Surface Elevation Table- Marker Horizon (SET-MH) Data Handling and Analaysis in R

The Nature Conservancy - Long Island Chapter

Adam Starke

## Current Data and Analysis File Structure

This document is intended to document the process used for database storage and data analysis of our SET-MH data. Our current workflow uses Access2010 to store the data and RStudio (running the R language) for data handling. R is a powerful, though often times overwhelming tool. It is hoped that this document will aid anyone unfamiliar with R through our current data processing methods.

### Data Storage

The SET-MH method of marsh monitoring generates a large set of data that needs to be carefully maintained and catalogued. Fortunately, there has been a substaintial effort put forth by the National Park Service's National Capital Region Network Inventory and Monitoring Program to create a user-friendly Access database (DB). This DB allows for easy field data entry, data browsing and the creation of summary plots. There are also methods for exporting data to other software if desired. Managing SET-MH data in such as system is critical.  
Although the statistical analyses needed to calculate trends in marsh elevation are somewhat basic the restructuring of the data to allow for analysis can be somewhat complex. With this as a characteristic of the data we decided to use the statistical programming language R. R is one of the most widely used programming languages and contains an enormous amount of tools and scripts (known as packages in R) that make these restructuring of data possible. The ability to create 'scripts' allow for quick updates to analysis after data entry. A number of other features also exist that make R a good choice, particulary the graphics capabilities and the support for creation of 'data-driven documents' which allows for reproducible research. This document is actually written with R code embedded within it. This allows one to actually run the code that performs the analysis along side the documentation of the code. This allows the user to describe and show the exact steps taken for data organization, restructuring and analysis. Actual R code run in this document will be highlighted and have a slight grey background. This code can be cut and pasted or run as a whole document.

As mentioned the data are currently stored in an Access database located at:

[1] "T:\\Coastal\\Marsh-Wetlands\\SET-MH\_project\\SET\_Data\\SET\_Monitoring\_Database\\Database\_storage\\SET\_DB\_BE\_ver\_2.94\_TNC\_Master.mdb"

This file is referred to as the 'backend database' which holds all the raw SET-MH data.The 'front-end' of the database contains the user interface (UI) and is linked to the backend allowing multiple 'front ends' all pointing to one dataset. It's suggested that the front-end UI be used for quick checks on data and it is where **ALL** data entry takes place.

### Accessing Our Data Through R

Using the power of R's scripting language capabilities the quickest, most direct, way to update SET-MH analyses is to simply navigate to the folder containing the RStudio project file (SET-MH\_Analysis.Rproj) currently located at: C:/Users/astarke/Documents/R\_Code/SET-MH\_Analysis.

Opening this file in RStudio will load all the code associated with this project. Run the script 'MarshProject.R' by opening it and clicking the *'Source'* button in the top-right of the window. This will pull all the code from the open file and run it. You will see activity in the "Console" window confirming that it is running.

The main feature in this script is the *ProjectTemplate* package command: load.project() which automatically runs (sources) scripts contained in the '/data' folder, then cleans it up using scripts in the '/munge' folder. Munging simply means to remove irrelevant or missing values, restructure the data, transforming variable names etc. The scripts in each of these folders will be explained below. There are then three other lines of code that source code that performs the actual analysis and creates summary tables and plots. This will also be explained in detail below.

## Data Processing Steps

This section will walk through each of our data processing steps; from loading data into R to exporting plots and summary tables. Most of these code blocks have been split to allow for commentary on what the code is doing. Again, any text that is bounded and highlighted is actual R code that is run in the software. If needed this text can be copied and pasted into an active R console and run.

#### Data Loading

The first step is to connect to the data. To do so R needs to communicate with the Access database through an *"ODBC- protocol"* or translator. For this we use the package **RODBC** which allows R to talk with Access.

###  
# Data importing from SET\_Monitoring\_Database 'Backend Data Storage File' currently SET\_DB\_BE\_ver\_2.92\_TNC\_Master  
# ProjectTemplate automatically runs this script, importing the dataframes below into the R workspace  
#  
###  
  
library(RODBC)  
  
SET.DB.path <- "T:\\Coastal\\Marsh-Wetlands\\SET-MH\_project\\SET\_Data\\SET\_Monitoring\_Database\\Database\_storage\\SET\_DB\_BE\_ver\_2.94\_TNC\_Master.mdb"  
  
  
SET.DB <- odbcConnectAccess2007(SET.DB.path)

This code block creates a pathway to the Access database that is then used to communicate with the database. Once the connection is established several tables will be pulled into R as 'dataframes'; R's equivelent to a spreadsheet.

### Pull in the tables of interest containing data needed for analysis ====  
#Location data- i.e. study sites and stations within sites  
Sites <- sqlFetch(SET.DB, sqtable= "tbl\_Sites")  
Locations <- sqlFetch(SET.DB, sqtable= "tbl\_Locations")  
  
# Sites and Locations will be joined in munge by 'site\_ID'  
# Use capwords function to standardize caps and revert Stratafication to as a factor  
Locations$Stratafication <- as.factor(capwords((as.character(Locations$Stratafication))))  
  
# Sampling events  
Events <- sqlFetch(SET.DB, sqtable= "tbl\_Events")  
  
# SET Rod data  
Positions <- sqlFetch(SET.DB, sqtable= "tbl\_SET\_Position")  
SETdata<- sqlFetch(SET.DB, sqtable= "tbl\_SET\_Data")  
  
# Surface Accretion data  
SAccret <- sqlFetch(SET.DB, sqtable= "tbl\_Accretion\_Data")   
SA\_Layers <- sqlFetch(SET.DB, sqtable= "tbl\_Feldspar\_Layers")  
  
close(SET.DB) #close the connection to the Access SET database.

This code block will load 7 tables into R containing:  
\* Sites- data associated with each study site, i.e. general location, name, date of establishment  
\* Locations- data associated with each station within each study site  
\* Events- dates that samples were made  
\* Positions- ID's for each SET arm position  
\* SETdata- SET readings raw data  
\* SAccret- Marker Horizon (MH) surface accretion raw data  
\* SA\_Layers- MH layer ID information

#### Munging

Data munging (aka data wrangling) is the process of formatting and rearranging the 'shape' of the data to allow for analysis. As mentioned earlier the actual analysis of SET-MH data is not that complex. The primary crux of the analysis is getting the data into a form that makes the analysis programatically easy. The basic steps are to merge the tables brought in from the DB, retaining all relevant data from each table and then reshaping the data to allow for the analysis to run quickly.

#SQL type joins to flatten the tables from the database. ----  
  
#Study Sites with Locations data --Location\_ID is a numeric key analagous to Plot\_Name   
StudySites <- join(Sites, Locations, by="Site\_ID", type="inner")   
  
#Surface Accretion data   
SA <- join(SA\_Layers, SAccret, by="Layer\_ID", type= "inner")  
  
#SET Rod data  
SET <- join(SETdata, Positions, by="Position\_ID", type= "inner")  
  
# Sampling events and sites  
# join site-location data with sampling 'events'  
Samplings <- join(StudySites, Events, by= "Location\_ID", type="inner")   
  
# Complete datasets with ALL variables  
SA.data <- join(SA, Samplings, by="Event\_ID", type="inner")  
SET.data <- join(SET, Samplings, by="Event\_ID", type= "inner")  
  
# Remove excess dataframes leaving only SA and SET data ----  
rm(Events,  
 Positions,  
 SA,  
 SA\_Layers,  
 SAccret,  
 SET,  
 SETdata,   
 Samplings,  
 Sites,  
 Locations  
 )

The next step is to clean up some of the variables in the tables.

We now have two complete datasets that contain all our relevant SET-MH data. One set contains SET data with location, date and pin height data. One has Surface Accretion data with location, date and accretion data.  
We need to continue cleaning up the variable names and reshape the way the data is stored in the dataframes.

SET.data$Stratafication <- as.character(SET.data$Stratafication)  
SET.data$Stratafication <- as.factor(capwords(SET.data$Stratafication)) # Standardize the character format  
SET.data$Start\_Date <- as.Date(SET.data$Start\_Date) # Ensure that 'Start\_Date' is of class 'date'  
  
# Clean up SET.data dataframe and reshape to allow for analysis-----  
# 'keeps' is list of variable names to keep; 'iders' are identifier variables used in reshaping the dataframe  
keeps <- c("SET\_Type",   
 "Position\_ID",   
 "Location\_ID",   
 "Site\_Name",   
 "Stratafication",   
 "Plot\_Name",   
 "Start\_Date",   
 "Pin1",   
 "Pin2",   
 "Pin3",   
 "Pin4",   
 "Pin5",   
 "Pin6",   
 "Pin7",   
 "Pin8",   
 "Pin9",   
 "Arm\_Direction",   
 "Position\_Name",   
 "X\_Coord",  
 "Y\_Coord",  
 "Coord\_Units",  
 "Coord\_System",  
 "UTM\_Zone",  
 "Datum")  
# Identifiers used for reshape ----  
iders <- c("SET\_Type",   
 "Position\_ID",   
 "Location\_ID",   
 "Site\_Name",   
 "Stratafication",   
 "Plot\_Name",   
 "Start\_Date",  
 "Arm\_Direction",   
 "Position\_Name",   
 "X\_Coord",  
 "Y\_Coord",  
 "Coord\_Units",  
 "Coord\_System",  
 "UTM\_Zone",  
 "Datum")  
  
SET.data <- SET.data[keeps]   
SET.data <- tbl\_df(SET.data)  
# Use reshape2 to melt wide table down to a long format (really only transposing the pin readings) =====  
SET.data.Melt <- melt(SET.data, id= iders, na.rm=TRUE)

By *'melting'* the SET and SA dataframes we will transpose or pivot the measurement data to allow for calculations of incremental change in accretion and elevation, the change in date and the cumulative change that has been measured. This technique relies on a split-apply-combine strategy common in data sciences. The process first groups the data by some variables and value(s). Each grouping of data is then processed through a set of functions or formulas and the results are then grouped back into the dataset to rebuild the original dataset. The grouping on the SET data is a combination of Study site, location within that study site, the direction of the SET reading arm and finally each pin.  
Be sure to read the comments embedded in the R code blocks as they provide some additional commentary on what's taking place in the code.

###  
# Munge dates to create sample date and establishment date----   
#   
# Rename 'Start\_Date' to just 'Date' to remove confusing variable name  
SET.data.Melt$Date <- as.Date((SET.data.Melt$Start\_Date))  
  
# Create column of 'establishment dates'= EstDate ----  
# Date the site was first established  
# Use ddply to split by Location\_ID and find the first date read, record that in -$EstDate  
SET.data.Melt <-ddply(SET.data.Melt,  
 .(Location\_ID),  
 transform,  
 EstDate= (as.Date(min(Date))))   
  
# Create variable DecYear- decimal years - in SET data, for regression analysis.   
# difftime- calculates the time difference between t1 and t2 in units identified  
  
SET.data.Melt$DecYear <- round((((as.numeric(difftime(SET.data.Melt$Date, SET.data.Melt$EstDate, units = "days"))))/365),3)  
SET.data.Melt <- plyr::rename(SET.data.Melt, c(value="Raw")) #rename 'value' to 'Raw'  
SET.data.Melt <- SET.data.Melt[order(SET.data.Melt$Start\_Date),]  
  
# Calculates a 'change' used for plots primarily- regressions are run through 'raw' data to reduce chance of error.  
SET.data.Melt <- ddply(SET.data.Melt,   
 .(Position\_ID,  
 variable),   
 mutate,   
 change = as.numeric(Raw-Raw[1]),  
 incrementalChange = c(NA, diff(change)))

The same basic steps are taken for the Surface Accretion data.

###  
# Surface Accretion -----  
#  
#Clean up SA.data dataframe and reshape; same as above but working on surface accretion data  
# similar steps as SET.data  
  
keepsA <- c("Location\_ID",   
 "Layer\_Label",   
 "Estab\_Date",   
 "Measure\_1",   
 "Measure\_2",   
 "Measure\_3",   
 "Measure\_4",  
 "Measure\_5",   
 "Measure\_6",   
 "Site\_Name",   
 "Stratafication",   
 "Plot\_Name",   
 "Start\_Date")  
SA.data <- SA.data[keepsA] # Drop unneeded variables  
  
idersSA <- c("Location\_ID",   
 "Layer\_Label",   
 "Estab\_Date",   
 "Site\_Name",   
 "Stratafication",   
 "Plot\_Name",   
 "Start\_Date")  
SA.data.M <- melt(SA.data, id= idersSA, na.rm=TRUE) # melt down dataframe into tidy table.  
  
  
# Create variable DecYear- decimal years - in SA.data, for regression analysis  
SA.data.M$DecYear <- round((((as.numeric(difftime(SA.data.M$Start\_Date, SA.data.M$Estab\_Date, units = "days"))))/365),3)  
  
#SA.data.M$DecYear <- round(((SA.data.M$Start\_Date-SA.data.M$Estab\_Date)/365),3)  
SA.data.M <- plyr::rename(SA.data.M, c(value="Accretion")) #rename 'value' to 'Accretion'  
  
  
  
### All clear to hear-- 07Mar2014- AFS -----

We now have two additional datasets that contain the SA and SET data in a transposed or *'long'* format