MIIN Part 1b: TRY data

Marissa Lee

June 1, 2015

Filename: MIIN_1_tryData.Rmd This markdown file does the following tasks: 1. Combine raw TRY data from 2 data releases
2. Clean and export trait data 3. Summarize the trait data file other. Clean and export trait meta-data

```
knitr::opts_chunk$set(cache=TRUE)
library(plyr)
synthdataPath<-file.path(getwd()[1], "DATA", "DATA_SYNTHESIZED", "tryData") #where to put the clean dat
```

---

1. LOAD AND COMBINE RAW TRY DATA

This chunk takes a while to run.

```
#1. Read-in txt files and rbind them
dataA<-read.table("DATA/TRY_Proposal_117_Data_Release_2012_08_26.txt",header=TRUE,sep="\t") #raw data
dataB<-read.table("DATA/TRY_Proposal_117_Data_Release_2012_09_20.txt",header=TRUE,sep="\t") #raw data
data<-rbind(dataA,dataB)

#2. Make sure that data types are identified correctly
data$ObservationID<-as.factor(data$ObservationID)
data$ObsDataID<-as.factor(data$ObsDataID)
data$TraitID<-as.factor(data$TraitID)
data$DataID<-as.factor(data$DataID)
data$OrigObsDataID<-as.factor(data$OrigObsDataID)

#3. General Info about the dataset/Column headers
#LastName
#FileName
#SpeciesName: original name of the species
#AccSpeciesName: accepted species name (USE THIS)
#ObservationID: identifies which ObsDataIDs are related to one another (USE THIS)
# e.g. two traits measured on the same leaf
# e.g. if plants were grown under experimental conditions, this is reported as a covariate entry with t
#ObsDataID: identifies the row - can be either a trait or a covariate (USE THIS)
#TraitVsNonTrait: identifies rows with trait info (USE THIS)
#TraitID (USE THIS)
#TraitName
#DataID: identifies the trait subgroup (USE THIS)
#DataName
#OrigValueStr: original value as a text string
#OrigUncertaintyStr: original uncertainty as text string
#OrigUnitStr: original unit as text string
#Unit.UnitName:  Unit name
#OrigValue: Original value
```

```
#ValueKindName: Value kind (single measurement, mean, median, etc.)
#StdValue:  Standardized values; not available in all cases (USE THIS)
#Unit_1.UnitName:   Standard unit: Always available for standardized traits (USE THIS)
#RelUncertainty%:   Relative uncertainty in %
#UncertaintyName:   Kind of uncertainty (standard deviation, standard error,...)
#OrigObsDataID: indicates that that row is a duplicate, contains the ObsDataID of the original entry
#NonTraitCategories:    Type of ancillary data

#head(data)
```

---

2. CLEAN TRY TRAIT DATA

This chunk takes a while to run. The trait data file has a trait value for each row and a unique ObsDataID for each row. Clean by 1) isolating only the rows with standardized trait data, and 2) removing duplicate trait data. Export clean TRY trait data to DATA/DATA_SYNTHESIZED/tryDataT.txt

```
#1. Subset the rows containing trait data
dataT<-subset(data, TraitVsNonTrait=='x') #subset the rows containing trait data

#2. Isolate the rows with original ObsDataID (and loose the rows that reference an OrigObsDataID),
data1<-dataT
# Put these columns back into a format that is easier to manipulate
data1$OrigObsDataID<-as.numeric(data1$OrigObsDataID)
data1$ObsDataID<-as.numeric(data1$ObsDataID)
# Original row is OrigObsDataID == NA or....
sum(is.na(data1$OrigObsDataID))
```

```
## [1] 138371
```

```
# Original row is OrigObsDataID == ObsDataID
sum(data1$OrigObsDataID == data1$ObsDataID) # 0 rows
```

```
## [1] NA
```

```
# Isolate original rows
data2<-data1[is.na(data1$OrigObsDataID),]
totalrows<-dim(data2)[1]
totalrows
```

```
## [1] 138371
```

```
## Are the ObsDataIDs unique? No
uniques<-length(unique(data2$ObsDataID))
uniques
```

```
## [1] 136543
```

```
totalrows - uniques # 1,828 duplicates?
```

```
## [1] 1828
```

```
# Clean data and figure out where the duplicates are...
## Get rid of the rows where there is no standardized trait value (StdValue == NA)
sum(is.na(data2$StdValue)) # 38,740 rows without a standardized trait value
```

```
## [1] 38740
```

```
data3<-data2[!is.na(data2$StdValue),]
totalrows<-dim(data3)[1]
totalrows
```

```
## [1] 99631
```

```
## Are the ObsDataIDs unique now? No
uniques<-length(unique(data3$ObsDataID))
uniques
```

```
## [1] 97803
```

```
totalrows - uniques # 1,828 duplicates still.
```

```
## [1] 1828
```

```
## Make a list of rows that have a duplicated ObsDataID
ID<-unique(data3$ObsDataID)
dup.list<-numeric(0)
dup.num<-numeric(0)
i<-0
for (i in 1:length(ID)){ #loop through unique ID list
  temp<-data3$ObsDataID==ID[i] # for each ID, see how many rows there are
  if(sum(temp)>1){dup.list<-c(dup.list,ID[i])} #if there is more than one row, then store that ID num i
  if(sum(temp)>1){dup.num<-c(dup.num,sum(temp))} #if there is more than one row, then store the number
}
length(dup.list) # 1,828 duplicates
```

```
## [1] 1828
```

```
sum(dup.num==2) # all of them have just 2 rows (no triples)
```

```
## [1] 1828
```

```
## Check out a bunch of the duplicate rows
temp<-subset(data3, ObsDataID==dup.list[10]) # checked many duplicate rows and they are in fact duplica

# Remove duplicate rows
```

```
# beware this takes forever
data4<-data3
i<-0
for (i in 1:length(dup.list)){
  temp<-which(data4$ObsDataID==dup.list[i]) # which row number matches dup.list element?
  data4<-data4[-temp[2],] # delete the second element of 'which' from data
}
dataT<-data4
#head(dataT)
#dim(dataT)
#colnames(dataT)

# Export dataframe
newfilename<-'tryDataT.txt'
write.table(dataT, file=paste(synthdataPath,newfilename, sep='/'), sep='\t')
```

---

3. SUMMARIZE TRY TRAIT DATA

The dataset has a unique ObsDataID for each row.

```
### Pull out the unique trait IDs and names ##########################################################
dataT<-read.table('DATA/DATA_SYNTHESIZED/tryData/tryDataT.txt', sep='\t', header=T)

#list of traits in dataT
traitids<-unique(dataT$TraitID) # TraitID nums
#traitids
traitnames<-as.character(unique(dataT$TraitName)) #TraitName
#traitnames
#length(traitids)

#list of subtraits (dataID) in dataT
dataids<-unique(dataT$DataID)
#dataids
datanames<-as.character(unique(dataT$DataName))
#datanames
#length(dataids)

### Pull out the unique data IDs and names ###########################################################
#make an index of subtraits (dataID) that fall under each traitID
id.list<-list()
name.list<-list()
i<-0
for (i in 1:length(traitids)){
  temp<-dataT[dataT$TraitID==traitids[i],]
  id.list[[as.character(traitids[i])]]<-unique(temp$DataID)
  name.list[[as.character(traitnames[i])]]<-unique(temp$DataName)
}
#id.list
#name.list
```

```
#ldply(name.list, length) #distribution of dataids per traitid
#reorganize these lists so that can be merged into one table
new.id.list<-list()
new.name.list<-list()
i<-0
for(i in 1:length(id.list)){
  dataIDs<-id.list[[i]]
  n<-length(dataIDs)
  traitID<-names(id.list[i])
  new.id.list[[i]]<-data.frame(traitID = rep(traitID,n), dataID = dataIDs)

  dataNames<-name.list[[i]]
  traitName<-names(name.list[i])
  new.name.list[[i]]<-data.frame(traitName = rep(traitName,n), dataName = dataNames)
}
new.id.tab<-ldply(new.id.list)
new.name.tab<-ldply(new.name.list)

tdIndex<-data.frame(traitID=new.id.tab$traitID, traitName=new.name.tab$traitName,
           dataID=new.id.tab$dataID, dataName=new.name.tab$dataName)
#tdIndex


### Make a list for the corresponding Units ##########################################################
#dataids #unique list of dataids
ids.units<-character(0)
i<-0
for (i in 1:length(dataids)){
  temp<-dataT[dataT$DataID==dataids[i],]
  row<-unique(temp$Unit_1_UnitName)
  ids.units<-rbind(ids.units,as.character(row))
} # all 'rows' were 1 element long, meaning that only 1 unit type is used for each DataID
#ids.units
#link units to their corresponding dataID
unittab<-data.frame(dataID=dataids, unit=ids.units)
#ammend the tdIndex with units by indexing dataID in the unittab
tdIndex$unit<-rep(NA,dim(tdIndex)[1])
i<-0
for(i in 1:length(dataids)){
  approp.unit<-as.character(unittab[unittab$dataID==dataids[[i]],'unit'])
  tdIndex[tdIndex$dataID==dataids[[i]],'unit']<-approp.unit
}
#head(tdIndex)


### Make a list of the corresponding paper traits of interest ########################################
# cn, percN, littercn, litterpercN
#subset the DataNames with 'nitrogen'
dataName_nitrogen<-tdIndex[grep("nitrogen", tdIndex$dataName),c('dataName','unit')]
#then, remove DataNames with the following attributes
#dataName_nitrogen$dataName
rows<-c(grep("Root",dataName_nitrogen$dataName),
        grep("root",dataName_nitrogen$dataName),
```

```
        grep("Stem",dataName_nitrogen$dataName),
        grep("Bark",dataName_nitrogen$dataName),
        grep("Whole",dataName_nitrogen$dataName),
        grep("area",dataName_nitrogen$dataName),
        grep("Total", dataName_nitrogen$dataName),
        grep("organic", dataName_nitrogen$dataName))
dataName_n1<-dataName_nitrogen[-rows,]
#dataName_n1
#identify traitsOfInterest
tdIndex$traitOI<-rep(NA, dim(tdIndex)[1])
#tdIndex[tdIndex$dataName %in% dataName_n1$dataName,]
tdIndex[tdIndex$dataName %in% dataName_n1$dataName, 'traitOI'] <- traitOI<-c('percN',
                                                                             'cn',
                                                                             'litterpercN',
                                                                             NA,
                                                                             'littercn')

#head(tdIndex)
# Export dataframe
newfilename<-'tryData_traitKey.txt'
write.table(tdIndex, file=paste(synthdataPath,newfilename, sep='/'), sep='\t')
```

Extra. CLEAN TRY META-DATA

This code takes a long time to run. The non-trait data file has meta-data about the traits data collected. For example, there is information such as the soil type or the latitude and longitude of the data collected.

```
# #1. Subset the rows containing non-trait data
# dataNT<-subset(data, !TraitVsNonTrait=='x')
#
# #2. What are the OrigObsDataIDs like? .. all NA
# d1<-dataNT
# ## Put these columns back into a format that is easier to manipulate
# d1$OrigObsDataID<-as.numeric(d1$OrigObsDataID)
# d1$ObsDataID<-as.numeric(d1$ObsDataID)
# dim(d1)[1] # total number of rows
# sum(is.na(d1$OrigObsDataID)) # all OrigObsDataID == NA
#
# # Isolate original rows
# d2<-d1[is.na(d1$OrigObsDataID),]
# totalrows<-dim(d2)[1]
# totalrows
# ## Are the ObsDataIDs unique? No
# uniques<-length(unique(d2$ObsDataID))
# uniques
# totalrows - uniques # 19,297 duplicates?
# ## Make a list of rows that have a duplicated ObsDataID
# #this took forever to run... be careful.
# ID<-unique(d2$ObsDataID)
```

```
# length(ID) #934,289
# dup.list<-numeric(0)
# dup.num<-numeric(0)
# i<-0
# #dup.list 57931:979343
# for (i in 1:length(ID)){ #loop through unique ID list
#   temp<-d2$ObsDataID==ID[i] # for each ID, see how many rows there are
#   print(i)
#   print(sum(temp))
#   if(sum(temp)>1){dup.list<-c(dup.list,ID[i])} #if there is more than one row, then store that ID num
#   if(sum(temp)>1){dup.num<-c(dup.num,sum(temp))} #if there is more than one row, then store the numbe
# }
# length(dup.list) # 1,828 duplicates
# sum(dup.num==2) # all of them have just 2 rows (no triples)
# ## Check out a bunch of the duplicate rows
# temp<-subset(d2, ObsDataID==dup.list[100]) # checked many duplicate rows and they are in fact duplica
# temp
#
# # Remove duplicate rows
# d3<-d2
# i<-0
# for (i in 1:length(dup.list)){
#   temp<-which(d3$ObsDataID==dup.list[i]) # which row number matches dup.list element?
#   d3<-d3[-temp[2],] # delete the second element of 'which' from data
# }
# dataNT<-d3
#
# # head(dataNT)
# # Export dataframe
# newfilename<-'tryDataNT.txt'
# write.table(dataNT, file=paste(synthdataPath,newfilename, sep='/'), sep='\t')
```