**Formatting datasets for the core-transient project**

In this project, we evaluate the proportion of core and transient species at a given site using data collected across a wide range of taxa, regions of the world, and environmental systems. A challenge that we face is that ecological data are collected at highly variable spatial, temporal, and biological scales and it is necessary to consistently format the data in preparation for analysis. Here we provide instructions associated with formatting datasets. The goal is to format a dataset as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| datasetID | site | species | year | count |
| 01 | d01\_Treatment1PlotA | Acrocephalus\_schoenobaenus | 1928 | 1 |
| 01 | d01\_Treatment1PlotA | Buteo\_buteo | 1928 | 3 |
| 01 | d01\_Treatment1PlotB | Corvus\_corax | 1929 | 7 |
| 01 | d01\_Treatment2PlotA | Cuculus\_canorus | 1929 | 1 |
| 01 | d01\_Treatment2PlotB | Gallinula\_chloropus | 1928 | 2 |

**datasetID:**

Goal: Add a column that repeats the name of the dataset.

DatasetID’s are available from the dataset\_summary\_table. This must be the first column of your formatted dataset. To do so, you simply use the rep command in the base package and tell it to repeat the value for the number of rows of the data frame. *Note: example\_df is the unformatted dataset being prepared for analysis.*

datasetID = rep(01, dim(example\_df)[1])

**site:**

Goal: Add a column that provides a unique site ID for each site. Determining sites can sometimes be challenging the steps required to create unique site ID’s depends on how researchers have coded their site data. The examples below provide the most common solutions to create unique site ID’s

***Concatenating multiple site columns into a single site:***

In the example above, the sites were broken down into different treatments and plots. To construct the sites field, we paste the datasetID as well as any site information provided. If the above example were constructed from a data frame with the fields “Treatment” and “Plot”, the site column would be made using the following code:

site = paste(‘d’, datasetID, example\_df$Treatment, example\_df$Plot, sep = ‘’)

***Removing site information:***

Some data sources include information in the site field that are problematic for analysis. For example, some sites include the time that a sample was collected as a part of the site field. Consider a column named “site” where year is included in the field, such that the first entry might be “Treatment1PlotA1928”. You can extract the plot information easily in one of two ways:

1) Substring the plot information by from the first to the last plot character. The following code extracts all characters from the first to the 15th within the site field. This is ONLY to be used is the plot information contains the same number of characters!

site1 = substr(example\_df$site, 1, 15)

*Note: If the field is not a character field, you can convert it on the fly using:*

site1 = substr(as.character(example\_df$site) , 1, 15)

2) Substring the plot information by removing the last characters in a field. This method is valid if the number characters that make up the true site field are not the same across sites but there is an equal number of characters that need to be removed. To do so, the easiest way is to use the str\_sub function in Hadley Wickham’s **stringr** package (though this can be easily accomplished by writing your own function in base).

require(stringr)

x = “hello world”

str\_sub(x, 1, -7)

[1] "hello"

***Separating a field to extract site information:***

It is also often necessary to separate the site field by some common character (such as, in the example below “\_”). This is done using the transform and colsplit functions. Colsplit is located in Hadley Wickham’s package **reshape2**. The output of this function is a multiple field dataset containing the original data (field 1) and a column for each split. In this case, the second column contains the site information, so using “[,2]” returns a vector with just the relevant site information.

require(reshape2)

x = 'Treatment1PlotB\_1927'

site1a = transform(x, site = colsplit(x, pattern = '\\\_', names = c('site','year')))[,2]

***Using latitude and longitude to define sites:***

It is sometimes necessary to define sites using latitude and longitude if this is the only site information provided. While making decisions for the appropriate scale to analyze the data requires an understanding of the taxa and method of collection, the process of creating the site information is relatively straitforward. Here we will use the “round\_any” function in Hadley Wickham’s **plyr** package to turn decimal latitude and longitude data sites composed of 2 degree lat-lon blocks.

x = 13.35679

y = 46.87

site1 = paste(round\_any(x, 2), round\_any(y, 2))

**species:**

Goal: Subset dataset to unique species records. While it is occasionally necessary to modify species records, such as if genus and species are provided in separate fields (in which case you would concatenate the two fields as above), your primary task is to remove records that are not really valid species. This may include problems such as NA records and the reporting of observations such as “bare ground”.

***Look at the species contained within a dataset:***

To isolate problem species, it’s often necessary to look at the list of species in a dataset. We do this using the “unique” function.

unique(example\_df$species)

***Subsetting a dataset to valid species observations:***

Once we have determined which species do not belong in our dataset, we are tasked with removing those records. There are several methods for doing so; here are a few examples.

Removing NA’s:

example\_df1 = na.omit(example\_df)

example\_df1 = example\_df[!is.na(example\_df$species),]

Removing a given species record (example is records called “Bare Ground”):

example\_df1 = example\_df[example\_df!='Bare Ground',]

example\_df1 = subset(example\_df1, species!= 'Bare Ground')

**year:**

Goal: Create a time column. The two challenges that may be associated with this are if the date is provided as a date formatted object (example 01/01/2015) or if there are multiple samples per year. If the latter is the issue, the appropriate temporal scale must be determined prior to formatting the dataset (see Allen, Ethan, or myself). Once the time scale is determined, data are reported as a decimal year and simply requires a bit of math (for example, if sampling was done monthly and a sample was taken on 1 Mar 2015, the time of the sample would be 2015 + 3/12 as March is the third month of the year).

***Extracting year from a date object:***

Convert the date column to an R formatted date (in this case pretending that our unformatted dataset contains a column called record\_date):

date = strptime(example\_df $record\_date, '%m/ %d/ %y')

Add a sampling year line (summarize by year):

example\_df $year = as.numeric(format(date, '%Y'))

**count:**

Goal: Summarize the dataset to the count of individuals per species, site, and year for a given dataset. To do so, we will use Hadley Wickham’s “ddply” function in the **plyr** package. Below is an example in which there is a count column that must be summarized.

example\_df2 = ddply(example\_df, .(site, year, species), summarize, count = sum(count))