Brief Introduction to Vegan: Community Analysis Package

Data sets and Libraries:
You need to install and load the vegan, MASS, and picante packages. Also lattice for some graphs in vegan

```r
## Load these libraries and datasets
library(vegan)
library(MASS)
library(picante)
library(lattice)
ferp<-read.csv("http://people.ucsc.edu/~ggilbert/Rclass_docs/FERP07data.csv")
data(BCI) #gets BCI community matrix form in 1-ha quadrats

##clean up FERP data and assign trees to 20x20m quadrats
ferp<-ferp[,c(1,2,4,5,6)]
ferp$qE<-trunc((ferp$east-0.001)/20,0)*20
ferp$qN<-trunc((ferp$north-0.001)/20,0)*20
ferp$Q<-paste(ferp$qE,ferp$qN,sep="_")

f<-table(ferp$Q,ferp$code) #makes a Community Matrix for the FERP
```

The vegan package is a collection of functions that are useful for a wide variety of analyses of ecological communities. These include the kinds of multivariate analyses, measures of diversity, and more. ?vegandocs gives a selection of helpful guides

Community data can be stored in a variety of formats. Vegan mostly uses a Community Matrix, where rows are samples and columns are species. Values in each cell \(i,j\) is the number of individuals of species \(j\) in sample \(i\).

<table>
<thead>
<tr>
<th></th>
<th>species1</th>
<th>species2</th>
<th>species3</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample1</td>
<td>5</td>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>sample2</td>
<td>9</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>sample3</td>
<td>22</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

About the FERP data.
We’ll use data from the UCSC Forest Ecology Research Plot for examples. The FERP data include 8,180 individually tagged, mapped, measured, and identified woody plants on the 6-ha UCSC Forest Ecology Research Plot (Gilbert et al. 2010. Beyond the tropics: forest structure in a temperate forest mapped plot. Journal of Vegetation Science 21:388-405.) Each tree is mapped on an East-North (x-y) coordinate system in meters from the southwest corner of the plot. There are 31 woody species.

The FERP data (FERP07data.csv) are in a data-frame form, with one line per tree. Each line includes identifying, size, and location information for that tree.

Here we create a sample by species community matrix assigning each tree to its 20x20-m quadrat, identified by the SW corner of the quadrat in meter from the SW corner of the plot. There are the 150 quadrats.

```r
table(ferp$code) #number of individuals per species on the FERP
```
Simple diversity measures
There are a variety of simple measures of diversity available in picante

Function diversity gives Shannon, Simpson, or Inverse Simpson indices
Shannon $H' = -\Sigma (p_i \ln(p_i))$ where $p$ is the proportion of individuals that are species $i$. 
Simpson's $1-D = 1 - \Sigma (p_i^2)$ using index="simpson"
Inverse Simpson = $1/D = 1/\Sigma (p_i^2)$ using index="invsimpson"

```
# for Fisher's alpha for complete FERP
fisher.alpha(x=table(ferp$code), se=TRUE)
```

# The number of species in each quadrat (simple richness)
```
specnumber(f) # or more specifically specnumber(f, MARGIN=1)
```
# The number of quadrats in which each species appears
```
specnumber(f, MARGIN=2) # MARGIN=2 does by columns
```
# The total number of species on the FERP
```
specnumber(table(ferp$code))
```

# Pielou's evenness $J'$ would then be $H'/H'_{\text{max}}$, where $H'_{\text{max}}=\log(S)$ or
```
diversity(table(ferp$code))/\log(specnumber(table(ferp$code)))
```
**Rarefaction and species accumulation curves**
You can use Hurlbert’s (1971) rarefaction to return the expected number of species in a given number of random individuals taken from a vector of species abundances.

```r
f2 <- table(ferp$code)  # simple vector of species counts on the ferp

# Get number of species expected in given number of individuals on FERP
rarefy(x=f2, sample=100, se=TRUE)  # spp expected in 100 individs

# Get species accumulation curves using specaccum
# several methods available including
# method = "random" (can specify number of permutations)
# method = "rarefaction" (by individuals rather than samples)
# method = "collector" (in order they happen to appear)
# method = "coleman" (from Coleman 1982)
# method = "exact" (expected mean species from Colwell 2004)

# method = random for sample-based rarefaction with 1000 permutations
specaccum(comm=f, method="random", permutations=1000)
sac <- specaccum(comm=f, method="random", permutations=1000)
# save to object
plot(sac)  # plot the species accumulation curve with error bars around the mean
# for various ways to display the curves
par(mfrow=c(1,4));
plot(specaccum(f, method="collector"), main="collector");
plot(sac, ci.type="bar", main="random bar");
plot(sac, ci.type="line", ci.lty=2, main="random line");
plot(sac, ci.type="polygon", main="random polygon");
par(mfrow=c(1,1))
```

![Graphs of species accumulation curves](image-url)
**Species frequency distributions**
The distribution of species in a community are often compared to Preston's lognormal (octaves) or Fisher's log series. In vegan this can be done with the `prestondistr` and `fisherfit` functions. Can also use `fitdistr` to fit to any names distribution. Look at this using the vector of species abundances `f2` where:

```r
f2<-table(ferp$code) #simple vector of species counts on the ferp
```

### # Preston octave (lognormal) distributions
Counts number of species of frequency by doubling octaves. Fits a left-truncated normal distribution to log2 transformed non-pooled observations with direct maximization of log-likelihood. Truncate indicates the left border for log-normal model in log2 units (default is -1 = zero). Truncation has big effect on model fit.

```r
pd<-prestondistr(f2,truncate=-1)
```

#note that prestonfit takes a binning approach that does not work as well

```r
pd #shows summary and observed and fitted frequencies per octave
plot(pd) #plot of observed (bars) and fitted (curve) frequencies
veiledspec(pd) #calculates extrapolated number of species to be expected if preston distn
```

Preston lognormal model
Method: maximized likelihood to log2 abundances
No. of species: 31

<table>
<thead>
<tr>
<th>mode</th>
<th>width</th>
<th>S0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.330030</td>
<td>4.531191</td>
<td>3.286933</td>
</tr>
</tbody>
</table>

#NOTE one the log2 scale, width is the sd and S0 is the expected # spp at the mode

Frequencies by Octave

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observed</td>
<td>2.500000</td>
<td>4.000000</td>
<td>1.500000</td>
<td>2.000000</td>
<td>6.500000</td>
<td>1.500000</td>
<td>2.000000</td>
<td>4.000000</td>
</tr>
<tr>
<td>Fitted</td>
<td>2.509056</td>
<td>2.879866</td>
<td>3.148341</td>
<td>3.278226</td>
<td>3.251199</td>
<td>3.071114</td>
<td>2.763095</td>
<td>1.932584</td>
</tr>
</tbody>
</table>

# To Fit Fisher's log series and get Fisher's alpha
```r
fd<-fisherfit(f2)
```

fd #gives alpha, but look at `str(fd)` for series data

```r
confint(fd,level=.95) #confidence interval for Fisher's alpha
plot(fd) #plots the data and Fisher's log series fit
```
**Extrapolating Species Richness estimates from relative abundances**

There are a number of ways to estimate how many species you would expect in total, based on your samples. See ?specpool help for description of how these are calculated. The estimate function is based on the functions used in Colwell's EstimateS.

From single vector of species abundances

```r
f2<-table(ferp$code)  # single vector of species abundances on the FERP
veiledspec(prestondistr(f2,truncate=-1))  # Number spp expected if from Preston distn
estimateR(f2)  # gives Chao1 and ACE estimates and SE as per EstimateS
```

From a community matrix of many samples

```r
f<-table(ferp$Q,ferp$code)  # makes a Community Matrix for the FERP
specpool(f)  # bootstrapped version of estimate from samples
```

```r
fea<-estaccumR(f, permutations=10)
# estaccum works like specaccum but includes Chao and ACE estimates
plot(fea)  # requires lattice library
```
Beta diversity
There are at least as many ways to measure Beta diversity as alpha diversity. They are mostly different formulations of how many species are in common between two samples, and how many unique. They can be quantitative or presence/absence.

Vegan uses the function betadiver to calculate the full range of diversity measures from a community matrix. The general form is betadiver(x=f, index = "w"). Here "f" is the FERP sample x species community matrix, and "w" refers to the Sorensen Index (w=(b+c)/(2*a+b+c), where a are shared species and b and c are unique to either sample).

To illustrate, let's use "small" – the first 5 quadrats of f.
small<-f[1:5,c(2,7,9,15,16,19,20,22,23,24,28,29,30)]

bdiv<-betadiver(x=small, method="w")
bdiv
0_0     0_100     0_120     0_140
0_100 0.3333333
0_120 0.2500000 0.1764706
0_140 0.2307692 0.1428571 0.2000000
0_160 0.4666667 0.2500000 0.4117647 0.2857143

To get the overall Beta diversity for the FERP plot, take the mean
mean(betadiver(f,"w"))

VARIATIONS
You can use any of the 24 beta-diversity metrics in Koleff et al. 2003 J. Animal. Ecol 72:367-382, either calling them by number or by letter (subscript of Beta). You can see the full list that are available in betadiver by typing betadiver(help=T)
The first 5 are:
1 "w" = (b+c)/(2*a+b+c)
2 "-1" = (b+c)/(2*a+b+c)
3 "c" = (b+c)/2
4 "wb" = b+c
5 "r" = 2*b*c/((a+b+c)^2-2*b*c)

Since most measures of Beta diversity are distance measures, you can use vegdist as well (See Dissimilarity matrices, below)
Note: you get exactly the same output as above from vegdist(small, method="bray", binary=TRUE).

Or, if 24 distance measures from Koleff isn't enough, use designdist to design your own distance metric!
**Dissimilarity matrices**

Vegdist allows you to calculate the multivariate distances between pairs of samples. These dissimilarity matrices form the basis of most kinds of ordinations. There are oodles of different possible measures, set as method. The current choices include manhattan, Euclidean, Canberra, bray, kulczynski, jaccard, gower, morisita, horn, mountford, raup, binomial, and chao. See ?vegdist for details on how each is calculated, and any of many books and papers arguing vehemently about which is best. Or use designdist to design your own distance metric!

vegdist returns a matrix of pairwise distances. This is too big to look at for the full FERP data set, so let's create a subset to play with.

```r
small<-f[1:5,c(2,7,9,15,16,19,20,22,23,24,28,29,30)]
vegdist(small, method="bray")
```

<table>
<thead>
<tr>
<th></th>
<th>0_100</th>
<th>0_100</th>
<th>0_120</th>
<th>0_140</th>
</tr>
</thead>
<tbody>
<tr>
<td>0_100</td>
<td>0.6000000</td>
<td>0.4876033</td>
<td>0.3902439</td>
<td>0.3932584</td>
</tr>
<tr>
<td>0_120</td>
<td>0.4876033</td>
<td>0.2653061</td>
<td>0.5000000</td>
<td>0.5471698</td>
</tr>
<tr>
<td>0_140</td>
<td>0.3902439</td>
<td>0.5000000</td>
<td>0.5151515</td>
<td>0.5471698</td>
</tr>
<tr>
<td>0_160</td>
<td>0.3932584</td>
<td>0.5471698</td>
<td>0.2835821</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>

Notice the default is to not include the distances along the diagonal (identities) or the upper part.

You can specify these with various arguments

```r
vegdist(small, method="bray", diag=T, upper=T)
```

<table>
<thead>
<tr>
<th></th>
<th>0_0</th>
<th>0_100</th>
<th>0_120</th>
<th>0_140</th>
<th>0_160</th>
</tr>
</thead>
<tbody>
<tr>
<td>0_0</td>
<td>0.0000000</td>
<td>0.6000000</td>
<td>0.4876033</td>
<td>0.3902439</td>
<td>0.3932584</td>
</tr>
<tr>
<td>0_100</td>
<td>0.6000000</td>
<td>0.0000000</td>
<td>0.2653061</td>
<td>0.5000000</td>
<td>0.5652174</td>
</tr>
<tr>
<td>0_120</td>
<td>0.4876033</td>
<td>0.2653061</td>
<td>0.0000000</td>
<td>0.5151515</td>
<td>0.5471698</td>
</tr>
<tr>
<td>0_140</td>
<td>0.3902439</td>
<td>0.5000000</td>
<td>0.5151515</td>
<td>0.0000000</td>
<td>0.2835821</td>
</tr>
<tr>
<td>0_160</td>
<td>0.3932584</td>
<td>0.5652174</td>
<td>0.5471698</td>
<td>0.2835821</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>

You can first standardize quantitative data to presence/absence with argument binary

```r
vegdist(small, method="bray", diag=T, upper=T, binary=T)
```

binary actually calls out to function decostand, which includes a broad range of **useful standardizations** for community data.
Non-metric multidimensional scaling

NMDS is the current favorite in community ordination. It avoids most of the pitfalls of the menagerie of ordination techniques of days gone by (available in vegan as cca, rda, decorana)
There are two approaches to MDS. The old (and simpler to look at) approach is called isoMDS. This picks random starting values, then iterates until convergence. It gives one solution, and is subject to getting stuck in local (rather than global) optima.
#note that because isoMDS starts with a distance matrix, it loses track of all species scores.

#get subset of 40 quadrats from the FERP, removing any spp not present selQ<- expand.grid(east=seq(0,160,40),north=seq(0,280,40))
selQ$Q<- paste(selQ$east,selQ$north,sep=" ")
f3<-f[rownames(f)%in%selQ$Q,]; tal<-apply(f3,MARGIN=2,FUN=sum); tal<-tal[tal>0];
f3<-f3[,names(tal)]

f.dis<- vegdist(f3,"bray") #get bray-curtis dissimilarity matrix
f.iso<- isoMDS(d=f.dis, k=2, maxit = 50, trace=T, tol=1e-3)
#k=2 looks for 2 dimensions. Maxit sets max number of iterations to read tolerance level tol.
#trace=T asks to print out progress every 5 steps.

#The output object is a list with points and stress. To visualize:
f.iso$stress
#stres is given in percent, the smaller the better.
#<5 is great, <10 good, >30 poor representation
stressplot(f.iso,f.dis)
#dots should be reasonably close to the line in this Shepard plot
ordiplot(f.iso, type="t",display="sites")
#plot on two ordination axes for sites

> f.iso$stress
[1] 19.35492
metaMDS
metaMDS uses a number of iterations through different starting configurations to avoid the pitfalls of local optima that can cause problems with isoMDS. This is the preferred approach.
f3.meta<-metaMDS(comm=f3, distance="gower", k=2)
# With metaMDS you can start with the community matrix and thus keep the species information
f3.meta  # note that here stress is in proportion and not percent as it is in isoMDS
stressplot(f3.meta)

fig<-%ordiplot(f3.meta, type="n")  # make a blank plot
points(fig,"sites", pch=1, col="blue")  # show sites as blue circles
text(fig,"species", cex=.7, col="red")  # show species as red text
identify(fig,"sites")  # click on figure to identify particular sites

# note, can get basic ordination plot with
plot(f3.meta, type="t")
but ordiplot has more annotation flexibility
Using clustering algorithms for community classifications

Let's use the dominant species of 40 FERP quadrats (f3 – see metaMDS), and classify sites into a smaller number of vegetation types.

```r
f3.dis <- vegdist(f3, method = "bray")  # calculate a distance matrix
clus <- hclust(f3.dis, method = "ward")  # cluster samples based on similarity
plot(clus, cex = .75)  # look at the clustering
rect.hclust(clus, 4)  # group sites into 4 groups
f3.4groups <- cutree(clus, 4)  # save membership in those 4 groups
# convert to a dataframe useful for plotting
bbb <- as.data.frame(f3.4groups); colnames(bbb) <- "group"
bbb$east <- unlist(strsplit(names(f3.4groups), "_"))[seq(1, 79, 2)]
bbb$north <- unlist(strsplit(names(f3.4groups), "_"))[seq(2, 80, 2)]
plot(bbb$east, bbb$north, pch = 15, cex = 3, col = bbb$group)  # plot groups
```