" ou "programa computacional R" etc. se quiser. Se você achar umas boas, favor, nos avisa. Obrigado!

> http://leg.ufpr.br/Rpira/Rpira/

http://feferraz.net/br/P/Aprenda_a_usar_o_R