Isotope Mixing Model Instructor Key

Hannah M Carroll and Derek D Houston

July 16, 2019

* This exercise will walk you through creating an isotope mixing model from the data we worked with in the food web exercise.
* Set your working directory by choosing Session from the menu at the top, then Set Working Directory -> To Source File Location

NOTE: WHEN YOU ARE DONE WITH THIS EXERCISE, KNIT THIS FILE TO WORD AND SUBMIT VIA BLACKBOARD

Your name:Jacqueline Galang

Now we’ll install and load the packages we need using a simple function

# This function will accept a list of packages, check to see whether they're installed, install any missing ones, and then load all packges in the list into memory.  
  
pack <- function(pkg){  
 new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])] # This line of the function defines "new packages" as those that are not already installed  
 if (length(new.pkg)) # If any of the packages we need are "new"...  
 install.packages(new.pkg, dependencies = TRUE) # Install them and any packages they depend on to work  
 sapply(pkg, library, character.only = TRUE) # And then load them all into memory so we can use them  
}  
  
# Our list of required packages  
packages <- c("rjags", "dplyr", "ggplot2", "simmr")  
  
# Run the function on our list of packages  
pack(packages)

## Loading required package: coda

## Linked to JAGS 4.3.0

## Loaded modules: basemod,bugs

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

## Loading required package: R2jags

##   
## Attaching package: 'R2jags'

## The following object is masked from 'package:coda':  
##   
## traceplot

## $rjags  
## [1] "rjags" "coda" "stats" "graphics" "grDevices" "utils"   
## [7] "datasets" "methods" "base"   
##   
## $dplyr  
## [1] "dplyr" "rjags" "coda" "stats" "graphics"   
## [6] "grDevices" "utils" "datasets" "methods" "base"   
##   
## $ggplot2  
## [1] "ggplot2" "dplyr" "rjags" "coda" "stats"   
## [6] "graphics" "grDevices" "utils" "datasets" "methods"   
## [11] "base"   
##   
## $simmr  
## [1] "simmr" "R2jags" "ggplot2" "dplyr" "rjags"   
## [6] "coda" "stats" "graphics" "grDevices" "utils"   
## [11] "datasets" "methods" "base"

#################################################################################################  
### Note: If you do not already have dplyr installed, this will take a long time. Be patient. ###  
#################################################################################################

Load the iso.data.clean2 dataset that we created in the Aquatic Food Web Exercise if it isn’t already in your Global Environment. If it is in your Global Environment already, you can skip this step.

iso.data.clean2 <- read.csv("isodataclean2.csv", header=TRUE)

We are using the package simmr to create an isotope mixing model of food sources for both the trout (apex predator) and leech. We assumed that the leech was parasitic, but the food web analysis we completed indicated that we probably have a free-living, nonparasitic leech instead.

The mixing model will evaluate the diet contributions and either confirm or refute our *a priori* trophic level assignment.

The first set of steps creates the model for *trout*.

* The simmr package requires that we extract parts of our dataset in a very precise way. Follow these steps carefully to ensure you create the correct model.
* You can view help files for any package by typing help(packagename)

# View the help file for the simmr package:  
help("simmr")

## starting httpd help server ... done

* We will need to subset our data frame called iso.data.clean2 in several separate steps.
* You can open the data frame in spreadsheet form to view it easily.
* Leave it open in another tab as you work so that you can refer back.
* You can also view objects by clicking the small white icon to the right of the object name in the Global Environment.

View(iso.data.clean2)  
  
# (1pt) View the dataset. What is the ID and trophic level of the organism on row 17 of the data frame?  
# The ID is trout and the trophic level is apex

Notes on the simmr package

* Simmr requires that elements of your data are extracted separately and turned into new matrix objects. We will create each of those required matrices in a series of steps.
* These elements are: – All data pertaining to the individual or group of individuals whose diet you are modeling (the Mix) – The names of the potential food sources of the Mix (in our case, these are the species of aquatic organisms that we found) – The isotopes of C and N of the potential food sources of the Mix – The standard deviation of the C and N isotopes of the potential food sources of the Mix – The carbon and nitrogen percentages of the potential food sources of the Mix

In simmr, the *mix* is the organism or organisms for which we are examining diet. We’ll start by looking at the diet of our trout.

# You can quickly call the row(s) or column(s) in a data frame by indexing like this: df[row, column], where df is the name of your dataframe. In our case, this is iso.data.clean2.  
  
# The line of code below can be read in sentence form as follows:  
 # Create an object called trout.mix. Subset iso.data.clean2 to include only row 17 (the row containing the trout data), and columns 5 through 6 (d13C and d15N).  
  
# The colon in between column (or row) numbers is read as "through." If we wanted columns 2 through 8, we would write 2:8.  
  
trout.mix <- iso.data.clean2[17, 5:6] # Note the comma in between row (17) and the columns (5 through 6).  
  
# Key things to remember when indexing:  
 # The square brackets are always used for indexing, and always come after the name of your data frame or matrix, with no space in between  
 # Leaving a blank either before or after the comma means you want to include all rows/columns  
  
#(1pt) On the line below, call row 12, and all columns of the iso.data.clean2 dataset, without assigning it to an object. Remember, the form is df[row, column]  
iso.data.clean2[12, 1:8]

## Sample Identifier ID Trophic corrected.d13C..VPDB.  
## 12 S14 Algae2 Algae Primary Producer -22.58175  
## corrected.d15N..Air. C.... N....  
## 12 8.620807 0.2168661 0.0275247

Now we will turn our trout.mix data frame into a matrix that simmr can use

# Both data frames and matrices are two-dimensional tables. However, data frames can store multiple types of data (numbers, characters, factors), while a matrix must contain all data of the same type.  
  
# Let's create the matrix of trout data using as.matrix  
trout.mix2 <- as.matrix(trout.mix)

Now we’ll set column names of our trout.mix2 matrix so that simmr knows which column is which

# Set the column names with the function colnames  
  
# Note that the c on the right side of the arrow means "combine" and is used to tell R you are giving it multiple pieces of information.  
colnames(trout.mix2) <- c("d13C", "d15N")  
  
# (1pt) trout.mix2 is a small matrix. Call it by writing the name of the matrix on the line below and running it to look at the data.  
trout.mix2

## d13C d15N  
## 17 -27.47983 11.33372

Next, we’ll create a matrix of data for the trout’s potential food sources.

The package magrittr gives us the ability to use pipes to efficiently perform operations. Pipes are the symbols %>%. They link many commands together to process complex data or perform multiple operations in the same step. The code below combines the data processing abilities of the dplyr package with the pipes from magrittr.

# The d13C and d15N of the trout's potential food sources need to be averaged by type (mayflies, stoneflies, algae, etc.). That information is stored in the ID column of our data frame, iso.data.clean2.  
  
# Since we are now looking at potential food sources, we have to REMOVE the trout from this part of the data. We can use column indexing like we did above, but with a minus sign to indicate that it should be left out.  
  
# The first line of code is creating an object called source.means, to which we assign the output. We start by dropping the trout row (row 17), and keeping the ID column (3), d13C column (5), and d15N column (6) for the rest of the rows of iso.data.clean2.  
# Remember, "c" means "combine" and is used to tell R that we want multiple rows or columns that are not adjacent to each other in the data frame.  
  
source.means <- iso.data.clean2[-17, c(3,5:6)] %>%  
 group\_by(ID) %>% # This line then groups the data by ID  
 summarize\_all(.funs = c(mean="mean")) # This line then calculates the mean d15C and d15N by the groups we created above. The summarize\_all function will calculate any number of summary statistcs for you. The command "funs" will accept a list of one or more functions. Writing mean="mean" is asking for the mean to be calulcated, and to have the word "mean" appended to the column name so that we know what it contains.

# Some of our IDs have only one observation. We can't create a meaningful average for those, so we have to remove them. We have only one observation each for Amphipods (row 2), Caddisflies (row 3), Leeches (row 4), and Wood (row 8). We will ignore those, and just take rows 1 and 5 through 7.  
# We also need to drop the ID column (column 1) that we had used for grouping  
source.means2 <- source.means[c(1,5:7), -1]  
  
# (2pts) On the line below, write an explanation in your own words of what the code on the line above is doing:  
#It is combining to to tell r that there is multiple pieces of information not adjecent calling for rows 1 and 5-7 and removing column 1 and assigning that data to source.means2

# (1pt) Now we will convert the source.means2 data frame into a matrix. Use the command as.matrix() on the line below to create a new object called source.means3 from source.means2 and convert it to a matrix. Remember that you can refer to the trout.mix2 code above for a reminder of the correct form.  
source.means3<- as.matrix(source.means2)

(1pt) Our data need to be in numeric form for analysis. Look in your Global Environment. Are the data in numeric (num) or character (chr) format? num

# (1pt) Check that the conversion to matrix format worked by asking R for the class of the source.means3 object on the line below. Remember, the command for this is class(objectname).  
class(source.means3)

## [1] "matrix"

We now need to get the names (ID) of the trout’s potential food sources. This is stored in column 1 of source.means.

# The source names object needs to be a vector for simmr to read it. A vector is a one-dimensional array which holds only one type of data (numeric, character, factor, etc.). Ours will be a character vector containing the IDs of the trout's food sources.  
  
# We will start by subsetting the source.means object to get just the rows and column we need, dropping those for which we have only one observation  
source.names <- source.means[c(1,5:7), 1]  
  
# (2pts) On the line below, write an explanation of what the code on the line above is doing:  
# the line of code is combining to have multiple pieces of information calling from source.means for rows 1 and 5-7, and column 1 assigning that data to the object source names

# Now we can ask for just the four IDs we need, dropping any row information (remember that this must be one-dimensional). We convert this to a vector with the as.vector command.  
source.names2 <- as.vector(source.names$ID)  
  
  
# (1pt) Call the source.names2 object by writing the name of the object and executing the code to make sure we have the correct names.   
source.names2

## [1] "Algae" "Mayflies" "Midges" "Stoneflies"

Now we will get the standard deviation of the d15C and d15N values for the trout’s potential food sources

# We can wrap the whole summary step in the as.matrix command to make it more efficient.   
  
# Remember, the rest of the code is exactly what we did to get source means, except we are asking the summarize\_all function to calculate standard deviation (sd) instead of mean. Refer back to the source.means step if you need a refresher on reading this code.  
source.sds <- as.matrix(iso.data.clean2[-17, c(3, 5:6)] %>%  
 group\_by(ID) %>%  
 summarize\_all(.funs = c(sd="sd")))  
  
# (2pts) On the line below, write an explanation in your own words of what the first line of code starting with source.sds <- is doing. Remember that you can refer to the means step for help on any of these lines.  
#The first line is assigning an output and creating a matrix  
# (2pts) On the line below, write an explanation in your own words of what the group\_by step is doing.  
# This step is grouping the data by its ID  
# (2pts) On the line below, write an explanation in your own words of what the summarize\_all step is doing.  
# The summarize\_all is to get summary statistics and the .funs is to accept a funtion. sd="sd" is calculating the standard deviation and assigning that value to sd and appending sd to the column name so we know it is the standard deviation.

The next step is to drop the ID column from our matrix.

source.sds2 <- source.sds[c(1, 5:7), 2:3]  
  
# (2pts) On the line below, write an explanation in your own words of what the code on the line above is doing. Remember, this is the same thing we did for source.means.  
#The code is saying that there is multiple pieces of information that are not adjacent calling for rows 1 and 5-7, and columns 2-3 (which is removing the id column) and assigning that data to source.sds2.

The *apply* function will apply a function that you specify over the margins (rows or columns) of a matrix or vector.

# A matrix can only contain one data type (character, numeric, factor, etc.). Because we converted our data frame to a matrix without removing the ID column first (character), it recoded all of our variables to character. We now need to convert the data type of both variables to numeric.  
  
source.sds3 <- apply(source.sds2, 2, as.numeric) # This line is applying the as.numeric function to the columns (1 for rows, or 2 for columns) of the source.sds2 matrix.  
  
# (2pts) What is the class of the source.sds3 object? Why do we need it to be this class? Remember that you can find the class using the command class(objectname). The class of the object is a matrix. It needs to be a matrix because that is the form that simmr reads and it has multiple rows and columns.  
class(source.sds3)

## [1] "matrix"

Simmr lets us include the % carbon and nitrogen of the potential food sources’ tissues in our analyses as the *concentration*. This helps improve the robustness of our diet estimates.

We will use the colummns for the carbon and nitrogen percentages of the trout’s potential food sources as the concentration values.

# This code is exactly what we used for the source.sds step, except that we are using columns 7 through 8 (carbon and nitrogen percentage) and calculating the mean. Remember, we are converting this to a matrix in the same step as doing the summary because it's more efficient.  
conc <- as.matrix(iso.data.clean2[-17, c(3, 7:8)] %>%  
 group\_by(ID) %>%  
 summarize\_all(.funs = c(mean="mean")))

# We are dropping the ID column and the organisms for which we have only one observation  
conc2 <- conc[c(1, 5:7), 2:3]

# Now we convert the data type to numeric so that Simmr can read it correctly  
conc3 <- apply(conc2, 2, as.numeric)   
  
# (1pt) There are many ways to check the structure of your data. This is handy for making sure that datasets have imported or converted correctly, or for quickly checking the arrangement of rows and columns. The command head(objectname) displays the column names and the first few rows of data. Use this command on conc3. What left to right order are the carbon and nitrogen columns in?  
#carbon on the left and nirtogen on the right  
head(conc3)

## C....\_mean N....\_mean  
## [1,] 0.2018419 0.02673105  
## [2,] 0.4783827 0.09792757  
## [3,] 0.4357503 0.08749803  
## [4,] 0.4883785 0.10385011

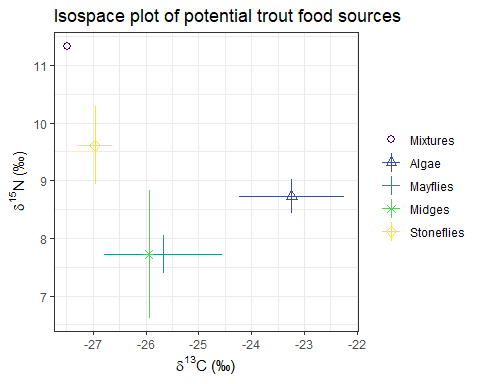
Now we have all of the data we need to create the mixing model. The simmr\_load command puts everything into the correct format for the mixing model.

trout.simmr.in = simmr\_load(mixtures=trout.mix2,  
 source\_names=source.names2,  
 source\_means=source.means3,  
 source\_sds=source.sds3,  
 concentration\_means = conc3)  
  
# (1pt) What is the class of trout.simmr.in? Write the command to find the class on the line below and execute the code.  
class(trout.simmr.in)

## [1] "simmr\_input"

Let’s make a plot of the isospace of our aquatic macroinvertebrate community

# You can use expressions to include symbols in plots. Delta can be called by name in an expression, and ^ indicates that what comes next should be superscript. Note the placement of the quotation marks - these are key for getting the labels to work.  
plot(trout.simmr.in, xlab=expression(paste(delta^13, "C (\u2030)",sep="")), # \u2030 is the unicode for the permil symbol  
 ylab=expression(paste(delta^15, "N (\u2030)",sep="")),   
 title="Isospace plot of potential trout food sources")



# Note: The point or points labeled "Mixtures" are the isotopic signatures of the organism for which you are modeling diet. The Mixture in our case is the trout.

# The command simmr\_mcmc creates an isotope mixing model using Markov chain Monte Carlo.  
trout.simmr.out = simmr\_mcmc(trout.simmr.in)

## Only 1 mixture value, performing a simmr solo run...

## module glm loaded

## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 2  
## Unobserved stochastic nodes: 6  
## Total graph size: 111  
##   
## Initializing model

We now need to make sure the model has run correctly. We can get diagnostics using the summary command.

summary(trout.simmr.out,type='diagnostics')

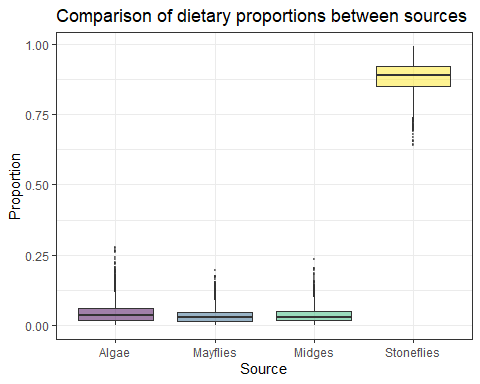
##   
## Summary for 1   
## Gelman diagnostics - these values should all be close to 1.  
## If not, try a longer run of simmr\_mcmc.  
## deviance Algae Mayflies Midges Stoneflies sd[d13C]   
## 1 1 1 1 1 1   
## sd[d15N]   
## 1

# (1pt) Did our model run correctly? What evidence do you see?  
#all the values of the diagnostics were 1

We now have everything we need to estimate the food sources of the trout

# This command creates a box and whisker plot of the potential food sources  
compare\_sources(trout.simmr.out)

## Most popular orderings are as follows:  
## Probability  
## Stoneflies > Algae > Midges > Mayflies 0.2133  
## Stoneflies > Algae > Mayflies > Midges 0.2031  
## Stoneflies > Midges > Algae > Mayflies 0.1614  
## Stoneflies > Midges > Mayflies > Algae 0.1475  
## Stoneflies > Mayflies > Algae > Midges 0.1467  
## Stoneflies > Mayflies > Midges > Algae 0.1281



# (1pt) What is the trout's main food source?  
# Stoneflies are the trouts main food source.  
# (1pt) Look in the console at the bottom of your screen. What is the probability that the trout's diet consists of Stoneflies > Algae > Midges > Mayflies?  
#0.2072

* You can reuse all the code we wrote above, changing the column indexes and object names to estimate diet for the leech instead of the trout
* Don’t forget that you can use View(iso.data.clean2) (capital V!) at any time to see your dataset as a spreadsheet.

(25 points total)

(2pts) Create the mix object for the leech. Don’t forget to insert a code chunk and write your R code inside of it.

leech.mix <- iso.data.clean2[1, 5:6]  
class(leech.mix)

## [1] "data.frame"

(1pt) Set the column names for the leech inside of a code chunk.

leech.mix2 <- as.matrix(leech.mix)  
  
colnames(leech.mix2) <- c("d13C", "d15N")  
class(leech.mix2)

## [1] "matrix"

(2pts) Create the source means object. Remember that you will need to do this in three steps and end up with numeric values in a matrix. Each of these steps needs to be in a code chunk. Refer back to the source means step for the trout for a reminder of what to write.

source.means <- iso.data.clean2[c(3:4,6,8:16), c(3,5:6)] %>%  
 group\_by(ID) %>%  
 summarize\_all(.funs = c(mean="mean"))  
source.means2 <- source.means[, -1]  
source.means3<- as.matrix(source.means2)

(2pts) Create the source names object from the ID column. Be sure to do this inside a code chunk.

source.names <- source.means[, 1]  
source.names2 <- as.vector(source.names$ID)

(2pts) Create the source sds object inside a code chunk.

source.sds <- as.matrix(iso.data.clean2[c(3:4,6,8:16), c(3,5:6)] %>%  
 group\_by(ID) %>%  
 summarize\_all(.funs = c(sd="sd")))  
  
source.sds2 <- source.sds[,2:3]  
  
source.sds3 <- apply(source.sds2, 2, as.numeric)

(2pts) Create the concentration object from the carbon and nitrogen percentages inside a code chunk.

conc <- iso.data.clean2[c(3:4,6,8:16), c(3, 7:8)] %>%  
 group\_by(ID) %>%  
 summarize\_all(.funs = c(mean="mean"))  
conc2 <- conc[, 2:3]  
conc3 <- apply(conc2, 2, as.numeric)   
class(conc3)

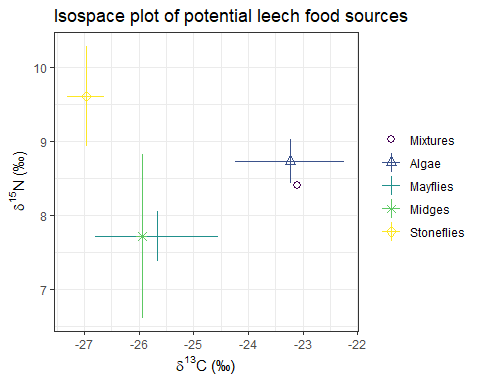
## [1] "matrix"

(2pts) Create the simmr\_in object in a code chunk. *Call it leech.simmr.in*

leech.simmr.in = simmr\_load(mixtures=leech.mix2,  
 source\_names=source.names2,  
 source\_means=source.means2,  
 source\_sds=source.sds3,  
 concentration\_means = conc3)

(1pt) Create the isospace plot of leech.simmr.in within a code chunk.

plot(leech.simmr.in, xlab=expression(paste(delta^13, "C (\u2030)",sep="")), # \u2030 is the unicode for the permil symbol  
 ylab=expression(paste(delta^15, "N (\u2030)",sep="")),   
 title="Isospace plot of potential leech food sources")

 (2pts) Create leech.simmr.out using the simmr\_mcmc command in a code chunk.

leech.simmr.out = simmr\_mcmc(leech.simmr.in)

## Only 1 mixture value, performing a simmr solo run...  
## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 2  
## Unobserved stochastic nodes: 6  
## Total graph size: 111  
##   
## Initializing model

(2pts) Get diagnostics on leech.simmr.out using the summary command in a code chunk.

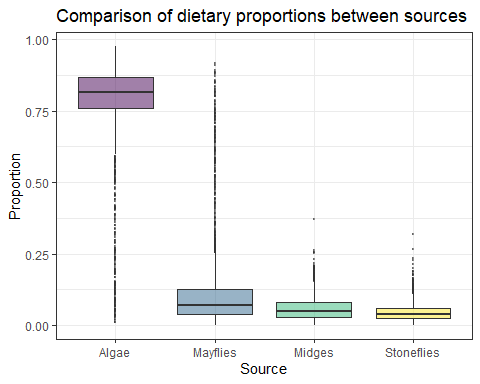
summary(leech.simmr.out,type='diagnostics')

##   
## Summary for 1   
## Gelman diagnostics - these values should all be close to 1.  
## If not, try a longer run of simmr\_mcmc.  
## deviance Algae Mayflies Midges Stoneflies sd[d13C]   
## 1.00 1.01 1.00 1.00 1.00 1.00   
## sd[d15N]   
## 1.00

(2pts) Produce the box and whisker plot comparing the sources in the leech’s diet within a code chunk.

compare\_sources(leech.simmr.out)

## Most popular orderings are as follows:  
## Probability  
## Algae > Mayflies > Midges > Stoneflies 0.2711  
## Algae > Mayflies > Stoneflies > Midges 0.2367  
## Algae > Midges > Mayflies > Stoneflies 0.1739  
## Algae > Midges > Stoneflies > Mayflies 0.1350  
## Algae > Stoneflies > Midges > Mayflies 0.0700  
## Algae > Stoneflies > Mayflies > Midges 0.0678  
## Mayflies > Algae > Midges > Stoneflies 0.0153  
## Mayflies > Algae > Stoneflies > Midges 0.0142  
## Mayflies > Stoneflies > Algae > Midges 0.0067  
## Mayflies > Midges > Algae > Stoneflies 0.0044  
## Mayflies > Midges > Stoneflies > Algae 0.0033  
## Mayflies > Stoneflies > Midges > Algae 0.0017



(5pts) We assumed that this leech was parasitic when we initially collected it. Were those assumptions correct? What makes up the greatest proportion of the leech’s diet? Is it free-living or parasitic? The initial assummption was incorrect. The leech is free living and its diet is made up primarily from algae. —

When you are done with this exercise, knit the file to a Word document and submit via Blackboard!