dietr Tutorial

Samuel R. Borstein

12 December, 2021

1: Introduction

This is a tutorial for using the R package `dietr`. `dietr` uses diet or food item data to calculate trophic levels following the procedures described in `TrophLab` (Pauly et al., 2000), which currently is only avaialble as a Microsoft Access database program. Our implementation is very easy to use and extremely fast as users can specify all their data as dataframes. It also differs from TrophLab in that users can specify a taxonomic hierarchy and measure trophic levels from their data at various levels (e.x. individual,population, species,genus, etc.). `dietr` also works well with FishBase (Froese & Pauly, 2018) data and can use diet and food item data obtained in R using the rfishbase package (Boettiger et al., 2012). In addition to estimating trophic levels, `dietr` can also estimate various electivity indices used in diet studies.

For this tutorial we will refer to diet data as quantitative stomach content data where the proportion of prey items are known (e.x. percent volume or weight of prey items, be cautious using percent frequency as various literature suggest it may be misleading). In this case, trophic level is simply estimated by adding one to the sum of trophic levels of the prey items consumed weighted by their contribution to the diet. The trophic level of consumer i($Troph_i$) is defined by the equation below:

$$Troph_i = 1 + \sum_{j=1}^{G} DC_{ij} \times Troph_j$$

Here $Troph_j$ is the trophic level of the jth prey item consumed in the diet of i, DC_{ij} is the fraction of prey item j in the diet of i, and G is the number of prey species in the diet.

As estimates of prey trophic levels may not be exact, the standard error around the estimate of the trophic level can be calculated with equation 2. TrophLab gets around this by assigning a standard error for each trophic level and using the below equation to measure the standard error (which they also refer to as the omnivory index). Here, the standard error of the focal species i, $s.e._i$, is calculated as the square root of the sum of the product of the standard errors of the prey items $s.e._j$ to the second weighted by their respective contribution in the diet, DC_{ij} over 100.

$$s.e_i = \sqrt{\frac{\sum_{j=1}^{G} DC_{ij} * s.e._j^2}{100}}$$

For estimating trophic levels from food items found in the diet that don't have proportions, a random sampling and ranking of the food items is used to get an estimate of the trophic level. The simulated proportion of prey items for calculating trophic level, `P` is calculated using the following equation:

$$log_{10}P = 2 - 1.9log_{10}R - 0.16log_{10}G$$

Here, `R` is the rank of the food item and `G` is the number of food items, up to 10. If more than 10 food items are listed, then a subsample of ten are randomly selected. The trophic level is then calculated using the following equation:

$$Troph = \sum(P_i * Troph_i) / \sum P_i$$

Here, P_i is the simulated proportion of the prey item i in the diet and $Troph_i$ is the trophic level of prey item i. This procedure is repeated n times and the mean of these n simulations is taken as the trophic level. In cases where only a single prey item is in the diet, the procedure is much simpler and the estimated trophic level is simply calculated by adding 1 to the trophic level of the single prey item.

The estimate of the standard error around the trophic level from food item data is defined below. Here, the standard error is the square root of the sum of the product of the standard error of a prey item squared and the contribution of the prey item minus 1 divided by the sum of the contribution of the prey items, P, minus the total number of prey items, G. For food item data, the random sampling routine and calculation to estimate trophic level and standard error is repeated 100 times, with the final trophic level and standard error being the mean of these 100 calculations.

$$s.e_i = \sqrt{\frac{(s.e_1)^2 * (P_1 - 1) + (s.e._2)^2 * (P_2 - 1) ... (s.e_G)^2 * (P_G - 1)}{\sum P - G}}$$

In this tutorial we will cover the basics of how to use `dietr` to measure trophic levels. We will first discuss how to read in data from FishBase using rfishbase and how to pass that data into `dietr`. We will then show how one can input their own raw data and use the package to estimate trophic levels.

2: Installation

2.1: Installation From CRAN

In order to install the stable CRAN version of the `dietr` package:

```
install.packages("dietr")
```

2.2: Installation of Development Version From GitHub

While we recommend use of the stable CRAN version of this package, we recommend using the package `devtools` to temporarily install the development version of the package from GitHub if for any reason you wish to use it:

```
#1. Install 'devtools' if you do not already have it installed:
install.packages("devtools")

#2. Load the 'devtools' package and temporarily install the development version of
#'dietr' from GitHub:
library(devtools)
dev_mode(on=T)
install_github("sborstein/dietr")  # install the package from GitHub
library(dietr)# load the package

#3. Leave developers mode after using the development version of 'dietr' so it will not
#remain on your system permanently.
dev_mode(on=F)
```

3: Using dietr

To load `dietr` and all of its functions/data:

```
library(dietr)
```

3.0: Basic Data Necessary to Run dietr for Trophic level Calculations

To run `dietr` you will need the following data as inputs. First is diet or food item data, which our functions call in DietTroph and FoodTroph as `DietItems` and `FoodItems` respectively. These should be organized from most inclusive to least inclusive from left to right and have column names.

The second is a dataframe of trophic levels of the prey, or as we name them in the functions, `PreyValues`. We include a few different datasets with prey values users can use, though you can also supply your own. `FishBasePreyVals` are the values FishBase/TrophLab use to calculate trophic levels. `CortesPreyVals` are standardized diet prey values for sharks from Cortes, 1999. Both of these can be loaded into R from `dietr` using the `data()` function:

```
data(FishBasePreyVals)#Load the Fishbase trophic levels of prey items
#Standardized trophic levels of prey items for elasmobranchs
data(CortesPreyVals)#Load the Cortes (1992)
```

The last data object we need is a data frame we call `Taxonomy`. Columns of this dataframe should move from least inclusive to most inclusive from left to right. This data is used to assign individuals to groups for measuring hierarchical trophic levels (ex. trophic levels for an individuals, populations, species, etc.). This can be as simple as just a single column data frame if you only want to measure trophic levels for each individual and not at a higher level.

3.1: Using dietr to Calculate Trophic Levels From FishBase Diet Data

Our first example will use FishBase diet data. Unfortunately, this was easier to do with previous versions of `rfishbase` where you could specify the taxa you wanted data for from the `diet function` and this function had to be re-written to work with `rfishbase 3.0`. In version 3.0 and above, the `rfishbase` seperated the function into two different functions that are not intuitively useful as one returns the actual diet data (`diet_items`) and the other returns the metadata for the diet records (`diet`). Unfortunately, one must manually merge these records together to see what species belong to which diet data and uses can no longer specify a single taxon to retrieve data for. `dietr`'s `ConvertFishBaseDiet` function does the merging of these two data sets for you and allows you to exclude life history stages while converting the data into a format that can be used to calculate trophic levels in `dietr`. However, as `rfishbase` no longer allows you to specify species to return, the function downloads all available diet data, thus, users will need to filter out their focal taxa.

The function has only one argument, `ExcludeStage`, in which users specify if they want to exclude a stage (ex. larvae) or not (in which case `ExcludeStage = NULL`). It returns a list of two data frames. The first, named `DietItems` contains the diet items while the second, `Taxonomy` , contains the taxonomy for calculating heirarchical trophic levels. Below, we will get FishBase diet for use with `dietr` while removind records from immature specimens.

```
# Convert Fishbase Diet Data and exclude juvenile and larval records
my.diets <- ConvertFishbaseDiet(ExcludeStage=c("recruits/juv.","larvae"))
```

We can see that `my.diets` object we created is a list containing two data frames. The first one, called DietItems, contains the information about diet composition, with `FoodI`, `FoodII`,`FoodIII`, `Stage` and `DietPercent` being the fields with descriptors for the various diets. In this case, each diet item has its own row in the data frame. The first column, `Individual`, contains the fish base diet reference number unique to that study. By including this, one can have multiple studies of the same species and then pool the data using the Taxonomy, which is the second data frame in our list. The `Species` column in the `DietItems` data frame contains the species name. For example, from the `my.diets$DietItems` object we created above, we can see that the first record in our dataset *Merlangius merlangus* with the diet record number of 4 ate eight different items, mollusks, squids/cuttlefish, bony fish, n.a./other annelids, and a variety of other invertebrates. If we look at `my.diets$Taxonomy` we will see it is a data frame of two columns. For FishBase data, our function returns each individual diet study for a species in the column `Individual`. The next column has the respective species name. This information will then be used to calculate the trophic levels for each individual diet record and then for all records belonging to that species.

We can now measure the trophic level from diet items using the function `DietTroph`. Here, we will specify our `DietItems` and `Taxonomy` using the `cleaned.diets` object we generated above. We will also specify the `PreyValues` as being the included `FishBasePreyVals` that are part of the package. As the columns for the `PreyVals` and `DietItems` use a calssification of "FoodI","FoodII","FoodIII","Stage", we will specify these as a vector for the `PreyClass` argument. For this example, lets just focus on a single species with multiple diet records, *Epinephelus itjara*, which has a total of three records.

```
#Remove Data for Epinephelus itajara
my.diets$DietItems <- my.diets$DietItems[my.diets$DietItems$Species ==
  "Epinephelus itajara",]
my.diets$Taxonomy <- my.diets$Taxonomy[my.diets$Taxonomy$Species ==
  "Epinephelus itajara",]
```

We can now calculate the trophic levels. We will use the trophic levels of prey from FishBase, which is a data object that can be loaded named `FishBasePreyVals`. We will specify our diet data and taxonomy with the parameter `DietItems` and `Taxonomy` respectively. We also need to specify how the prey is classified. This is meant to be flexible and tells `dietr` how to relate the values in `DietItems` to the values in `PreyValues`. If we look at the columns of both, we can see they use a classification scheme of `FoodI`, `FoodII`, "FoodIIIandStage`, so we will input that information in`PreyClass`. We will show the flexibility of this parameter a little later in this vignette in another example.The final parameter,`SumCheck`checks that the diet data do infact sum to 100 as would be expected if they are percent composition and will recalculate if they are not. I strongly recommend this always be set to`TRUE‘.

```
data(FishBasePreyVals)#load the FishBase prey values that are part of the dietr package
#Calculate trophic level with DietTroph function
my.TL<-DietTroph(DietItems = my.diets$DietItems,PreyValues = FishBasePreyVals,
Taxonomy = my.diets$Taxonomy, PreyClass=c("FoodI","FoodII","FoodIII","Stage"),
  SumCheck = TRUE)
```

We can see that the `my.TL` object we just created contains a list of length two, each with a data frame, with the names of these data frames matching the `Taxonomy` we provided. We can see that the first data frame in this list, named `Individual` contains the trophic levels for each individual study that we input, while the data frame named `Species` provides the mean trophic level and SE and the number of studies across all individuals for each species.

3.2: Using dietr to Calculate Trophic Levels from FishBase Food Item Data

Our second example will estimate trophic level from food item data. The process is extremely similar to the above. For this example, let's get some data from rfishbase using the `fooditems` function for the same two species we did above.

```
#Get some food item data from rfishbase
my.food<-rfishbase::fooditems(c("Lutjanus apodus","Epinephelus itajara"))
```

In order to use this in `dietr`, we will need to convert it using the function `ConvertFishbaseFood`. This function is basically identical to the `ConvertFishbaseDiet` function we used above, except our data frame is from the `fooditems` function. We will then use `FoodTroph` function to calculate the trophic level. It is important to note that as this method randomly samples food items and ranks them for calculating the trophic level, that each time you run the function you will get a slightly different result, though they should largey be close in value.

```
#Convert FishBase food item data to a format usable for FoodTroph
converted.foods<-ConvertFishbaseFood(my.food)
#Calculate trophic level from food items
my.TL<-FoodTroph(FoodItems = converted.foods$FoodItems,PreyValues = FishBasePreyVals,
  Taxonomy = converted.foods$Taxonomy,PreyClass=c("FoodI","FoodII","FoodIII","Stage"),
  Iter = 100, SE.Type = "TrophLab")
```

3.3: Using dietr to Calculate Trophic Levels From Your Own Data

`dietr` was written with flexability in mind so it is easy for users to use their own data or data from non-FishBase sources to calculate trophic levels. In this section, we will discuss how we can use this flexibility to customize trophic level calculations for a dataset. For this example, we will use data from Magalhaes et al., 2015, which analyzed the stomach contents of the cichlid *Herichthys minckleyi* volumetrically. Do note that this example is likely more complex than how it would be to use `dietr` on your own data given the data from Magalhaes et al., 2015 is not formatted in a way that is super compatible with `dietr`. A lot of this code in the example for formatting the data for use with `dietr` may not be necessary if your data is collected and formatted in a way that is similar to what is used by `dietr`.

First we can load the data. We will mostly be focusing on the last ten columns, which contain the volumetric proportions of the prey items as well as the first three columns which contain information on the individual fish, the lake it was caught, and the year in which it was caught.

```
data(Herichthys)
```

Note, this is the raw data from their paper and contains other data on morphology and genotypes. As such, not all individuals have diet data associated with them, so we will want to remove them.

```
#Subset out individuals with diet data
HMdat<-Herichthys[Herichthys$total==100,]
```

For this tutorial lets measure four hierarchical trophic levels. Specifically, we will measure trophic levels at the individual, lake by year, lake (across all years), and for the species inclusive of all *Herichthys minckleyi* individuals. To do this, we will need to generate a four column data frame to input as our `Taxonomy` parameter. We can do this by doing the following.

```
#Make a data frame of the individuals, lake by year, and lake.
HMtax<-cbind.data.frame(HMdat$individual,paste(HMdat$lake,HMdat$year),HMdat$lake)
#Name the data frame
colnames(HMtax)<-c("Individual","Lake x Year","Lake (all years)")
#To calculate trophic level for the entire species, add a vecotr to the data frame of
#the species name
HMtax$species<-"Herichthys minckleyi"
```

Next, we need to organize the data for input with dietr. First, lets grab out the data.

```
HMdat<-HMdat[,c("individual","X.Gastrop","X.Insect","X.Fish","X.Zoopl","X.plants",
            "X.algae", "X.detritus")]
```

Remember that `dietr` requires each row to be a unique diet item per individual and that the first column contains the individual's name. We will need to format our data to work. Currently, we can see that for each individual, there are columns for the prey items, so we will need to take the data in these columns and make each prey item a row.

```
#Repeat the individual name the number of unique prey types (6)
Inds<-rep(x = HMdat$individual, times=6)[order(rep(x = HMdat$individual, times=6))]
```

```r
#Repeat the number of food typed the length of the number of individuals
FoodTypes<-rep(x = colnames(HMdat[2:7]),times=length(unique(HMtax$Individual)))
#Make a data frame, the length of the individuals with three columns
HM.mat<-as.data.frame(matrix(nrow = length(Inds),ncol = 3))
#Name these columns
colnames(HM.mat)<-c("Individual","FoodItem","Percent")
#Populate the dataframes first column with the individual and the second column with
#the prey type
HM.mat$Individual<-Inds
HM.mat$FoodItem<-FoodTypes
#Run this for loop to find the diet data based on the individual and then match the
#diet percentage based on the name of the prey type
for(i in 1:nrow(HMdat)){
  rows2match<-which(HM.mat$Individual==HMdat$individual[i])
  HM.mat$Percent[rows2match]<-as.vector(as.numeric(HMdat[i,2:7]))
}
#Remove prey that do not contribute to diets
HM.mat<-HM.mat[!HM.mat$Percent==0,]
```

We can see that our data frame `HM.mat` is of three columns. The first row has our individual names, the second, the prey name, and the third, the dietary contribution of that prey. We now need to generate the prey values we want to use for calculating trophic levels into a format that can be used with `dietr`. For our example, we can easily do this by creating a data frame, which we will call PreyMat. The dimensions of this data frame will be three columns, so we can input the prey name, prey trophic level, and prey SE (if we want to have one, we can also just set them to 0, and functionally not measure it). Note for this example, the values of the prey will be equivalent to those in `FishBasePreyVals`, but as an example, I will show a simple way one could make prey values with vectors.

```r
#Create a empty data frame for prey values
PreyMat<-as.data.frame(matrix(ncol = 3,nrow = 6))
#Name the columns something useful
colnames(PreyMat)<-c("FoodItem","TL","SE")
#Add in the prey name to the PreyMat
PreyMat[,1]<-unique(FoodTypes)
#Add in the trophic levels of the prey
PreyMat[,2]<-c(2.37,2.2,3.5,2.1,2,2)
#Add in the SE of the prey
PreyMat[,3]<-c(.58,.4,.8,.3,0,0)
```

We now have all our needed information to calculate trophic levels (`DietItems`, `Taxonomy`, and `PreyValues`). We can now call `dietr's DietTroph` function and calculate trophic levels at the individual, lake by year, lake, and species level we defined in our Taxonomy data frame.

```r
HM.TL<-DietTroph(DietItems = HM.mat,PreyValues = PreyMat, PreyClass = "FoodItem",
  Taxonomy = HMtax, SumCheck = TRUE)
```

We can see that the object returned `HM.TL` is a list that has a length of four (one for each of our specified levels in taxonomy). Each element of the list is a data frame containing the trophic level calculations at the respective hierarchical levels.

3.4: Using dietr to Calculate Electivity Indices

While `dietr` can estimate trophic levels from food item and diet composition data as highlighted above, it can also measure a number of popular electivity indices used in studies of trophic ecology. The `dietr` function `Electivity` implements Ivlev's (1961), Strauss' (1979), Jacob's Q and D (1974), Chesson's (1983)(Which is similar to Manl'y Alpha (1974)), and Vanderploeg & Scavia (1979) electivity indices. `Electivity` takes two data frames as input. One should contain the relative abundance of consumed prey (argument `Diet`) and the other should contain the relative abundance of available prey items in the habitat (argument `Available`. Not that abundance percent should be as a decimal (i.e. sum to 1, not 100). If users choose, they can upload input data that is not in units of percent abundance and use the `CalcAbundance` option to do so. The input data frames should be formatted as such. The data frame for `Diet` should have each individual row as a diet record. The first column should contain the name of the diet record, the second, a identifier that links that record to an identifier in the first row in the data frame for `Available` (i.e. a habitat name, year, etc.). All remaining columns should contain a unique prey item and its relative abundance in the diet. The `Available` data frame contains similar data, however, in this data frame, the first column contains the identifier for the available prey relative abundance and all remaining columns should represent prey items available in the habitat and match those in `Diet`. Note that the prey items in `Diet` should be the same as those in `Available` and vice versa.

To highlight an example of how to use this function and how data should be formatted for it, we can load data from Horn, 1982. This is a dataset containing data on relative abundance by percent weight of macroalgae prey consumption and availability for two species of Stichaeidae in two different years. We can load it as such and see the format of the data by doing the following.

```
data(Horn1982)# load data
Horn1982$Consumed#See data for prey consumption
Horn1982$Available#See data for available prey
```

We can see here that we are identifying available prey her by year and month in the first column of `Horn1982$Available` and that all remaining columns are prey items available during those two time periods. In `Horn1982$Consumed` we can see that the first column is the name of the record and the second column matches a record in the first column of `Horn1982$Available` to link it to prey availability. We can then run one or any of the indices using the following code and arguments. We will define the consumed prey with the `Diet` argument as `Horn1982$Consumed`, `Available` prey as `Horn1982$Available`, and we will denote we want to calculate all of the indices using the `Indices` argument. For calculating Jacobs' Q, we will use the argument `LogQ` which takes `log10` of Q, which is recommended. We will also denote that we do not want to account for prey depletion in the calculation of Chesson's index. If set to true, Chesson's case 2 will be calculated.

```
my.indices <- Electivity(Diet = Horn1982$Consumed, Available = Horn1982$Available,
  Indices = c("ForageRatio","Ivlev","Strauss","JacobsQ","JacobsD","Chesson",
  "VanderploegScavia"),LogQ = TRUE, CalcAbundance = FALSE, Depleting = FALSE)
```

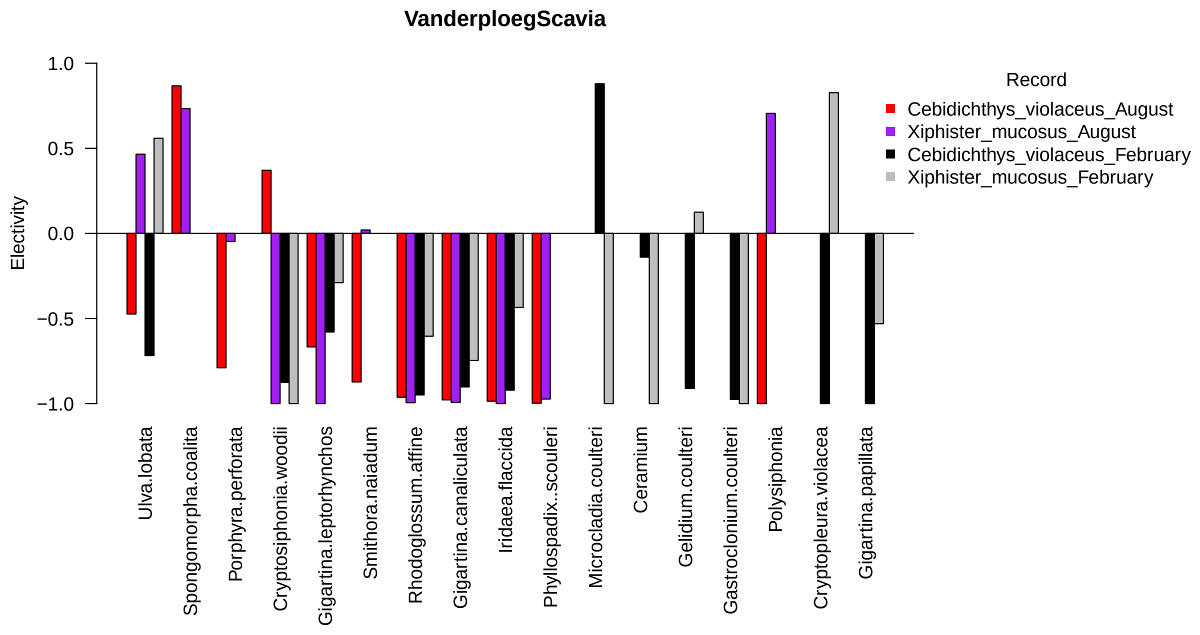
```
## Warning in Electivity(Diet = Horn1982$Consumed, Available =
## Horn1982$Available, : Some values are greater than 1, but you specified not
## to calculate relative abundance. You may get odd results for some indices. For
## relative abundance, data should sum to 1 and therefore cannot be greater than 1.
## If your data are relative abundance, you may need to divide them by 100 so they
## sum to 1. If they are not yet in relative abundance, please use CalcAbundance =
## TRUE

## Warning in lapply(X = x, FUN = .Generic, ...): NaNs produced

## Warning in lapply(X = x, FUN = .Generic, ...): NaNs produced

## Warning in lapply(X = x, FUN = .Generic, ...): NaNs produced

## Warning in lapply(X = x, FUN = .Generic, ...): NaNs produced

## Warning in lapply(X = x, FUN = .Generic, ...): NaNs produced

## Warning in lapply(X = x, FUN = .Generic, ...): NaNs produced
## Warning in lapply(X = x, FUN = .Generic, ...): NaNs produced
```

We can see that the `Electivity` function returned a list of data frames of class `Electivity`. Each data frame contains a calculated index and is named respective of the index calculated. Do not that in some cases of these calculations, certain calculations may yield `NaN` values or `Inf` values, for instance due to division by 0 or log of zero or negative number. These typically arise when a prey item is either presentin the habitat but not consumed, present in the diet but not in the habitat, or where they are both absent from a record. In these cases, `NA` is returned.

We can also use the function `PlotElcectivity` to plot our Electivity calculations. This function takes an object of class `Electivity` and a few other parameters to choose what indices are plotted as well as options for the plots themselves, such as font size and color. Electivity calculations are input using hte `Electivity.Calcs` parameter. Users use the parameter `indices` to choose which of Ivlev's, Strauss', Jacob's D, and Vanderploeg & Scavia indices they want plotted. If numerous indices are selected, numerous plots will be made. If users supply colors via the `BarColor` parameter, they must be the same length as the number of records calculated in Electivity and supplied via `Electivity.Calcs`. IF no colors are supplied, `dietr` will automatically assign colors. The parameters `NameSize`, `AxisFontSize`, `BorderCol`, `LegendFontSize`, and `LegendTitle` control the font size of the names of prey plotted, axis label size, color of bar plot border outlines, font size of the legend, and the text used in the title of the legend respectively.

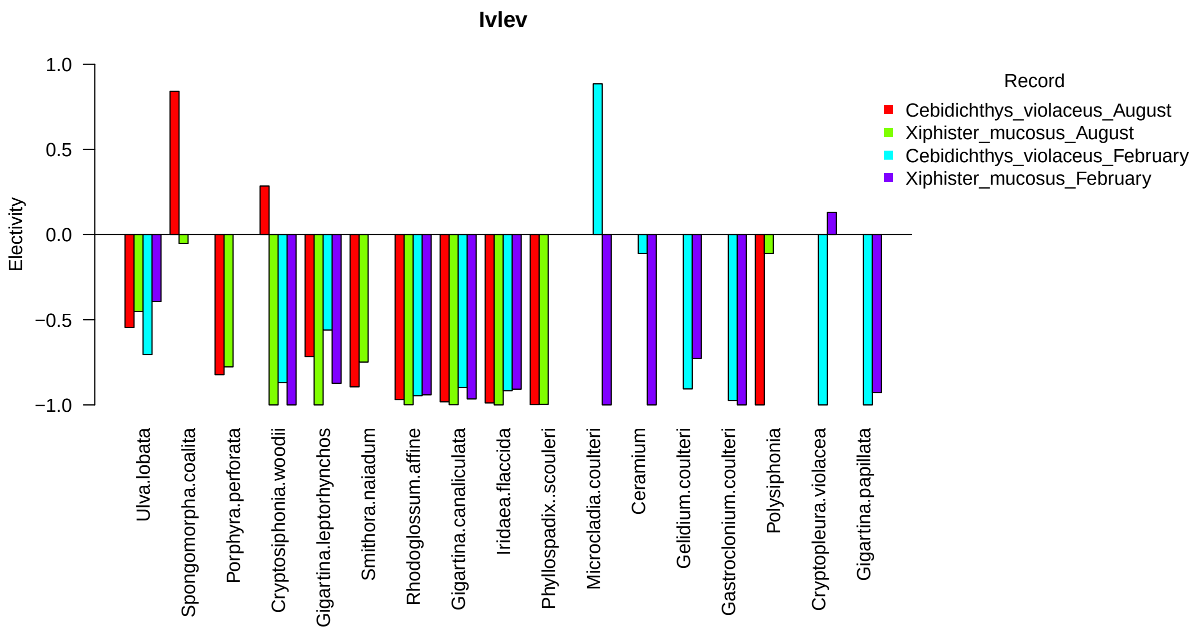
For instance, if we wanted to plot the Vanderploeg & Scavia index we calculated above, we could run the following. This will assign a seperate color using the `BarColor` argument.

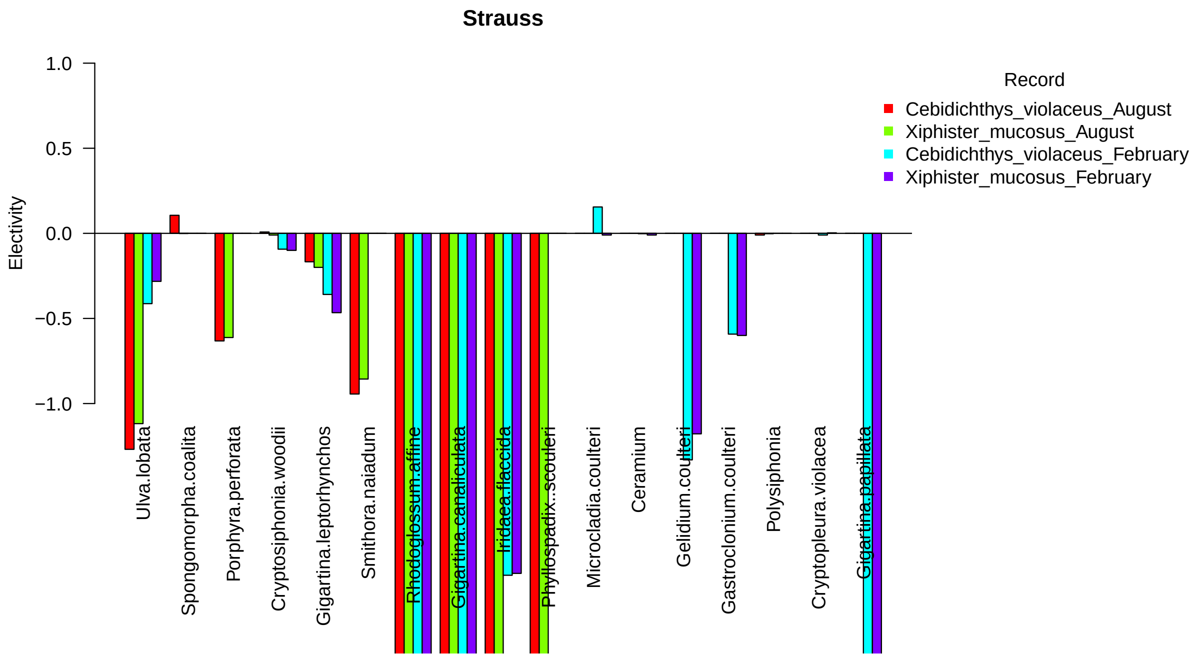
```
PlotElectivity(Electivity.Calcs = my.indices, Indices = "VanderploegScavia",
  BarColor = c("Red","Purple","Black","Grey"))
```

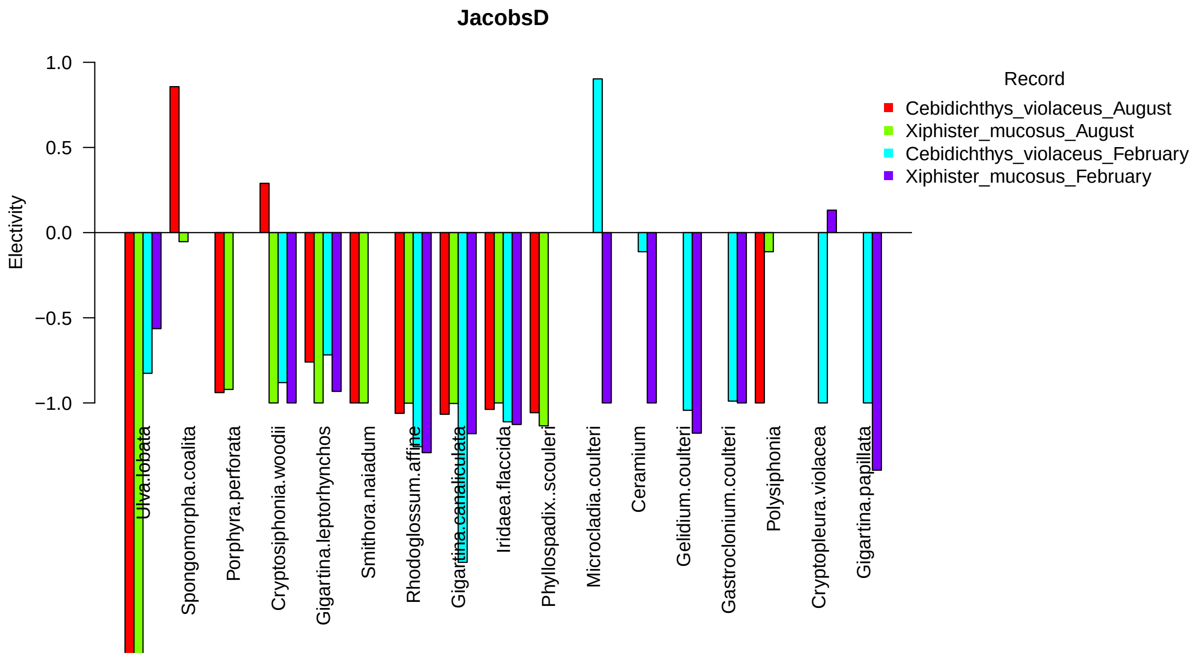


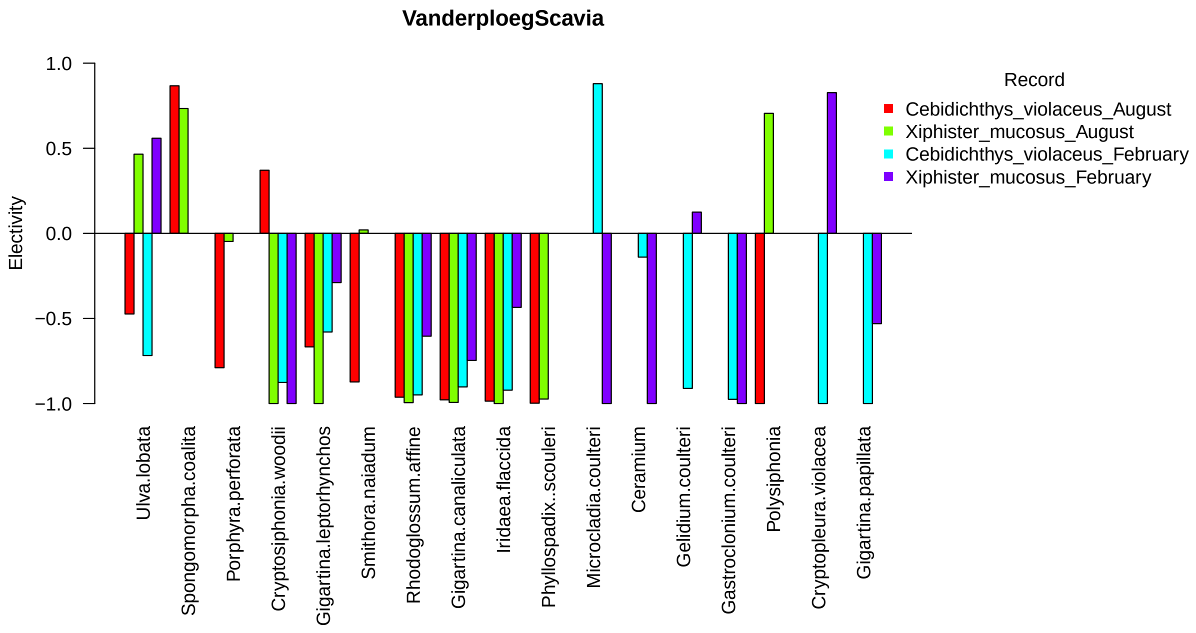
If we wanted to plot all four of the possible indices we can plot that were previously calculated, we could also run this chunk of code. For this example, I will not specify colors using `BarColor` to highlight the default color selection. Note that based on the size of your plotting window, you may need to adjust the size of the font.

```
PlotElectivity(Electivity.Calcs = my.indices)
```


Ivlev
Electivity
1.0
0.5
0.0
−0.5
−1.0
Record
Cebidichthys_violaceus_August
Xiphister_mucosus_August
Cebidichthys_violaceus_February
Xiphister_mucosus_February
Ulva.lobata
Spongomorpha.coalita
Porphyra.perforata
Cryptosiphonia.woodii
Gigartina.leptorhynchos
Smithora.naiadum
Rhodoglossum.affine
Gigartina.canaliculata
Iridaea.flaccida
Phyllospadix..scouleri
Microcladia.coulteri
Ceramium
Gelidium.coulteri
Gastroclonium.coulteri
Polysiphonia
Cryptopleura.violacea
Gigartina.papillata


Strauss
Electivity
1.0
0.5
0.0
−0.5
−1.0
Record
Cebidichthys_violaceus_August
Xiphister_mucosus_August
Cebidichthys_violaceus_February
Xiphister_mucosus_February
Ulva.lobata
Spongomorpha.coalita
Porphyra.perforata
Cryptosiphonia.woodii
Gigartina.leptorhynchos
Smithora.naiadum
Rhodoglossum.affine
Gigartina.canaliculata
Iridaea.flaccida
Phyllospadix..scouleri
Microcladia.coulteri
Ceramium
Gelidium.coulteri
Gastroclonium.coulteri
Polysiphonia
Cryptopleura.violacea
Gigartina.papillata





3.5: Using dietr to Calculate Composite Diet Indices

If users have diet data that is in percent frequency of occurrence, percent number, and percent volume or weight, `dietr` can be used to calculate three different composite indices. The first, the Index of Preponderance (Natarajan & Jhingran, 1961 AKA Feeding Index Kawakami & Vazzoler, 1980) is the product of the frequency of occurrence of prey with either their volumetric or weight contribution to the diet. The Index of Relative Importance (Pinkas et al., 1971) is calculated as the the product of the sum of the weight or volume of a prey item and the percent number and the percent frequency occurrence. The Feeding Quotient (Hureau, 1970) is the product of the percent number and the percent weight. For a review of these indices, I recommend de Silviera et al., 2020.

The formatting for the input data is rather minimal for this function. It is mandatory for this function that the first column contain the diet record identifier and the second column the names of the prey. The diet record identifier should be a unique name for each record, which allows one to calculate feeding indices for numerous records with a single call of the function. As the second column contains the prey identifier, if a species feeds upon three different prey, the first three rows of the dataset would have the same record identifier. The remaining columns should contain information on the percent frequency of occurrence, percent number, and percent weight or volume, though the order of these does not matter as they are assigned using the arguments `PercentOccurrence`, `PercentNumber`, and `PercentVolWeight` respectively. Users can specify which index/indices they want to calculate with the `Indices` argument, with the options being `IRI`, `IOP`, `FQ` for Index of Relative Importance, Index of Preponderance, and Feeding Quotient respectively. Users have the option of returning the full calculations and the percents or just the percents with the `PercentOnly` argument. If set to `TRUE` only the percent of these indices is returned. Additionally, if users want the raw data returned as well, they can do so with the `ReternRaw` argument. The dataset `Casaux1998` contains data highlighting how data should be formatted for these functions as well as to run the examples.

```r
#Load diet data from Casaux1998, which contains diet for two populations
#and is listed in percent frequency, percent number, and percent mass.
data(Casaux1998)

#Calculate all three diet indices (IOP, IRI, FQ), return the raw data, and all calculations.
CompositeIndices(DietData = Casaux1998, Indices = c("IOP","IRI","FQ"), PercentNumber = 4,
  PercentOccurrence = 3, PercentVolWeight = 5, ReturnRaw = TRUE, PercentOnly = FALSE)
```

```
## $Harpagifer_antarcticus_PotterCove
##          Prey PercentNumber PercentOccurrence PercentVolWeight     IOP
## 1   Gammarids          86.5              98.4             95.9 9436.56
## 2     Isopods           2.0               8.9              0.3    2.67
## 3  Gastropods           5.8              23.6              0.5   11.80
## 4    Bivalves           0.3               1.6              0.0    0.00
## 5     Chitons           0.1               0.8              0.0    0.00
## 6 Polychaetes           0.7               4.9              1.7    8.33
## 7       Algae           4.6              30.9              1.6   49.44
##          %IOP      IRI          %IRI      FQ          %FQ
## 1 99.24028269 17948.16 9.796384e+01 8295.35 99.854948600
## 2  0.02807925    20.47 1.117284e-01    0.60  0.007222476
## 3  0.12409557   148.68 8.115185e-01    2.90  0.034908636
## 4  0.00000000     0.48 2.619914e-03    0.00  0.000000000
## 5  0.00000000     0.08 4.366524e-04    0.00  0.000000000
## 6  0.08760306    11.76 6.418790e-02    1.19  0.014324578
## 7  0.51993942   191.58 1.045673e+00    7.36  0.088595710
##
## $Harpagifer_antarcticus_HarmonyPoint
##                 Prey PercentNumber PercentOccurrence PercentVolWeight     IOP
## 1 Euphausia superba          88.7              95.5             95.9 9158.45
## 2         Gammarids          11.3              18.2              4.1   74.62
##         %IOP      IRI      %IRI      FQ        %FQ
## 1 99.1918181 17629.30 98.435028 8506.33 99.4582972
## 2  0.8081819   280.28  1.564972   46.33  0.5417028
```

```r
#Calculate all three diet indices and return only the percent of the index
CompositeIndices(DietData = Casaux1998, Indices = c("IOP","IRI","FQ"), PercentNumber = 4,
  PercentOccurrence = 3, PercentVolWeight = 5, ReturnRaw = FALSE, PercentOnly = TRUE)
```

```
## $Harpagifer_antarcticus_PotterCove
##          Prey        %IOP         %IRI          %FQ
## 1   Gammarids 99.24028269 9.796384e+01 99.854948600
## 2     Isopods  0.02807925 1.117284e-01  0.007222476
## 3  Gastropods  0.12409557 8.115185e-01  0.034908636
## 4    Bivalves  0.00000000 2.619914e-03  0.000000000
## 5     Chitons  0.00000000 4.366524e-04  0.000000000
## 6 Polychaetes  0.08760306 6.418790e-02  0.014324578
## 7       Algae  0.51993942 1.045673e+00  0.088595710
##
## $Harpagifer_antarcticus_HarmonyPoint
##                 Prey       %IOP      %IRI        %FQ
## 1 Euphausia superba 99.1918181 98.435028 99.4582972
## 2         Gammarids  0.8081819  1.564972  0.5417028
```

```r
#Calculate Feeding Quotient and return the raw data and the all calculations.
CompositeIndices(DietData = Casaux1998, Indices = "FQ", PercentNumber = 4,
  PercentVolWeight = 5,ReturnRaw = FALSE, PercentOnly = FALSE)
```

```
## $Harpagifer_antarcticus_PotterCove
##          Prey      FQ          %FQ
## 1   Gammarids 8295.35 99.854948600
## 2     Isopods    0.60  0.007222476
## 3  Gastropods    2.90  0.034908636
## 4    Bivalves    0.00  0.000000000
## 5     Chitons    0.00  0.000000000
## 6 Polychaetes    1.19  0.014324578
## 7       Algae    7.36  0.088595710
##
## $Harpagifer_antarcticus_HarmonyPoint
##                 Prey      FQ        %FQ
## 1 Euphausia superba 8506.33 99.4582972
## 2         Gammarids   46.33  0.5417028
```

As we can see, this function returns a list with the calculated composite indices for each record in the diet data supplied as well as how the different options for `PercentOnly` and `ReturnRaw` work.

3.5: Using dietr to Calculate the Gastro-somatic Index and Vacuity Index

`dietr` also has functions to calculate the gastro-somatic and Vacuity indices. The Vacuity index is simply the percentage frequency of fishes that lack stomach contents. The first column should contain the specimen identifier and the second column should contain the weight of the stomach contents. Below is a quick example showing how to use the `dietr` function `VacuityIndex` to calculate the vacuity index using the example dataset `SebastesStomachs`, which contains NOAA data on the stomach content weight and predator weight for *Sebastes flavidus* specimens. For now, we will ignore the `PredatorWeight` column which contains the weight of the specimens as we will use that later in the tutorial.

```r
data(SebastesStomachs)#load example data
VacuityIndex(StomachData = SebastesStomachs)
```

```
## [1] 18.18182
```

To gastro-somatic index is the percentage weight the stomach contents represents relative to the total weight of a specimen, which can provide information on the feeding condition of a fish. The function `GastrosomaticIndex` calculates this. Data should be formatted similar as described above for the `VacuityIndex`, but a third column containing the wight of the specimen is also needed. For an example of how this data should be formatted, see the dataset `SebastesStomachs`. Besides the argument `StomachData` which should be the data frame with the stomach content and predator weights, an additional argument `Calc.Vacuity` is available. If `Calc.Vacuity = TRUE`, then in addition to the gastro-somatic index, the vacuity index is also calculated for the same data. This function will return a list contianing the gastro-somatic index of each specimen, the mean gastro-somatic index of all specimens, and, if `Calc.Vacuity = TRUE`, the vacuity index. We could run the function as such using the `SebastesStomachs` dataset.

```
data(SebastesStomachs)#load example data
#Calculate the Gastro-somatic Index and Vacuity Index
GastrosomaticIndex (StomachData = SebastesStomachs, Calc.Vacuity = TRUE)
```

```
## $IndividualGSI
##  Sebastes_flavidus_1_8_11_11   Sebastes_flavidus_1_8_9_11
##                 0.0000000000                 0.0012410250
## Sebastes_flavidus_10_8_11_11  Sebastes_flavidus_2_8_11_11
##                 0.0221241702                 0.0010421057
##  Sebastes_flavidus_3_8_11_11  Sebastes_flavidus_4_8_11_11
##                 0.0288197699                 0.0002769944
##  Sebastes_flavidus_5_8_11_11  Sebastes_flavidus_6_8_11_11
##                 0.0001150405                 0.0012811259
##  Sebastes_flavidus_7_8_11_11  Sebastes_flavidus_8_8_11_11
##                 0.0009243458                 0.0048773364
##  Sebastes_flavidus_9_8_11_11
##                 0.0000000000
##
## $MeanGSI
## [1] 0.005518356
##
## $VacuityIndex
## [1] 18.18182
```

4: Final Comments

Further information on the functions and their usage can be found in the helpfiles `help(package=dietr)`. For any further issues and questions send an email with subject 'dietr support' to sam@borstein.com or post to the issues section on GitHub(https://github.com/sborstein/dietr/issues).

5: References

Boettiger C, Lang DT, and Wainwright PC. 2012. rfishbase: exploring, manipulating and visualizing FishBase data from R. Journal of Fish Biology 81:2030-2039.

Chesson, J. 1983. The estimation and analysis of preference and its relatioship to foraging models. Ecology 64:1297-1304.

Cortes E. 1999. Standardized diet compositions and trophic levels of sharks. ICES Journal of marine science 56:707-717. da Silveira EL, Semmar N, Cartes JE, Tuset VM, Lombarte A, Ballester ELC, and Vaz-dos-Santos AM. 2019. Methods for Trophic Ecology Assessment in Fishes: A Critical Review of Stomach Analyses. Reviews in Fisheries Science & Aquaculture 28:71-106. 10.1080/23308249.2019.1678013

Froese R, and Pauly D. 2019. FishBase. http://www.fishbase.org/2019).

Horn M, Murray S, and Edwards T. 1982. Dietary selectivity in the field and food preferences in the laboratory for two herbivorous fishes (Cebidichthys violaceus and Xiphister mucosus) from a temperate intertidal zone. Marine Biology 67:237-246.

Hureau J-C. 1970. Biologie comparee de quelques poissons antarctiques (Nototheniidae). Bulletin de l'Institut Oceanographique de Monaco 68:1-244.

Ivlev, U. 1961. Experimental ecology of the feeding of fish. Yale University Press, New Haven.

Jacobs, J. 1974. Quantitative measurement of food selection. Oecologia 14:413-417.

Kawakami E, and Vazzoler G. 1980. Metodo grafico e estimativa de indice alimentar aplicado no estudo de alimentacao de peixes. Boletim do Instituto Oceanografico 29:205-207.

Magalhaes IS, Ornelas-Garcia CP, Leal-Cardin M, Ramirez T, and Barluenga M. 2015. Untangling the evolutionary history of a highly polymorphic species: introgressive hybridization and high genetic structure in the desert cichlid fish Herichtys minckleyi. Mol Ecol 24:4505-4520. 10.1111/mec.13316.

Manly, B. 1974. A model for certain types of selection experiments. Biometrics 30:281-294.

Natarajan A, and Jhingran A. 1961. Index of preponderance-a method of grading the food elements in the stomach analysis of fishes. Indian Journal of Fisheries 8:54-59.

Pauly D, Froese R, Sa-a P, Palomares M, Christensen V, and Rius J. 2000. TrophLab manual. ICLARM, Manila, Philippines.

Pinkas L, Oliphant MS, and Iverson IL. 1971. Food Habits of Albacore, Bluefin Tuna, and Bonito In California Waters. Fish Bulletin 152:1-105.

Strauss, R. E. 1979. Reliability Estimates for Ivlev's Electivity Index, the Forage Ratio, and a Proposed Linear Index of Food Selection. Transactions of the American Fisheries Society 108:344-352.

Vanderploeg, H., and D. Scavia. 1979. Two electivity indices for feeding with special reference to zooplankton grazing. Journal of the Fisheries Board of Canada 36:362-365.