1 November 2016

TO: RSessions file
FROM: Al Cooper
SUBJECT: Intro to R and RStudio

Access: Some Recommended Set-up Work

I'm going to use barolo.eol.ucar.edu, a linux machine, and occasionally my linux or my Windows laptop, for these sessions. R and RStudio are available on barolo or tikal, but you may also want them on your own machines or laptops. The central repository for R is CRAN (CRAN)http://cran.r-project.org/, the Comprehensive R Archive Network. That page has links for downloading versions for Windows, Mac, and linux. For RStudio, you can download from http://www.rstudio.com/products/RStudio/. Or submit a support request to EOL/SIG to have R and RStudio installed. To run R (without RStudio) on barolo or tikal or any linux machine where it is installed, just type "R" into a terminal window[1] and you get the classic command-line interface that has been in use since at least 1998. But I'm going to work in RStudio for these sessions. To start your RStudio session, point a web browser to barolo.eol.ucar.edu:8787 or tikal.eol.ucar.edu. You must be inside the UCAR firewall or use VPN, and you will have to sign in using your EOL user name and password. I suggest using barolo because there are some inconsistencies between the two EOL systems and the libraries built on one may not work on the other.

My suggestion for getting started is this:

1. Make a new directory "RStudio" in your EOL workspace or on the machine you will use, named '~/RStudio'. You can do this using a terminal or using the RStudio interface; for the latter,

   (a) click the 'Files' tab in the lower right pane

   (b) use the 'New Folder' item just below that tab to create the new folder named 'RStudio' under your home directory.

2. On barolo, if that is where you will be running, copy this line into a file named .Renviron in your home directory on that system:
   R_LIBS="~cooperw/R/x86_64-redhat-linux-gnu-library/3.2"
   (Watch for '3.2' to change in the future as new versions of R are installed. Select the latest is the listed library directory.) This provides access to a library of routines that I will be using in these sessions. The appendix to this memo discusses how to download that library and install it on another machine.

3. If it's not already running, start RStudio; for barolo, you can use the link above.

4. Go to Tools -> Global Options -> R General

[1] To exit, type "quit()"; reply 'n' to 'save workspace' for now.

   (a) enter '~/RStudio' (without quotes) in the line that says "Default Working Directory". You may need to use the "browse" button and search for and select the directory with that name that you created earlier.

5. Go to Sweave under the same window

   (a) in the top where it says "Weave Rnw files using:", select 'knitr'. You may get a message saying knitr is not available if you are not on barolo; in that case, knitr will be added to the available packages when you follow the steps in the Appendix for installation on a personal machine.

6. Go to "Git/SVN" under the same window

   (a) check "Enable version control". I'll be using 'git' for these sessions, so this will give you access to my code via git.

7. For other options, the defaults are probably reasonable to start, so click "OK" to save the options.

8. Make a new project called RSessions for the purpose of these meetings:

   (a) Go to the 'File' menu and select 'New Project'
   (b) Select 'Version Control'
   (c) Select 'Git"
   (d) Use this repository name: https://github.com/WilliamCooper/RSessions.git (or alternately git@github.com:WilliamCooper/RSessions.git).
   (e) Give the project the name 'RSessions' and create it as a subdirectory of '~/RStudio'
   (f) At this point, RStudio will switch to windows that apply to this new project and copy from the repository into your new project.

9. If you are preparing your personal machine, you may eventually want LaTeX, and you may need to install git, although it seems to be standard on linux and Mac machines. These are both available on barolo. If you are installing on another machine that doesn't have them try these links: For LaTeX, http://miktex.org/ and for git http://git-scm.com.

A quick tour of the RStudio window panes:

You have seen the 'Files' window before; click on that and "Open File" to see the files in RStudio/RSessions. Then click on 'Session1' and 'RSessionsIntro.Rnw' to have something to look at. It will appear in the top left window. Don't worry about the details of what is there just yet; we'll get back to that. First, notice that there are four panes visible. At start-up, you will usually see these panes:

1. Top left: The source editor, where a working file (or many working files in tabs) are displayed.

2. Bottom left: The console, where you can enter R commands just as you would in the command-line mode that you would use without RStudio. Try it now: Click where the '>' prompt is displayed, type 2+2 [enter]. I use this frequently as a very capable calculator, because the math functions and R functions are all available here. Try 'pi', 'rnorm(5, 3.5, 0.5)', plot(0:360, sin(0:360 * pi / 180), type='l'), and 'Ranadu::WetEquivalentPotentialTemperature (P=800., AT=25., E=Ranadu::MurphyKoop (DP=20., P=800), w=0).

3. Top right: This pane has several functions.

   (a) The 'History' tab displays the history of what you have entered in the console, along with options to transfer lines either to the source editor or to the console for re-execution, possibly with editing (although that function is available more conveniently from the console using the up and down arrows to move through the history).

   (b) Another tab will show the environment, where defined variables for your workspace are shown. [Try X = 5 in the console and see what appears here.] There is also a useful tab for importing a data set from either a web location or a text file.

   (c) This pane also has a 'Git' tab if you have set up the project as recommended above, where you can monitor the status of what has changed since you last downloaded from the repository and can commit back to your local repository (although my external github repository does not allow you to 'Push" back to that repository – you will have to set up your own github repository for that).

4. Bottom right: Again, there are several functions.

   (a) Plot: If you tried the examples in item 2 above, you will have noticed this pane switch to 'Plots' and display the latest plot. RStudio also maintains a history of the plots that have been generated, and you can skip back through that history, which seems endless in my experience. The 'export' function in the 'Plot' tab allows you to save the plot externally to either png or pdf format.

   (b) Files: This displays the files in the working directory, which for now should be the RSessions project directory. There are a few functions for dealing with those files, including to delete, rename, or move.

   (c) Help: Requests for help are displayed in this window. If you click on the 'Home' (dog-house-shaped icon) button, you are taken to a page that shows links to some of the basic R and RStudio manuals and also a definitive definition of the language, along with other more specialized manuals. If you want to know how a particular function works, you can enter your search phrase into the console as, for example, ?atan2 to see information on the two-argument arctangent function, or you can enter this into the entry line on the help window (with the magnifying glass), which has the advantage that it displays auto-completion options as you type. [Try 'plot' to see this in action.]

There are many cross-references among the help pages that are quite useful. If you click on the left drop-down menu you will see a history of the help topics you have consulted.

(d) Viewer: I'm going to postpone discussion of this until later; we won't need the viewer for a while.

(e) Packages: This displays the packages that are available for your use, either in the system-wide repository or in mine (if you have established assess to mine via one of the methods discussed earlier or in the appendix). To test if you have access to my functions, look for 'Ranadu' in the list. If it is there, let your mouse linger over the version number and its location will be displayed. You could try this for other packages also; check, for example, ggplot2, a very useful and standard plotting package not yet part of the EOL repository. The check boxes indicate which packages will be available in your session without special loading; if unchecked, you can either check the box or include a statement similar to 'require(Ranadu)' in your session (or, alternately, when you use these functions use the form Ranadu::routine_you_want). The 'install' and 'update' functions at the top of the 'Packages' pane provide easy access to the 'CRAN' archive of packages or to packages you may have from other sources that you want to load for use.

Practice With This [suggested, start before next time]

As homework, in addition to working through the preceding information to ensure that you understand it and can do all the steps, pick a few of the following and, where I have provided sample code, extend them to other data:

Easy ones:

1. Plot a time history of the wind speed from DEEPWAVE flight 20 and compare the plot to one generated by ncplot. Hint: See the next example for an illustration of how to access the data from this flight.

2. Construct a plot of temperature and humidity vs pressure altitude for the final descent from the same flight.

```
require(Ranadu, quietly = TRUE, warn.conflicts=FALSE) # my routines
opts_chunk$set(echo=TRUE, fig.lp="fig:", fig.width=6, fig.height=7.5)
opts_chunk$set(tidy=TRUE, digits=3)
require(knitr)
```

```
Directory <- DataDirectory()  # for portability; sets the local data directory
Flight <- "rf20"  # select a flight
Project = "DEEPWAVE"  # select a project
fname = sprintf("%s%s/%s%s.nc", Directory, Project, Project, Flight)
# XXX set variables needed, here a standard list including DPX and ATX
# preliminary look shows that final descent was from 84400 to 91100
Data <- getNetCDF(fname, c("Time", "DPXC", "ATX", "PALT"), 84400, 91100)
saveDataFile <- "RSessionsIntro.Rdata.gz"
save(Data, file = saveDataFile)
# for future runs, it will be much faster to use: load(saveDataFile)
```

```
plot(Data$DPX, Data$PALT, type = "l")  # type='l': line plot
lines(Data$ATX, Data$PALT, col = "darkgreen")  # add temperature
s <- Data$DPX > Data$ATX
lines(Data$ATX[s], Data$PALT[s], col = "red", lwd = 3)
```

```
# will show how to add legends, titles, axis labels, etc, later
```

3. Calculate the mean and standard deviation of the vertical wind (WIC) for measurements above FL400 for the same flight.

Harder ones:

1. Estimate what change in recovery factor would be needed to bring DEEPWAVE temperature measurements ATHR1 and ATRL from RF20 into agreement.
2. Plot a time history or scattergram of the pressure corrections used in DEEPWAVE flight 20 compared to the corrections recommended on the basis of the LAMS calibration. [The latter are in Ranadu:PCorFunction().)
3. Construct a wind hodograph for the same flight. (This is a display that shows the direction *toward which* the wind is blowing, in an x-y display where x is EW, y is NS, and the distance from the center point represents the wind strength. Points along the hodograph should be labeled by pressure, but this can be a future extension of this homework problem.
4. Is the correct recovery factor being used for CONTRAST flight 8?
5. – or construct your own and be ready to show it for a future session –

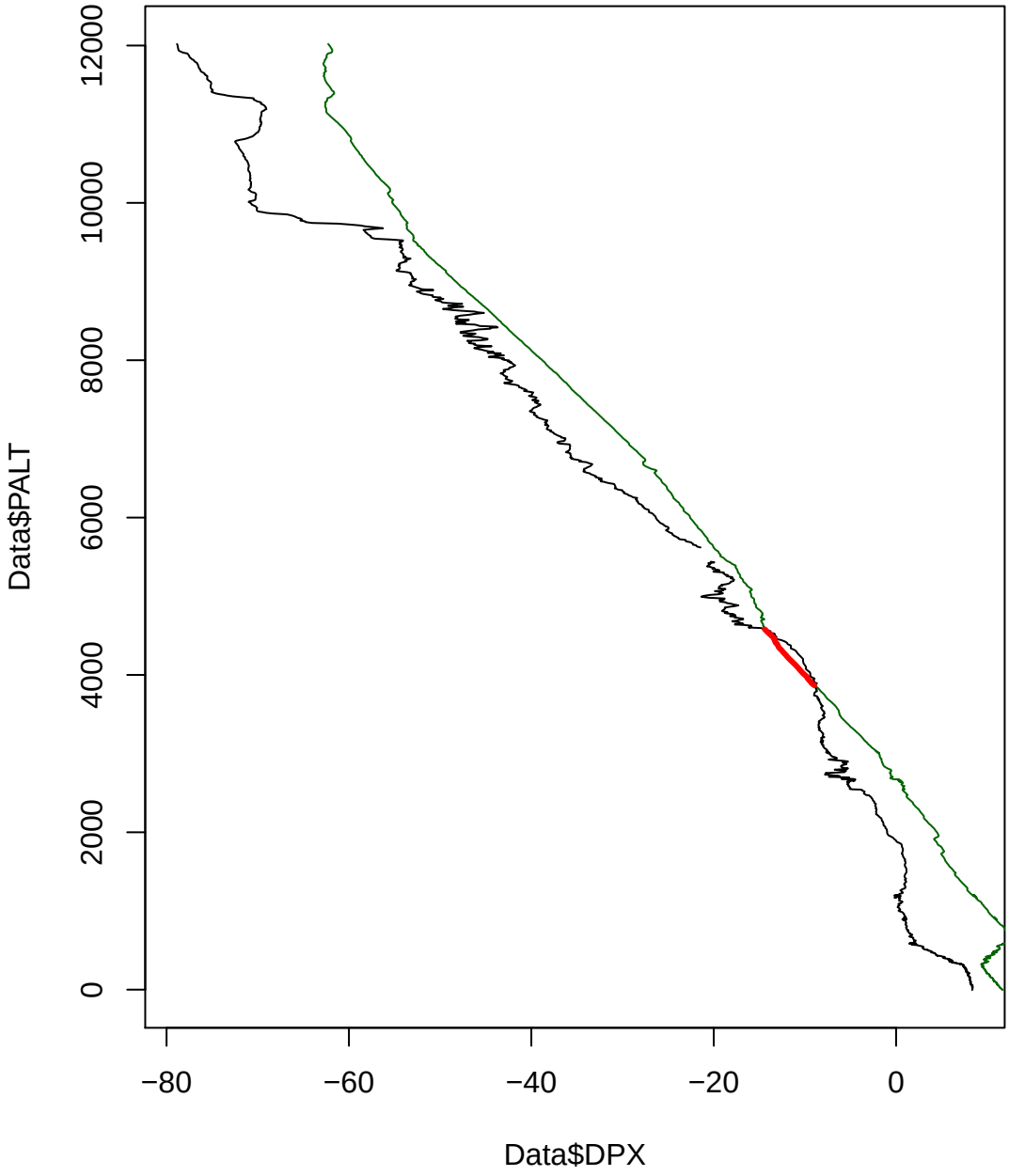

Figure 1: DEEPWAVE flight 20, final descent (8:44:00 – 9:11:00) sounding of temperature and dewpoint, with region where dewpoint exceeds temperature shown as the thick red line.

An example of what I'll be advocating

A sample program file and resulting memo are included in the 'RSessions/Session1' directory as CONTRASTcalAOA.Rnw and CONTRASTcalAOA.pdf. I'll use those next to give an overview of how files like this can be used to combine text and code to produce a self-consistent document based on code contained in the file that generates that document.

Characteristics of the CONTRASTcalAOA.Rnw file to note:

1. Text is intermixed with R code in this file. R code 'chunks' begin with a header line <<chunk-name>>= and end with @. Results from the R code can be incorporated into subsequent parts of the text.
2. The chunks provide a way to structure your program and to skip to appropriate sections of the code.
3. The text (here LaTeX) will have references to the R results in the form of '\Sexpr{x}' where x is an R variable. (The name Sexpr is a remnant from R's origin in S; Sexpr = S expression.)
4. If a figure is generated in an R chunk, it appears in the text document after the place where the R chunk appears. Captions for the figures appear in the chunk header are are placed below the figure.
5. I will be emphasizing reproducibility and have included a section on how to use this file with other data. There is also a table on reproducibility at the end of the memo, which is standard for programs like this that I write.. That table contains:
   (a) the project name
   (b) the location of the source file
   (c) the name of a zip-format archive containing all the information needed to repeat the production of the memo
   (d) the original data used. (Only the subset of the data used here is saved in the archive to keep the size of these archive files small.)
   (e) the archive location on github, which anyone can access

   You will find a similar table at the end of this memo.

Next

For a preview of what is planned to follow this, see RSessionsOutline.pdf in the RSessions directory.

– End of Memo –

APPENDIX: Getting the 'RAF' ('Ranadu') Package

For use on barolo, a procedure suggested in the first section of this memo should provide access to my package of routines for use with RAF netCDF files. Here are steps for obtaining that package on other systems.

You may first need to install some packages on which it depends. Its dependencies are: ncdf4, maps, ggplot2, ggthemes. It also will import some additional packages when needed: tcltk, nleqslv, zoo, fields, stats, signal. You will also want knitr for generating text. If any of those do not appear in your list of packages (displayed by selecting the 'Packages' tab in the bottom right pane of RStudio), click 'Install' at the top of that pane, select 'repository (CRAN)' in the first drop-down menu, and enter a list of packages that you are missing in the second line labelled 'Packages'. The default location for local libraries on the last line should be kept, and the check-box labelled 'Install dependencies' should be checked. Click 'Install" and those packages should be downloaded from the CRAN repository, compiled where necessary, and added to your list of packages. After doing that, to get the package 'Ranadu', follow these steps:

1. Within RStudio, select 'File' -> 'New Project' -> 'Version Control' -> 'Git' (clone a repository)

2. For the package URL, enter 'git@github.com:WilliamCooper/Ranadu.git' (without the quote marks). Use the package name 'Ranadu' and make the directory for it a subdirectory under 'RStudio'.

3. Click 'Create Project' and RStudio will download the files and change the working project to Ranadu.

4. Go to the 'Packages' window and click 'Install'

5. In the new window that appears, change the top drop-down menu from CRAN to 'Package archive file'

6. A window will appear that displays all the files in your home directory. Go to the directory RStudio, then to directory Ranadu, and in that directory select 'Ranadu_2.4-16-06-26.tar.gz' (or whatever the highest-version-number is there). If installing on a machine other than barolo, select the tar file appropriate for your architecture, -Win, -Mac, or -std. The standard archive should eventually work on all systems but Windows and Mac may have temporary compatibility problems with parts of the package so for now you need to install these special packages.

7. For 'Install to library ...' you should be able to use the default for your system. It should point to a directory in your local file space. On my laptop it is '/home/cooperw/R/x86_64-redhat-linux-gnu-library/3.3'. This should have been defined during R installation as the directory for local packages.

8. Click 'Install'. The package 'Ranadu' should appear in your list of packages. If an error occurs, be sure you have installed the packages listed in the paragraph above this numbered list, and then tell me so I can try to help.

Reproducibility:

PROJECT:	RSessions
ARCHIVE PACKAGE:	RSessionsIntro.zip
CONTAINS:	attachment list below
PROGRAM:	/h/eol/cooperw/RStudio/RSessions/RSessionsIntro.Rnw
ORIGINAL DATA:	/scr/raf_data/CONTRAST/CONTRASTrf20.nc
GIT:	Session1 in https://github.com/WilliamCooper/RSessions.git

Attachments: ProgramFile
Document.pdf
SessionInfo
RSessionsIntro.Rdata.gz

```
system(sprintf("zip RSessionsIntro.zip %s %s %s %s", "RSessionsIntro.Rnw",
    "RSessionsIntro.pdf", "SessionInfo", "RSessionsIntro.Rdata.gz"))
```