Review

- Dropping rows and columns based on criteria.
  - Apply function – create a vector summarizing rows or columns (mean, min, max, sd etc.)
  - Test that criteria using logical operators (>, <, == etc.) to create a vector of TRUE/FALSE
  - Use that vector to drop rows or columns
    - matrix[,vector] to drop columns
    - Matrix[vector,] to drop rows

Species Accumulation and Rarefaction

- As you accumulate more samples, the number of observed rare events increases.
- Curves display the rate of new species addition with additional sampling.
- Alpha, beta and gamma diversity
- Diverse ecosystems (high gamma and high beta diversity) will display rapid increase to higher plateau. Less diverse: slow increase to lower plateau.

Species Accumulation and Rarefaction

- Function: specaccum
  - specaccum(matrix,method)
- Methods:
  - Accumulate species by:
    - Adding samples in the order collected (“collector”)
    - Adding samples in random order (“random”)
    - Calculating expected (mean) number for each sample size (“exact”)
    - Follow methods of Coleman 1982 (“coleman”)
    - Permuting individuals instead of samples (“rarefaction”)
- “collector” does not provide error estimates.
- “exact” provides a mean and standard deviation for observed data
- “random” provides mean and standard deviation for permuted data

Sites or Samples

Low beta, high alpha, high gamma

Low beta, low alpha, low gamma

Species or Alleles

Species Accumulation and Rarefaction

- Function: specaccum
  - specaccum(community,method="coleman")
- specaccum(community,method="exact")
- specaccum(community,method="random")
Randomizations, bootstraps, jacknifes, Monte Carlo, MCMC

- Function `sample` can be used to randomly sample any data
  - `sample(thing to sample, number of samples, replace)`
- **Bootstrapping** – estimating some parameter by randomly sampling with replacement
  - See also functions in permute package

In our sample community dataset:
- `mean(community) = 14.283`
- `sample(community, 100, replace=TRUE)` # pick 100 values
- `sample(community, 100000, replace=TRUE)` # pick 100000, take mean
- `sample(community, 180, replace=FALSE)` # sample all without replacement

Species Accumulation and Rarefaction

- **Rarefaction** – most commonly used technique for estimating diversity
  - Pool **individuals** within a dataset and resample individuals and not individual cells within the matrix.
  - Use the same `specaccum` function (method="rarefaction").
  - Rarefaction requires integer (count) data, untransformed. Other methods will work with log or proportional transformed data.
  - Number of individuals sampled is set as the mean number of individuals per sample in the dataset.

Species Accumulation and Rarefaction

- Rarefaction curve for sample community data using `specaccum, method="rarefaction"` which set the number of individuals per sample to 143.

Species Accumulation and Rarefaction

- The `specaccum` function uses the function `rrarefy` to do the resampling. You can use `rrarefy` directly if you want more control.
  - Function `rrarefy(communty,sample)`
    - Provide the community matrix and how many individuals to sample and it will return a permuted sample.
  - To duplicate what `specaccum` did:
    - Pool all individuals into 1 sample
    - `pooled<-apply(community,2,sum)`
    - Select 143 individual at random
    - `rrarefy(pooled, sample=143)`
    - Repeat…
Species Accumulation and Rarefaction

- One common use for rarefaction is to control for sampling effort (or efficiency) in estimating diversity.

- In the sample community, the number of individuals per sample ranges from 39 to 630.
  
  ```r
  apply(community, 1, sum)
  ```

- Diversity and the number of individuals is correlated. What is sample diversity if we control for this?

Species Accumulation and Rarefaction

- Function `rarefy` will select a constant number of individuals from each sample
  - `rarefy(community, sample, se)`
    - Provide community data, number to subsample and whether or not you want standard error calculated

- How many individuals should you sample?
  - The number should be less than the number in your least abundant sample
  
  ```r
  min_individuals < min(apply(community, 1, sum))
  ranking(community, sample=min_individuals, se=TRUE)
  ```

- Note that sampling this number results in no error for the sample with the fewest individuals (all individuals are sampled in each permutation)

Species Abundance Curves

- Plot of rank abundance (x-axis) vs abundance or $P_i$ (y-axis).

- More diverse communities lack numerically dominant species, flatter line.

Species Abundance Curves

- Functions: `rankabundance`, `rankabuncomp` in the BiodiversityR package
  - `rankabundance(community)`
    - Can use either raw abundance or proportion data
    - `rankabuncomp` allows for comparison among factors
    - `rankabunplot` will plot the results

```r
rank_data <- rankabundance(community)
rankabunplot(rank_data)
```