Assignment: Local (alpha) Diversity

Roy Z Moger-Reischer; Z620: Quantitative Biodiversity, Indiana University

22 January, 2017

OVERVIEW

In this exercise, we will explore aspects of local or site-specific diversity, also known as alpha (α) diversity. First we will quantify two of the fundamental components of (α) diversity: **richness** and **evenness**. From there, we will then discuss ways to integrate richness and evenness, which will include univariate metrics of diversity along with an investigation of the **species abundance distribution (SAD)**.

Directions:

1. Change "Student Name" on line 3 (above) with your name.
2. Complete as much of the exercise as possible during class; what you do not complete in class will need to be done on your own outside of class.
3. Use the handout as a guide; it contains a more complete description of data sets along with the proper scripting needed to carry out the exercise.
4. Be sure to **answer the questions** in this exercise document; they also correspond to the handout. Space for your answer is provided in this document and indicated by the ">" character. If you need a second paragraph be sure to start the first line with ">".
5. Before you leave the classroom, **push** this file to your GitHub repo.
6. For homework, follow the directions at the bottom of this file.
7. When you are done, **Knit** the text and code into a PDF file.
8. After Knitting, please submit the completed exercise by creating a **pull request** via GitHub. Your pull request should include this file `alpha_assignment.Rmd` and the PDF output of `Knitr` (`alpha_assignment.pdf`).

1) R SETUP

In the R code chunk below, please provide the code to: 1) Clear your R environment, 2) Print your current working directory, 3) Set your working directory to your `/Week2-Alpha` folder, and 4) Load the `vegan` R package (be sure to install if needed).

```
rm(list=ls())
getwd()
```

```
## [1] "C:/Users/rmoge/GitHub/QB2017_Moger-Reischer/Week2-Alpha"
```

```
#setwd("~/GitHub/QB2017_Moger-Reischer/Week2-Alpha")
#setwd("/Users/rzmogerr/GitHub/QB2017_Moger-Reischer/Week2-Alpha")
setwd("C:\\Users\\rmoge\\GitHub\\QB2017_Moger-Reischer\\Week2-Alpha")
#install.packages("vegan")
require(vegan)
```

```
## Loading required package: vegan

## Loading required package: permute

## Loading required package: lattice

## This is vegan 2.4-2
```

2) LOADING DATA

In the R code chunk below, do the following: 1) Load your dataset, and 2) Display the structure of the dataset (if the structure is long, use `max.level=0` to show just basic information).

```
data(BCI)
str(BCI) #okay, this is something new I learned, should make a note
```

```
## 'data.frame':    50 obs. of  225 variables:
##  $ Abarema.macradenia              : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Vachellia.melanoceras           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Acalypha.diversifolia           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Acalypha.macrostachya           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Adelia.triloba                  : int  0 0 0 3 1 0 0 0 5 0 ...
##  $ Aegiphila.panamensis            : int  0 0 0 0 1 0 1 0 0 1 ...
##  $ Alchornea.costaricensis         : int  2 1 2 18 3 2 0 2 2 2 ...
##  $ Alchornea.latifolia             : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ Alibertia.edulis                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Allophylus.psilospermus         : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ Alseis.blackiana                : int  25 26 18 23 16 14 18 14 16 14 ...
##  $ Amaioua.corymbosa               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Anacardium.excelsum             : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Andira.inermis                  : int  0 0 0 0 1 1 0 0 1 0 ...
##  $ Annona.spraguei                 : int  1 0 1 0 0 0 0 1 1 0 ...
##  $ Apeiba.glabra                   : int  13 12 6 3 4 10 5 4 5 5 ...
##  $ Apeiba.tibourbou                : int  2 0 1 1 0 0 0 1 0 0 ...
##  $ Aspidosperma.desmanthum         : int  0 0 0 1 1 1 0 0 0 1 ...
##  $ Astrocaryum.standleyanum        : int  0 2 1 5 6 2 2 0 2 1 ...
##  $ Astronium.graveolens            : int  6 0 1 3 0 1 2 2 0 0 ...
##  $ Attalea.butyracea               : int  0 1 0 0 0 1 1 0 0 0 ...
##  $ Banara.guianensis               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Beilschmiedia.pendula           : int  4 5 7 5 8 6 5 9 11 14 ...
##  $ Brosimum.alicastrum             : int  5 2 4 3 2 2 6 4 3 6 ...
##  $ Brosimum.guianense              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Calophyllum.longifolium         : int  0 2 0 2 1 2 2 2 2 0 ...
##  $ Casearia.aculeata               : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Casearia.arborea                : int  1 1 3 2 4 1 2 3 9 7 ...
##  $ Casearia.commersoniana          : int  0 0 1 0 1 0 0 0 1 0 ...
##  $ Casearia.guianensis             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Casearia.sylvestris             : int  2 1 0 0 0 3 1 0 1 1 ...
##  $ Cassipourea.guianensis          : int  2 0 1 1 3 4 4 0 2 1 ...
##  $ Cavanillesia.platanifolia       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Cecropia.insignis               : int  12 5 7 17 21 4 0 7 2 16 ...
##  $ Cecropia.obtusifolia            : int  0 0 0 0 1 0 0 2 0 2 ...
##  $ Cedrela.odorata                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Ceiba.pentandra                 : int  0 1 1 0 1 0 0 1 0 1 ...
##  $ Celtis.schippii                 : int  0 0 0 2 2 0 1 0 0 0 ...
##  $ Cespedesia.spathulata           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Chamguava.schippii              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Chimarrhis.parviflora           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Maclura.tinctoria               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Chrysochlamys.eclipes           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Chrysophyllum.argenteum         : int  4 1 2 2 6 2 3 2 4 2 ...
##  $ Chrysophyllum.cainito           : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ Coccoloba.coronata              : int  0 0 0 1 2 0 0 1 2 1 ...
```

```
##  $ Coccoloba.manzinellensis       : int  0 0 0 0 0 0 0 2 0 0 ...
##  $ Colubrina.glandulosa           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Cordia.alliodora               : int  2 3 3 7 1 1 2 0 0 2 ...
##  $ Cordia.bicolor                 : int  12 14 35 23 13 7 5 10 7 13 ...
##  $ Cordia.lasiocalyx              : int  8 6 6 11 7 6 6 3 0 4 ...
##  $ Coussarea.curvigemma           : int  0 0 0 1 0 2 1 0 1 1 ...
##  $ Croton.billbergianus           : int  2 2 0 11 6 0 0 4 2 0 ...
##  $ Cupania.cinerea                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Cupania.latifolia              : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Cupania.rufescens              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Cupania.seemannii              : int  2 2 1 0 3 0 1 2 2 0 ...
##  $ Dendropanax.arboreus           : int  0 3 6 0 5 2 1 6 1 3 ...
##  $ Desmopsis.panamensis           : int  0 0 4 0 0 0 0 0 0 1 ...
##  $ Diospyros.artanthifolia        : int  1 1 1 1 0 0 0 0 0 1 ...
##  $ Dipteryx.oleifera              : int  1 1 3 0 0 0 0 2 1 2 ...
##  $ Drypetes.standleyi             : int  2 1 2 0 0 0 0 0 0 0 ...
##  $ Elaeis.oleifera                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Enterolobium.schomburgkii      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Erythrina.costaricensis        : int  0 0 0 0 0 3 0 0 1 0 ...
##  $ Erythroxylum.macrophyllum      : int  0 1 0 0 0 0 0 1 1 1 ...
##  $ Eugenia.florida                : int  0 1 0 7 2 0 0 1 1 3 ...
##  $ Eugenia.galalonensis           : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Eugenia.nesiotica              : int  0 0 1 0 0 0 5 4 3 0 ...
##  $ Eugenia.oerstediana            : int  3 2 5 1 5 2 2 3 3 3 ...
##  $ Faramea.occidentalis           : int  14 36 39 39 22 16 38 41 33 42 ...
##  $ Ficus.colubrinae               : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ Ficus.costaricana              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Ficus.insipida                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Ficus.maxima                   : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ Ficus.obtusifolia              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Ficus.popenoei                 : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ Ficus.tonduzii                 : int  0 0 1 2 1 0 0 0 0 0 ...
##  $ Ficus.trigonata                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Ficus.yoponensis               : int  1 0 0 0 0 1 1 0 0 0 ...
##  $ Garcinia.intermedia            : int  0 1 1 3 2 1 2 2 1 0 ...
##  $ Garcinia.madruno               : int  4 0 0 0 1 0 0 0 0 1 ...
##  $ Genipa.americana               : int  0 0 1 0 0 0 1 0 1 1 ...
##  $ Guapira.myrtiflora             : int  3 1 0 1 1 7 3 1 1 1 ...
##  $ Guarea.fuzzy                   : int  1 1 0 1 3 0 0 2 0 3 ...
##  $ Guarea.grandifolia             : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Guarea.guidonia                : int  2 6 2 5 3 4 4 0 1 5 ...
##  $ Guatteria.dumetorum            : int  6 16 6 3 9 7 8 6 2 2 ...
##  $ Guazuma.ulmifolia              : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Guettarda.foliacea             : int  1 5 1 2 1 0 0 4 1 3 ...
##  $ Gustavia.superba               : int  10 5 0 1 3 1 8 4 4 4 ...
##  $ Hampea.appendiculata           : int  0 0 1 0 0 0 0 0 2 1 ...
##  $ Hasseltia.floribunda           : int  5 9 4 11 9 2 7 6 3 4 ...
##  $ Heisteria.acuminata            : int  0 0 0 0 1 1 0 0 0 0 ...
##  $ Heisteria.concinna             : int  4 5 4 6 4 8 2 5 1 5 ...
##  $ Hirtella.americana             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Hirtella.triandra              : int  21 14 5 4 6 6 7 14 8 7 ...
##  $ Hura.crepitans                 : int  0 0 0 0 0 2 1 1 0 0 ...
##  $ Hieronyma.alchorneoides        : int  0 2 0 0 0 0 0 0 1 0 ...
##   [list output truncated]
```

```
##  - attr(*, "original.names")= chr  "Abarema.macradenium" "Acacia.melanoceras" "Acalypha.diversifolia
```

```
dim(BCI)
```

```
## [1]  50 225
```

3) SPECIES RICHNESS

Species richness (S) is simply the number of species in a system or the number of species observed in a sample.

Observed Richness

In the R code chunk below, do the following:

1. Write a function called `S.obs` to calculate observed richness
2. Use your function to determine the number of species in `site1`, and
3. Compare the output of your function to the output of the `specnumber()` function in vegan.

```
S.obs <- function(x=""   ){
   rowSums(x>0) *1
}#for each row x, take the sum of columns for whcih x > 0
S.obs(BCI) #now creates a vector with 50 row sums---one sum for each row (site) in that data set.
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  93  84  90  94 101  85  82  88  90  94  87  84  93  98  93  93  93  89
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
## 109 100  99  91  99  95 105  91  99  85  86  97  77  88  86  92  83  92
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50
##  88  82  84  80 102  87  86  81  81  86 102  91  91  93
```

```
specnumber(BCI)
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  93  84  90  94 101  85  82  88  90  94  87  84  93  98  93  93  93  89
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
## 109 100  99  91  99  95 105  91  99  85  86  97  77  88  86  92  83  92
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50
##  88  82  84  80 102  87  86  81  81  86 102  91  91  93
```

Question 1: Does `specnumber()` from `vegan` return the same value for observed richness in `site1` as our function `S.obs`? What is the species richness of the first 4 sites (i.e., rows) of the BCI matrix?

> ***Answer 1***: yes, for both richness was 93

Coverage. How Well Did You Sample Your Site?

In the R code chunk below, do the following:

1. Write a function to calculate Good's Coverage, and
2. Use that function to calculate coverage for all sites in the BCI matrix.

```
#Good's coverage
C<-function(x=""){
  1- (rowSums(x==1)/rowSums(x))
```

```
}
C(BCI)
```

```
##         1         2         3         4         5         6         7
## 0.9308036 0.9287356 0.9200864 0.9468504 0.9287129 0.9174757 0.9326923
##         8         9        10        11        12        13        14
## 0.9443155 0.9095355 0.9275362 0.9152120 0.9071038 0.9242054 0.9132420
##        15        16        17        18        19        20        21
## 0.9350649 0.9267735 0.8950131 0.9193084 0.8891455 0.9114219 0.8946078
##        22        23        24        25        26        27        28
## 0.9066986 0.8705882 0.9030612 0.9095023 0.9115479 0.9088729 0.9198966
##        29        30        31        32        33        34        35
## 0.8983516 0.9221053 0.9382423 0.9411765 0.9220183 0.9239374 0.9267887
##        36        37        38        39        40        41        42
## 0.9186047 0.9379310 0.9306488 0.9268868 0.9386503 0.8880597 0.9299517
##        43        44        45        46        47        48        49
## 0.9140049 0.9168704 0.9234234 0.9348837 0.8847059 0.9228916 0.9086651
##        50
## 0.9143519
```

```
xC<-function(x=""){
  (rowSums(x==1))
}
C(BCI)
```

```
##         1         2         3         4         5         6         7
## 0.9308036 0.9287356 0.9200864 0.9468504 0.9287129 0.9174757 0.9326923
##         8         9        10        11        12        13        14
## 0.9443155 0.9095355 0.9275362 0.9152120 0.9071038 0.9242054 0.9132420
##        15        16        17        18        19        20        21
## 0.9350649 0.9267735 0.8950131 0.9193084 0.8891455 0.9114219 0.8946078
##        22        23        24        25        26        27        28
## 0.9066986 0.8705882 0.9030612 0.9095023 0.9115479 0.9088729 0.9198966
##        29        30        31        32        33        34        35
## 0.8983516 0.9221053 0.9382423 0.9411765 0.9220183 0.9239374 0.9267887
##        36        37        38        39        40        41        42
## 0.9186047 0.9379310 0.9306488 0.9268868 0.9386503 0.8880597 0.9299517
##        43        44        45        46        47        48        49
## 0.9140049 0.9168704 0.9234234 0.9348837 0.8847059 0.9228916 0.9086651
##        50
## 0.9143519
```

```
xC(BCI)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## 31 31 37 27 36 34 28 24 37 35 34 34 31 38 30 32 40 28 48 38 43 39 44 38 40
## 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## 36 38 31 37 37 26 27 34 34 44 35 27 31 31 30 45 29 35 34 34 28 49 32 39 37
```

```
yC<-function(x=""){
  (rowSums(x))
}
yC(BCI)
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## 448 435 463 508 505 412 416 431 409 483 401 366 409 438 462 437 381 347
```

```
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
## 433 429 408 418 340 392 442 407 417 387 364 475 421 459 436 447 601 430
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50
## 435 447 424 489 402 414 407 409 444 430 425 415 427 432
```

Question 2: Answer the following questions about coverage:

a. What is the range of values that can be generated by Good's Coverage?
b. What would we conclude from Good's Coverage if n_i equaled N?
c. What portion of taxa in `site1` were represented by singletons?
d. Make some observations about coverage at the BCI plots.

Answer 2a:
0 through 1. If every sp. is a singleton, then Good's Coverage is 0. If there are no singletons, Good's Coverage is 1.

Answer 2b:
In this case C=0. We would conclude that a lot more sampling needs to be done. We have only sampled singletons so far.

Answer 2c:
By running the xC function, I see that there are 31 singletons. I know there are 225 spp. Therefore $31/225 = 0.1377778$

Answer 2d:
The sites are well-sampled. Typically less than 10% of the spp. are represented by singletons. Site 23 has is near the low end of coverage (13% individuals are singletons); site 8 is near the best coverage (~6% of individuals are singeltons)

Estimated Richness

In the R code chunk below, do the following:

1. Load the microbial dataset (located in the `/Week2-Alpha/data` folder),
2. Transform and transpose the data as needed (see handout),
3. Create a vector (`soilbac1`) with the bacterial OTU abundances at any site in the dataset,
4. Calculate the observed richness at that particular site, and
5. Calculate the coverage at that particular site

```
soilbac <- read.table("data/soilbac.txt", sep = "\t", header = TRUE, row.names = 1)
soilbac.t <- as.data.frame(t(soilbac))#make a transposed version of the data
soilbac1<-s<-soilbac.t[1,]#look at a single site. I'm make an assumption that 3. above means "at a give
#can only use Chao estimators on abundance, not relative abundance, data

#For #4. above, we use the
soilbac1Richness<-S.obs(soilbac1)
soilbac1Coverage<-C(soilbac1)
yC(soilbac1)
```

```
## T1_1
## 2119
```

Question 3: Answer the following questions about the soil bacterial dataset.

a. How many sequences did we recover from the sample `soilbac1`, i.e. N?
b. What is the observed richness of `soilbac1`?

c. How does coverage compare between the BCI sample (`site1`) and the KBS sample (`soilbac1`)?

Answer 3a:
From yC function I see that there were 2119 individuals.

Answer 3b: Observed richness was 1074.

Answer 3c:
Coverage was lower in the KBS sample. There was a higher proportion of individuals that were singletons.

Richness Estimators

In the R code chunk below, do the following:

1. Write a function to calculate **Chao1**,
2. Write a function to calculate **Chao2**,
3. Write a function to calculate **ACE**, and
4. Use these functions to estimate richness at both `site1` and `soilbac1`.

```
S.chao1<- function(x=""){
  S.obs(x)+(sum(x==1)^2 / (2*sum(x==2)))
}

S.chao2<-function(site="",SbyS=""){
  SbyS=as.data.frame(SbyS)
  x=SbyS[site, ]
  SbyS.pa<-(SbyS>0)*1
  Q1=sum(colSums(SbyS.pa)==1)
  Q2=sum(colSums(SbyS.pa)==2)
  S.chao2 = S.obs(x) + (Q1^2)/(2 * Q2)
  return(S.chao2)
}
#go back through the handout and make sure you understand this

S.ace<-function(x="",thresh=""){
  x<-x[x>0]#exclude zero abundance
  S.abund<-length(which(x>thresh))
  S.rare <-length(which(x<=thresh))
  singlt <-length(which(x==1))
  N.rare <-sum(x[which(x<=thresh)])
  C.ace  <-1-(singlt/N.rare)
  i      <-c(1:thresh)
  count<-function(i,y){length(y[y==i])}
  a.1    <-sapply(i, count, x)
  f.1    <-(i*(i-1))*a.1
  G.ace  <-(S.rare/C.ace)*sum(f.1)/(N.rare*(N.rare-1))
  S.ace  <- S.abund + (S.rare/C.ace) + (singlt/C.ace)*max(G.ace,0)
  return(S.ace)
}

print(S.chao1(BCI[1,]))
```

```
##        1
```

```
## 119.6944
```

```
S.chao1(soilbac1)
```

```
##     T1_1
## 2628.514
```

```
S.chao2(1, BCI)
```

```
##        1
## 104.6053
```

```
S.chao2(1,soilbac.t)
```

```
##     T1_1
## 21055.39
```

```
S.ace(BCI[1,],10)
```

```
## [1] 159.3404
```

```
S.ace(soilbac1,10)
```

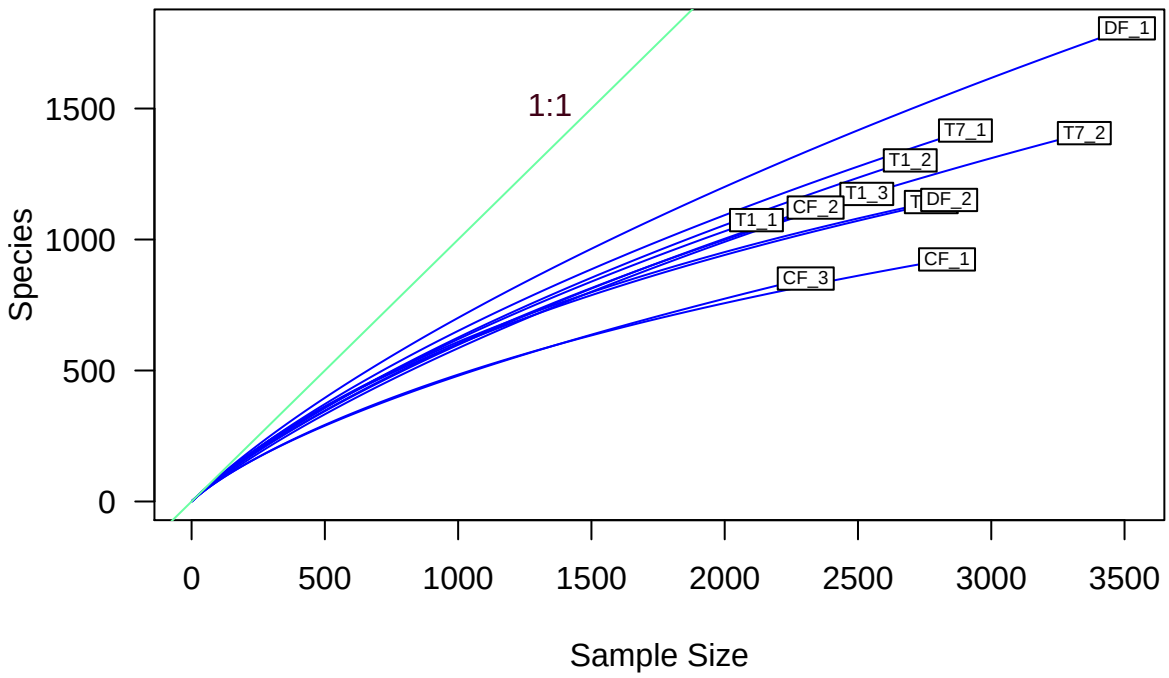
```
## [1] 4465.983
```

Rarefaction

In the R code chunk below, please do the following:

1. Calculate observed richness for all samples in `soilbac`,
2. Determine the size of the smallest sample,
3. Use the `rarefy()` function to rarefy each sample to this level,
4. Plot the rarefaction results, and
5. Add the 1:1 line and label.

```
soilbac.S <- S.obs(soilbac.t)#use vegan to calculate observed richness
min.N <- min(rowSums(soilbac.t))#which sample has the smallest sample size?
S.rarefy <- rarefy(x = soilbac.t, sample = min.N, se = TRUE)#use vegan to rarefy to the small sample si
rarecurve(x = soilbac.t, step = 20, col = "blue", cex = 0.6, las=1)#okay, using some arguments in vegan
abline(0, 1, col = '#6afea2')#make a 1:1 line
text(1500, 1500, "1:1", pos = 2, col = '#3f0016')#formatting its name
```



Question 4: What is the difference between ACE and the Chao estimators?

> ***Answer 4***: ACE allows the scientist to specify a maximum abundance for a site below which taxa are considered 'rare'. Based on my understanding of the equations I wrote above, I think that these rare taxa, in addition to singletons, can contribute to increasing the estimated richness. Chao estimators use number of singletons and number of doubletons in the calculations. Chao2 is also different in that it uses information from >1 site.

4) SPECIES EVENNESS

Here, we consider how abundance varies among species, that is, **species evenness**.

Visualizing Evenness: The Rank Abundance Curve (RAC)

One of the most common ways to visualize evenness is in a **rank-abundance curve** (sometime referred to as a rank-abundance distribution or Whittaker plot). An RAC can be constructed by ranking species from the most abundant to the least abundant without respect to species labels (and hence no worries about 'ties' in abundance).

In the R code chunk below, do the following:

1. Write a function to construct a RAC,
2. Be sure your function removes species that have zero abundances,
3. Order the vector (RAC) from greatest (most abundant) to least (least abundant), and
4. Return the ranked vector

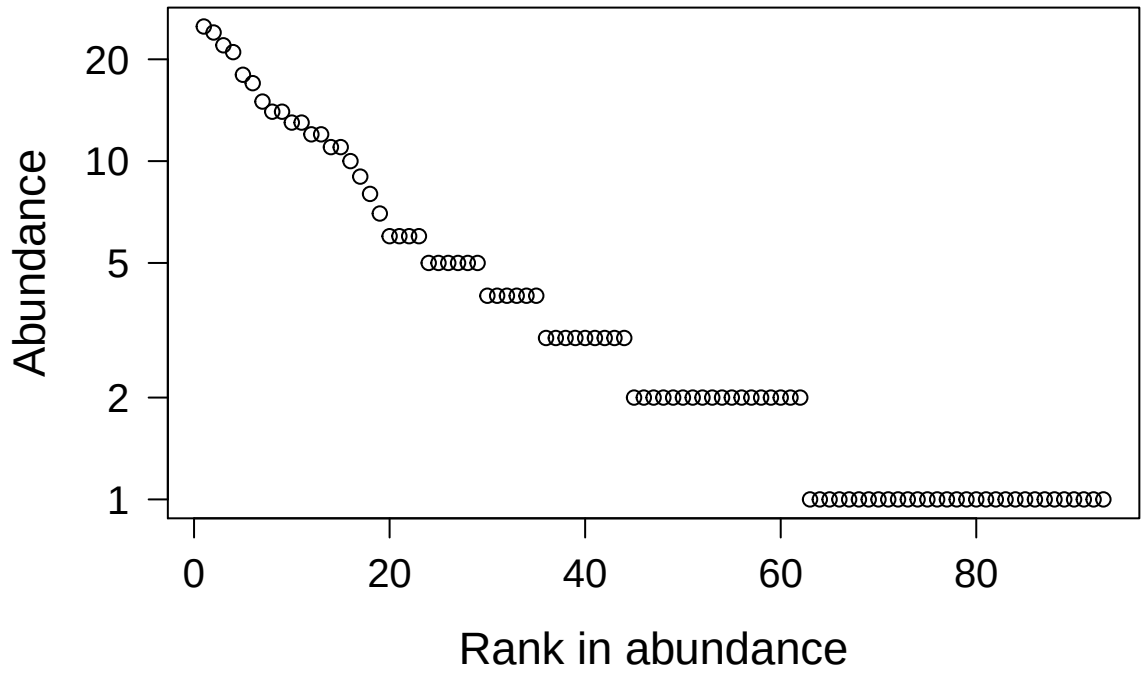
```
RAC <- function(x = ""){#begin RA curve function
x = as.vector(x)#force a varb type
x.ab = x[x > 0]#ignore spp for which no obsvns
x.ab.ranked = x.ab[order(x.ab, decreasing = TRUE)]#rank ordered
return(x.ab.ranked)#return the ranked vector so that it can be stored as an outside variable
}
```

Now, let's examine the RAC for `site1` of the BCI data set.

In the R code chunk below, do the following:

1. Create a sequence of ranks and plot the RAC with natural-log-transformed abundances,
2. Label the x-axis "Rank in abundance" and the y-axis "log(abundance)"

```
plot.new()#new plot?
site1 <- BCI[1, ]#choose ONE site (row)
rac <- RAC(x = site1)#make a RAC for that site
ranks <- as.vector(seq(1, length(rac)))#make a list (vector) of the ranks
opar <- par(no.readonly = TRUE)#save current par settings
par(mar = c(5.1, 5.1, 4.1, 2.1)) # New settings for par---this is how plots appear
plot(ranks, log(rac), type = 'p', axes = F, # Plots w/o axes
    xlab = "Rank in abundance", ylab = "Abundance",
    las = 1, cex.lab = 1.4, cex.axis = 1.25)#axis size
box() # Manually adds border
axis(side = 1, labels = T, cex.axis = 1.25) # Make an x-axis
axis(side = 2, las = 1, cex.axis = 1.25, # Make a log-Y axis
labels = c(1, 2, 5, 10, 20), at = log(c(1, 2, 5, 10, 20)))
```



```
par <- opar#reset the settings for displaying a plot
```

Question 5: What effect does visualizing species abundance data on a log-scaled axis have on how we interpret evenness in the RAC?

> ***Answer 5***: -We talked about the distributions that SADs might follow. Two of those were i) log-series and ii) lognormal. If we do this transformation and the resulting RAC appears linear, this could support a hypothesis of a log-series distribution. ##
> Now that we have visualized unevennes, it is time to quantify it using Simpson's evenness ($E_{1/D}$) and Smith and Wilson's evenness index (E_{var}).

Simpson's evenness ($E_{1/D}$)

In the R code chunk below, do the following:

1. Write the function to calculate $E_{1/D}$, and
2. Calculate $E_{1/D}$ for `site1`.

```
SimpE <- function(x = ""){
S <- S.obs(x)#obsvd richness
x = as.data.frame(x)
D <- diversity(x, "inv")
E <- (D)/S
return(E)
}

E1Dsite1<-SimpE(site1)
print (E1Dsite1)
```

```
##         1
## 0.4238232
```

```
#What are max/min values of E1/D? A: 0 through 1
```

Smith and Wilson's evenness index (E_{var})

In the R code chunk below, please do the following:

1. Write the function to calculate E_{var},
2. Calculate E_{var} for `site1`, and
3. Compare $E_{1/D}$ and E_{var}.

```
Evar <- function(x){#define
x <- as.vector(x[x > 0])#take nonzero values of a vector
1 - (2/pi)*atan(var(log(x)))#look at variance of the vector, then standarize it
}
EvarSite1<-Evar(site1)
print (EvarSite1)
```

```
## [1] 0.5067211
```

```
print (EvarSite1 - E1Dsite1)#Evar indicates higher evenness. This difference is valid bc both estimator
```

```
##          1
## 0.08289795
```

Question 6: Compare estimates of evenness for `site1` of BCI using $E_{1/D}$ and E_{var}. Do they agree? If so, why? If not, why? What can you infer from the results.

> ***Answer 6***: They differ by ~20%. We learned that Simpson's is biased to be more influenced by the most abundant taxa. That the two estimators provide a similar value for evenness could indicate that the patterns in the most abundant taxa do reflect evenness patterns more generally in the community. Alternatively it could be argued that ~20% difference is not similar enough; the logical subsequent inference is that rare taxa do contribute some unique patterns that is not reflected by the most-abundant taxa alone.

5) INTEGRATING RICHNESS AND EVENNESS: DIVERSITY METRICS

So far, we have introduced two primary aspects of diversity, i.e., richness and evenness. Here, we will use popular indices to estimate diversity, which explicitly incorporate richness and evenness We will write our own diversity functions and compare them against the functions in `vegan`.

Shannon's diversity (a.k.a., Shannon's entropy)

In the R code chunk below, please do the following:

1. Provide the code for calculating H' (Shannon's diversity),
2. Compare this estimate with the output of `vegan`'s diversity function using method = "shannon".

```
ShanH <- function(x = ""){
  H = 0
  for (n_i in x){
    if(n_i > 0) {
      p = n_i / sum(x)
      H = H - p*log(p)
    }
  }
  return(H)
}

ShanH(site1)
```

```
## [1] 4.018412
```

```
diversity(site1, index="shannon")
```

```
## [1] 4.018412
```

```
#the outputs are the same
```

Simpson's diversity (or dominance)

In the R code chunk below, please do the following:

1. Provide the code for calculating D (Simpson's diversity),
2. Calculate both the inverse (1/D) and 1 - D,
3. Compare this estimate with the output of `vegan's` diversity function using method = "simp".

```
SimpD <- function(x = ""){
  D = 0
  N = sum(x)
  for (n_i in x){
    D = D + (n_i^2)/(N^2)
  }
  return(D)
}

D.inv <- 1/SimpD(site1)
D.sub <- 1-SimpD(site1)
print (D.inv)
```

```
## [1] 39.41555
```

```
print (D.sub)
```

```
## [1] 0.9746293
```

```
print (diversity(site1, "inv"))
```

```
## [1] 39.41555
```

```
print (diversity(site1, "simp"))
```

```
## [1] 0.9746293
```

Question 7: Compare estimates of evenness for `site1` of BCI using $E_{H'}$ and E_{var}. Do they agree? If so, why? If not, why? What can you infer from the results.

> ***Answer 7***:
>
> Evar is normalized to vary between 0 and 1.
>
> Shannon's is not on that scale, so it is hard to compare them directly.
>
> The Internet indicated that Shannon's typically ranges 1.5 to 3.5, so it could be posited that Shannon's is implying a higher evenness than Evar is.
>
> Too, Shannon's is a diversity index, while Evar measures evenness.
>
> So Evar is intended to be independent of richness, while Shannon's is intended to depend on richness in its calculation.

Fisher's α

In the R code chunk below, please do the following:

1. Provide the code for calculating Fisher's α,
2. Calculate Fisher's α for `site1` of BCI.

```
rac<-as.vector(site1[site1>0])
invD<-diversity(rac,"inv")
print (invD)
```

```
## [1] 39.41555
```

```
Fisher<-fisher.alpha(rac)
print (Fisher)
```

```
## [1] 35.67297
```

Question 8: How is Fisher's $\boldsymbol{\alpha}$ different from $E_{H'}$ and E_{var}? What does Fisher's $\boldsymbol{\alpha}$ take into account that $E_{H'}$ and E_{var} do not?

> ***Answer 8***: Fisher's alpha is an estimator of diversity, much like Chao estimator estimate richness (in contradistinction to observed richness). The fact that Fisher's and invD will converge at much larger number for abundance and richnes paralelles the idea that if we sampled a community (assuming there were singletons or doubletons) absolutely completely, then a Chao estimator would be equal to the observed richness.
>
> Thus, Fisher's takes into account the incompleteness of our sampling (and the sampling error/ stochasticity that can arise from it). Too, Evar is different because it only measures evenness, not diversity.

6) MOVING BEYOND UNIVARIATE METRICS OF α DIVERSITY

The diversity metrics that we just learned about attempt to integrate richness and evenness into a single, univariate metric. Although useful, information is invariably lost in this process. If we go back to the rank-abundance curve, we can retrieve additional information – and in some cases – make inferences about the processes influencing the structure of an ecological system.

Species abundance models

The RAC is a simple data structure that is both a vector of abundances. It is also a row in the site-by-species matrix (minus the zeros, i.e., absences).

Predicting the form of the RAC is the first test that any biodiversity theory must pass and there are no less than 20 models that have attempted to explain the uneven form of the RAC across ecological systems.

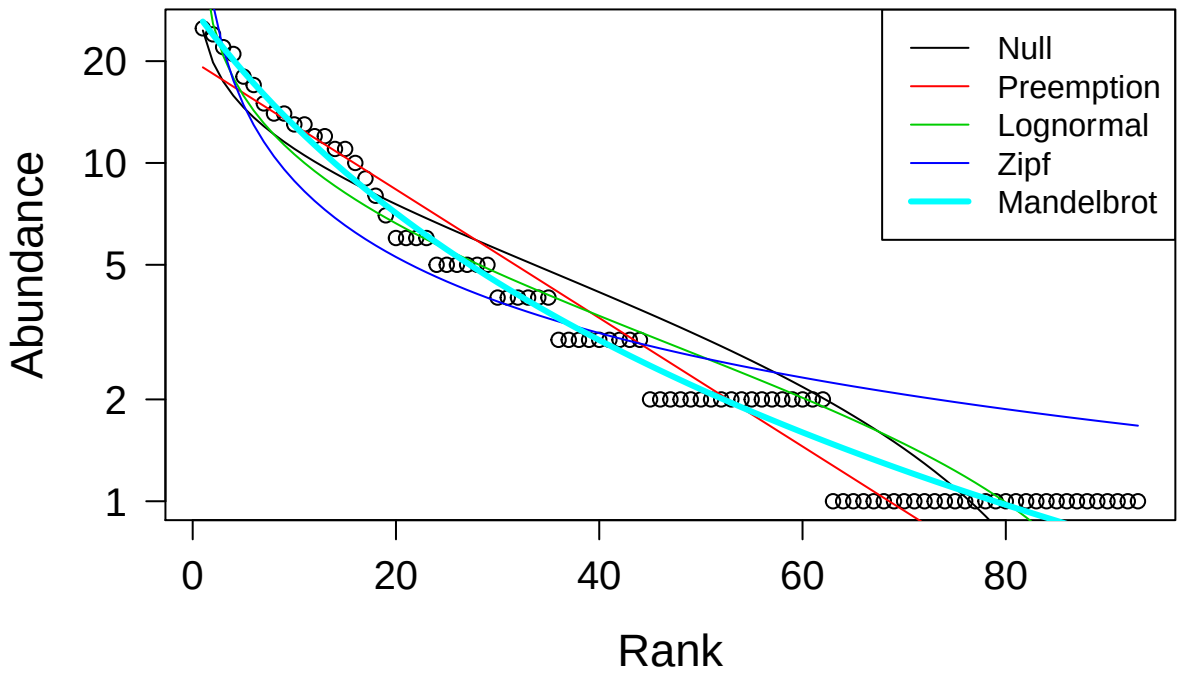
In the R code chunk below, please do the following:

1. Use the `radfit()` function in the `vegan` package to fit the predictions of various species abundance models to the RAC of `site1` in BCI,
2. Display the results of the `radfit()` function, and
3. Plot the results of the `radfit()` function using the code provided in the handout.

```
RACresults<-radfit(site1)
print (RACresults)
```

```
##
## RAD models, family poisson
## No. of species 93, total abundance 448
##
##            par1      par2     par3    Deviance AIC      BIC
## Null                                   39.5261 315.4362 315.4362
## Preemption  0.042797                    21.8939 299.8041 302.3367
## Lognormal   1.0687    1.0186           25.1528 305.0629 310.1281
## Zipf        0.11033  -0.74705          61.0465 340.9567 346.0219
## Mandelbrot  100.52   -2.312    24.084   4.2271 286.1372 293.7350
```

```
plot.new()
plot(RACresults, las = 1, cex.lab = 1.4, cex.axis = 1.25)
```



Question 9: Answer the following questions about the rank abundance curves: a) Based on the output of `radfit()` and plotting above, discuss which model best fits our rank-abundance curve for `site1`? b) Can we make any inferences about the forces, processes, and/or mechanisms influencing the structure of our system, e.g., an ecological community?

> ***Answer 9a***:
> The Mandelbrot model has the lowest AIC. The plot corroborates the idea that this model is the best fit for the data: the Mandelbrot curve passes through the centre of most of the (y-axis) abundances. This is true for all of the abundances.
>
> ***Answer 9b***: I'm not sure. The model has been applied in biology because it does a good job of fitting and predicting patterns, rather than, as far as I am aware, arising from some ecological theory that was posited *first* to explain some phenomenon. If there is a proportional relationship between rank-abundance and abundance, I don't think this would fit a distribution in which the vast majority of taxa are rare (which is, to my knowledge, a common pattern in communities in the wild). However, from the plot we do see that there is a very large. The Mandelbrot method might tells us something about the evenness in the community, but I don't know right now how to access the beta parameter. At a coarse scale, knowing whether the beta parameter is positive or negative could tell us whether there is high or low evenness among the most-abundant spp. Qualitatively, by looking at the graph, I would predict that the beta parameter is negative because there seems to be less evenness among the most-abundant spp. than among the least abundant spp.

Question 10: Answer the following questions about the preemption model: a. What does the preemption model assume about the relationship between total abundance (N) and total resources that can be preempted? b. Why does the niche preemption model look like a straight line in the RAD plot?

> ***Answer 10a***:

I think the preemption model assumes that there is an infinite amount of resources. Fisher's alpha is not dependent upon N, it is just another parameter of the model that can be set by the modeler.

Answer 10b: I believe it is because the decay is exponential, and the y-axis is a log-scale—there appears to be a log-transformation in vegan's radfit function, similar to the log transformation we did just prior to Question #5. I was having difficulty wrapping my head around this, so I wrote a script to try to figure out what was going on (see 10b.py). I played around with it until I saw that the log transformation caused the differences in predicted abundance between each pair of rank-abundances [r, r+1] were equal.

Question 11: Why is it important to account for the number of parameters a model uses when judging how well it explains a given set of data?

Answer 11: It would be best to avoid overfitting a model. With enough parameters, a model could fit any curve nearly perfectly. But if it is fit to that curve so perfectly, it is not very useful for making predictions about other curves. We want to be able to make predictions about patterns, but if a model assume that absolutely every variation is part of pattern, it wil be wrong because it won't be able to distinguish between signal (pattern) and noise (stochastic variation).

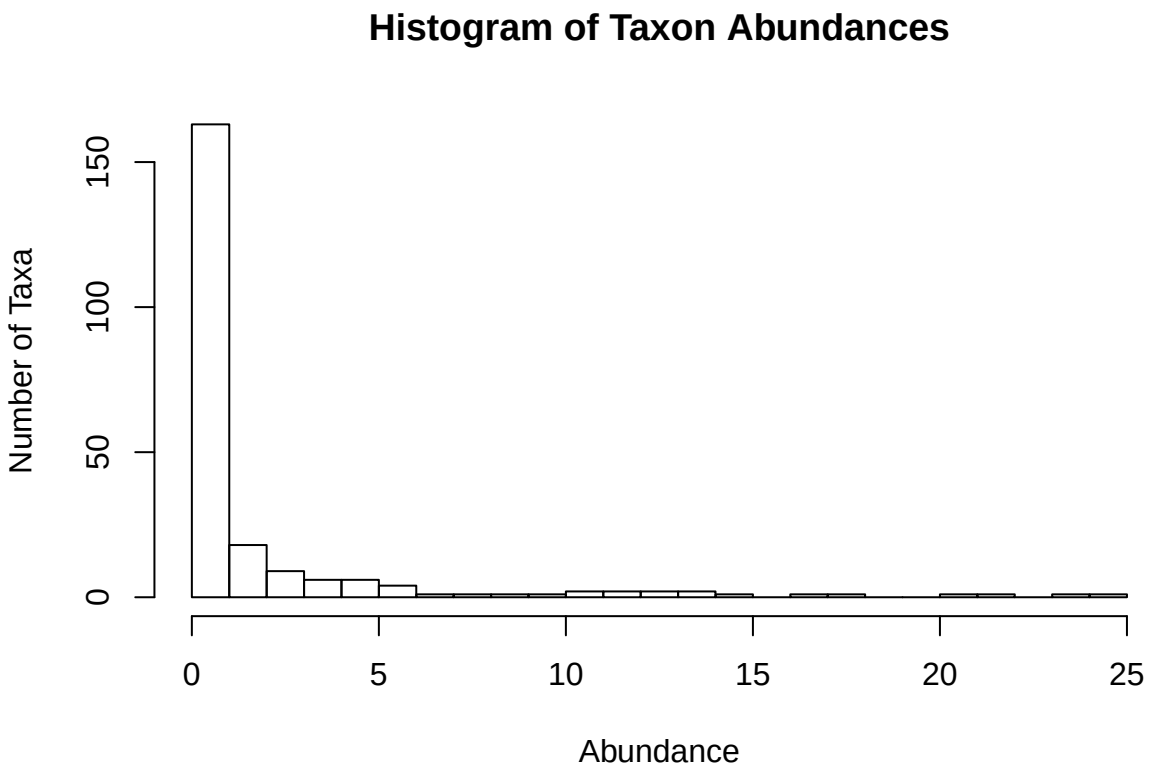
SYNTHESIS

1. As stated by Magurran (2004) the $D = \sum p_i^2$ derivation of Simpson's Diversity only applies to communities of infinite size. For anything but an infinitely large community, Simpson's Diversity index is calculated as $D = \sum \frac{n_i(n_i-1)}{N(N-1)}$. Assuming a finite community, calculate Simpson's D, 1 - D, and Simpson's inverse (i.e. 1/D) for `site 1` of the BCI site-by-species matrix.

```
D.vanilla<-SimpD(site1)
D.inv<-(1/D.vanilla)
D.subtract<-(1-D.vanilla)
```

2. Along with the rank-abundance curve (RAC), another way to visualize the distribution of abundance among species is with a histogram (a.k.a., frequency distribution) that shows the frequency of different abundance classes. For example, in a given sample, there may be 10 species represented by a single individual, 8 species with two individuals, 4 species with three individuals, and so on. In fact, the rank-abundance curve and the frequency distribution are the two most common ways to visualize the species-abundance distribution (SAD) and to test species abundance models and biodiversity theories. To address this homework question, use the R function **hist()** to plot the frequency distribution for `site 1` of the BCI site-by-species matrix, and describe the general pattern you see.

```
plotme<-as.numeric(site1)
#print (plotme)
#?hist
hist(plotme, xlab="Abundance", ylab="Number of Taxa", main="Histogram of Taxon Abundances", breaks=30)
```




```
#There are very few taxa with abundances larger than 5. More than ~67% of taxa have an abundance equal
#As has been noted, there are many more rare taxa than there are common taxa in this community.
```

3. We asked you to find a biodiversity dataset with your partner. This data could be one of your own or it could be something that you obtained from the literature. Load that dataset. How many sites are there? How many species are there in the entire site-by-species matrix? Any other interesting observations based on what you learned this week?

```
#setwd("C:\\Users\\rmoge\\GitHub\\QB2017_DivPro")

plant<-read.csv("C:\\Users\\rmoge\\GitHub\\QB2017_DivPro\\Data\\HF_plants.csv")
print(dim(plant))
```

```
## [1] 96 43
```

```
#There are 96 rows. This means there are 96 "sites"" overall.
#However, sites were resampled yearly each year for 4 years. Too, there were four different treatments.
#Thus there are 96/4/4 = 6 replicate sites for each treatment for each year.
#There are 43 columns.
print(names(plant))
```

```
##  [1] "year"      "plot"      "treatment" "acersp"    "acepen"
##  [6] "acerub"    "aranud"    "aritri"    "betale"    "betlen"
## [11] "betspp"    "carpen"    "casden"    "clibor"    "coptri"
## [16] "craspp"    "denobs"    "denpun"    "dipcom"    "dryspp"
## [21] "faggra"    "gaupro"    "goopub"    "hupluc"    "lyolig"
## [26] "maican"    "medvir"    "mitrep"    "monuni"    "pinstr"
## [31] "pruser"    "quealb"    "querub"    "smirac"    "snag"
```

```
## [36] "tribor"    "tsucan"    "unkspp"    "uvuses"    "vaccspp"
## [41] "vibace"    "vibden"    "viblen"
```

```
#Three columns correspond to year, plot, and treatment. The other 40 are spp.
#install.packages("dplyr")
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## 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
```

```
C_2009_plant<-filter(plant, treatment==1, year==2009)
C_2009_plant<-C_2009_plant[4:43]
print(C_2009_plant)
```

```
##   acersp acepen acerub aranud aritri betale betlen betspp carpen casden
## 1      0      2      4      9      0      0      1      0     18       0
## 2      0      0      4      0      0      0      0      0      0       0
## 3      0      6      6      9      0      0      1      0      0       0
## 4      0      8      3     27      0      0      0      0    108       1
## 5      0      8      5     15      0      0      0      0      0       5
## 6      0     14      5      4      0      0      0      0      0       0
##   clibor coptri craspp denobs denpun dipcom dryspp faggra gaupro goopub
## 1      0      0      0      0      1      0      0      0     54       0
## 2     18      0      0      3    432      0      0      0     44       0
## 3      0      0      0      0      5      1      0      0     52       0
## 4      0      0      0      9    210      0      0      5    166       0
## 5      0      0      0     43      1      4      0      0     52       0
## 6      0      0      0      6      1      0      0      2      4       0
##   hupluc lyolig maican medvir mitrep monuni pinstr pruser quealb querub
## 1      0      0    117      1     53      0      0      0      0      3
## 2      6      0    428      0    267      0      1      0      0      5
## 3      0      0      1      7     99      0      1      7      0      0
## 4      0      0    243     15      0      0      0      1      0      0
## 5     43      0     11      1      0      0      0      5      0      1
## 6      0      0     29      0      0      0      0      1      0      2
##   smirac snag tribor tsucan unkspp uvuses vaccspp vibace vibden viblen
## 1      0    0     11      0      0     17      20      0      0      1
## 2      0    0      0      0      0     21       0      0      0      0
## 3      1    1     13      0      1     39      41      0      4     57
## 4      0    0     31      0      0     21      16      2      0      0
## 5      0    0     55      0      0     23      85      3      1      8
## 6      0    0     14      0      0     10      17      5      0      0
```

```
#class(plant.matrix.C)
#So, for each year & treatment combination, there are 6 sites.
print(S.obs(C_2009_plant))
```

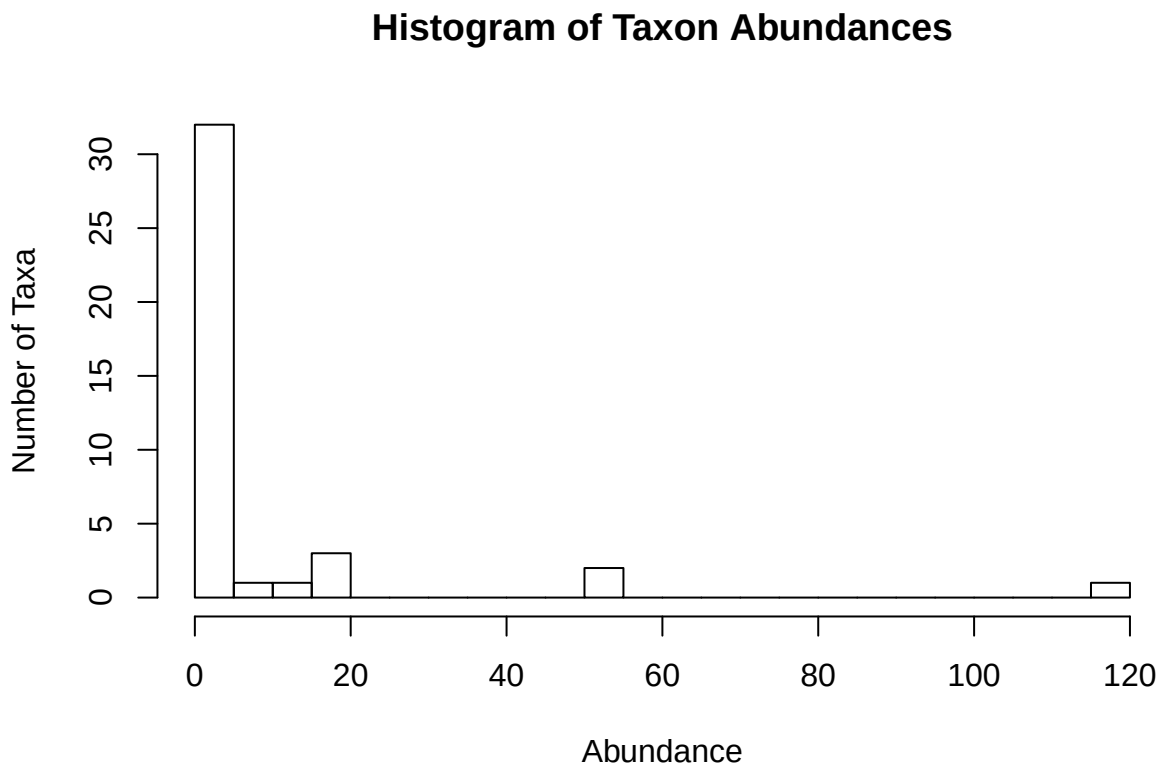
```
## [1] 15 11 20 16 19 14
```

```
#richness can vary a lot between plots: ranges 11 to 20 in these six plots.

#I will plot a histogram for the first site
plotme2<-as.numeric(C_2009_plant[1,])
print(plotme2)
```

```
##  [1]   0   2   4   9   0   0   1   0  18   0   0   0   0   0   1   0   0
## [18]   0  54   0   0   0 117   1  53   0   0   0   0   3   0   0  11   0
## [35]   0  17  20   0   0   1
```

```
hist(plotme2, xlab="Abundance", ylab="Number of Taxa", main="Histogram of Taxon Abundances", breaks=30)
```




```
#once again, many spp are rare, few are abundant

#C09_Chao2<-c(1:6)
#for (i in c(1:6)){
#  temp<-(S.chao2(i, C_2009_plant))
#  C09_Chao2<-temp
#}

C09_H<-c(1:6)
for (i in c(1:6)) {
  temp<-ShanH(C_2009_plant[i,])
  C09_H[i]<-temp
}
print(C09_H)
```

```
## [1] 1.898484 1.404628 2.166397 1.907690 2.300162 2.249852
```

```
C09_H_TEST<-sapply(1:6, function(x) ShanH(C_2009_plant[x,]))
print(C09_H)
```

```
## [1] 1.898484 1.404628 2.166397 1.907690 2.300162 2.249852
```

```
#Note that the plots with highest richness don't correspond very well to those with high Shannon's dive
#I see how I can use sapply to return the same vector as a for loop did.
C09_Chao2_TEST<-(sapply(1:6, function(x) S.chao2(x, C_2009_plant)))
print(C09_Chao2_TEST)
```

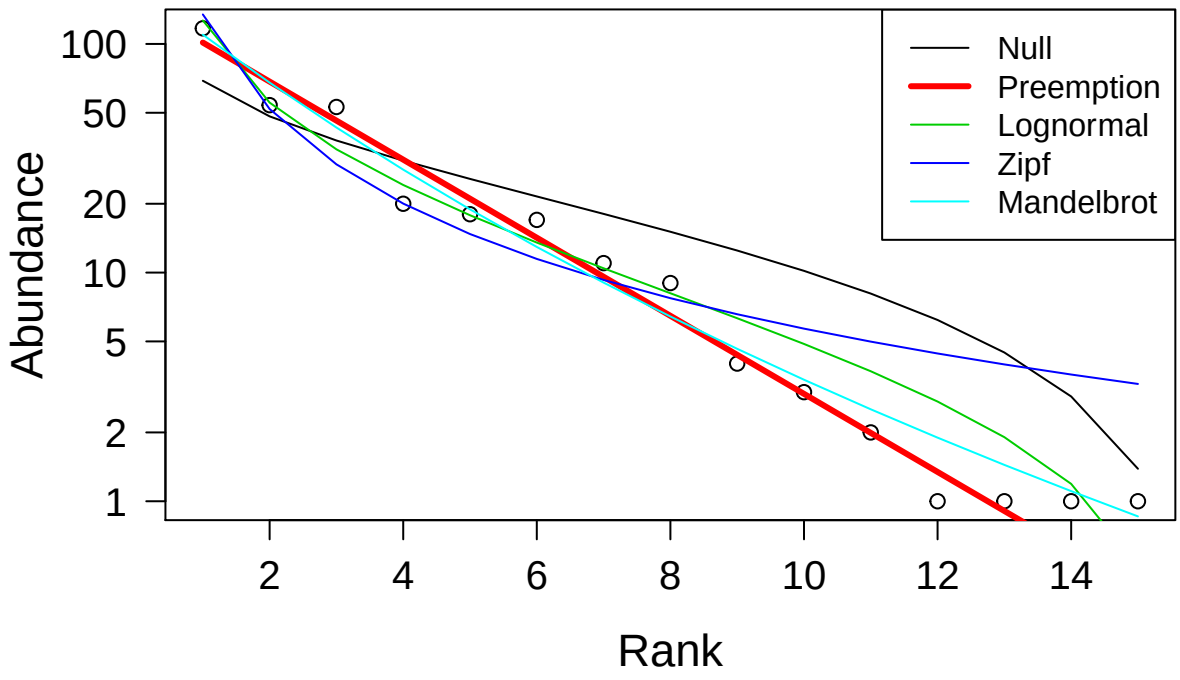
```
##  1  2  3  4  5  6
## 16 12 21 17 20 15
```

```
#estimated richness varies almost as much among these sites, here ranging from 12 to 21.
#estimated richness values are not much larger than observed richness,
#which indicates that the sites were sampled thoroughly.

#Now I would like to draw a rank-abundance curve for the first site
C09_1_RAC_vector<-radfit(C_2009_plant[1,])
print(C09_1_RAC_vector)
```

```
##
## RAD models, family poisson
## No. of species 15, total abundance 312
##
##            par1        par2    par3    Deviance AIC     BIC
## Null                                    81.648  139.327 139.327
## Preemption  0.32505                     14.126   73.805  74.513
## Lognormal   2.0945      1.4981          16.020   77.698  79.115
## Zipf        0.43136    -1.3741          37.544   99.223 100.639
## Mandelbrot  3.5684e+07 -6.9899  12.976  11.694   75.373  77.497
```

```
plot(C09_1_RAC_vector, las = 1, cex.lab = 1.4, cex.axis = 1.25)
```




```
#Lowest AIC model is preemption, although for Mandelbrot DeltaAIC=1.568,
#which may not be enough difference to distinguish between the two models definitively.
```

SUBMITTING YOUR ASSIGNMENT

Use Knitr to create a PDF of your completed alpha_assignment.Rmd document, push it to GitHub, and create a pull request. Please make sure your updated repo include both the HTML and RMarkdown files.

Unless otherwise noted, this assignment is due on **Wednesday, January 25th, 2015 at 12:00 PM (noon)**.