This document walks through an R script designed to process data from a LICOR-8100A system.

The first part of the script describes what the script is supposed to do, who wrote it, and when it was edited last.

#Emily Kyker-Snowman

#February 25th, 2014

#This script reads in LICOR data, cleans up bad data, and calculates averages and #standard errors for groups defined by the user (i.e. Terrace, Ring).

Next, the script sets the working directory and checks to see if the ‘plyr’ package is installed. plyr is necessary for using the ddply function later.

#Setting up packages

setwd("~/Desktop/R")

#library(lattice)

require(plyr)

library(plyr)

Next, the script reads in a csv full of LICOR data. file.choose() lets you browse files on your computer and pick the one you want. This part also removes records in the dataset that have been marked for exclusion, prints summary statistics to the console, and saves the columns of the table as individual variables that can be called later.

#Pulling in csv file

LICORraw = read.csv(file.choose())

LICOR = subset(LICORraw, Exclude !=1)

summary(LICOR)

attach(LICOR)

This part looks for records that have a poorly fit flux curve, which is often an indicator that something was wrong with the measurement. The last line removes all records with an R2 less than 0.9. This line can be left out if you don’t want to automatically remove those records. The basic structure of this part could be used to scan the records for other errors, such as negative fluxes or records where the linear and exponential fits to flux data diverge by more than 50%.

#Scanning for weird values in dataset

weirdRecords1 = subset(LICOR, Exp\_R2<.9)

weirdRecords1

LICOR = subset(LICOR, Exp\_R2>.9)

Negative temperature values are a sign that the temperature probe malfunctioned (at least in California), so this part replaces all negative temperature values with NA so they don’t skew the averages.

#Replaces negative temp values with NA

LICOR$Mean.T1[LICOR$Mean.T1 < 0] <- NA

Sometimes, if something went wrong with a measurement, it was immediately rerun in the field and denoted as ‘1b’ or ‘3c’ etc. in the data. Since the data cleanup processes removes all the faulty measurements and only leaves the last (successful) rep of a measurement, this line trims the letters off the ring numbers.

#Truncate 'Ring' at the first character to get rid of b and c

LICOR$Ring = substr(LICOR$Ring,1,1)

This part produces a grid of scatter plots comparing the numeric variables colored by month. “col=” can be given a different argument to color the plots by terrace, year, or some other categorical variable.

#Looking at correlations between variables (colored by month)

pairs(LICOR[9:20], col=Month)

This chunk splits the data into user-defined groups (i.e. grouped by terrace and month) using one or more categorical variables and then calculates the means and standard errors for the numerical variables. To change the variable(s) used to group records, change what’s inside .() in ddply. To run a different calculation on the groups (i.e. median), use any of the lines in the body of the function as a model and change the function (i.e. mean, stdErr) to whatever function you want to run. The variable names on the left will be the header names in the table that is produced. Num\_Records is a way of seeing how many records get averaged in each of the groups.

#Defining the standard error function.

stdErr <- function(x) {sd(x)/ sqrt(length(x))}

#Calculating averages and standard errors for defined groupings.

#To group by one or more variables, change what's inside .() in ddply.

LICORMeans <- ddply(LICOR, .(Month, Terrace), summarise,

Num\_Records = length(Mean.Tcham),

Mean\_Tcham = mean(Mean.Tcham),

stdErr\_Tcham = stdErr(Mean.Tcham),

Mean\_Pressure = mean(Mean.Pressure),

stdErr\_Pressure = stdErr(Mean.Pressure),

Mean\_H2O = mean(Mean.H2O),

stdErr\_H2O = stdErr(Mean.H2O),

Mean\_RH = mean(Mean.RH),

stdErr\_RH = stdErr(Mean.RH),

Mean\_T1 = mean(Mean.T1),

stdErr\_T1 = stdErr(Mean.T1),

Mean\_V1 = mean(Mean.V1),

stdErr\_V1 = stdErr(Mean.V1),

Mean\_Exp\_Flux = mean(Exp\_Flux),

stdErr\_Exp\_Flux = stdErr(Exp\_Flux),

Mean\_Lin\_Flux = mean(Lin\_Flux),

stdErr\_Lin\_Flux = stdErr(Lin\_Flux)

)

#Prints the table of means to the console

LICORMeans