Most of the time in this class we will be working with large matrices of data.

```r
community <- read.csv("community_data.csv", header=TRUE, row.names="site")
```

### Summarizing a matrix
- `dim(community)` lists the dimensions of the matrix
- `sum`, `mean` etc. will provide descriptive statistics
- `colnames` and `rownames` will name rows and columns

You can directly transform a matrix like this:
- `community_log <- log(community + 1)`
- `community_sqr <- community^0.5`

### Basic operations
- **Arithmetic operations:** `+`, `-`, `*`, `/`, `%`, `^` - all apply to variables, vectors and a matrix
- **Logical operators:** `==`, `>`, `<`, `<=`, `>=`, `<>` - all apply to variables, vectors and a matrix
- **Variable assignment:** `<-` or `=` (note logical operator is `==`, easy to mistake these and cause errors)
  - `A=5` assigns a value of 5 to `A`
  - `A==5` tests whether variable `A` has a value of 5 (TRUE in this case)

### Selecting portions of a matrix or vector
- Referring to specific parts of a matrix is done with `[row, column]`
- `community[] = whole matrix`
- `community[1] = first column, as a vector`
- `community[2,3] = second row, third column`
- `community[1,] = first row, all columns`
- `community[,1] = first column, all rows`
- You can also specify rows or columns to use by name
  - `community["D_07"]`
  - `community["Species5"]`
Selecting portions of a matrix or vector

- First three rows of the matrix
  - `community[1:3]`
- First three columns of the matrix
  - `community[,1:3]`
- First three rows and columns 2-6 of the matrix
  - `community[1:3,2:6]`
- The same applies to a vector:
  - `community$Species1[1:5]`

Data (matrix) Management

- Function `decostand` (vegan package) performs a number of standard ecological transformations
  - `decostand(matrix, margin, method)`
    - Margin – standardize by row or column
    - Method:
      - total – standardize by row or column total
      - max – standardize by row or column max
      - normalize – row or column sum of squares = 0
      - standardize – scale to mean of 0 and unit variance
      - pa – presence/absence

Functions `sort` and `order`

- `sort(vector, decreasing)`
  - Will sort any vector (number, TRUE/FALSE, string)
  - Sort in either increasing or decreasing order
- `order(vector, decreasing)`
  - Will return a vector that orders the provided vector (number, TRUE/FALSE, string)
  - Order in either increasing or decreasing order

Function `apply`

- Apply a function to a matrix by either row or column
  - `apply(matrix, margin, function)`
  - `colmeans<-apply(community,2,mean)`
  - `rowmeans<-apply(community,1,mean)`

```
> community$Species5
[1] 25  5  2  0  5  0  80  0  4 15  8 19 32  0 11 11 36 21
> sort(community$Species5)
[1]  0  0  0  0  2  4  5  5  8 11 11 15 19 21 25 32 36 80
> sort(community$Species5,decreasing=T)
[1] 80 36 32 25 21 19 15 11 11 8 5 5 4 2 0 0 0 0
> order(community$Species5)
[1] 4 6 8 14 3 9 2 5 11 15 16 10 12 18 1 13 17 7
> order(community$Species5, decreasing=T)
[1] 7 17 13 1 18 12 10 15 16 11 2 5 9 3 4 6 8 14
```
Data (matrix) Management

- Combine standardizations and function `apply` to get quick community metrics
  - `community_log <- log(community+1)`
  - `community_total <- decostand(community, MARGIN=1, method="total")`
  - `community_pa <- decostand(community, method="pa")`
- Number of species by site
  - `apply(community_pa,1,sum)`
- Number of occurrences by species
  - `apply(community_pa,2,sum)`
- Change the sum to mean to get proportional measures
- Proportion of species found by site
  - `apply(community_pa,1,mean)`
- Proportion of sites where each species occurs
  - `apply(community_pa,2,mean)`

Factors

- Factors are categorical variables that are treated differently.
- Factors have levels and are used to categorize data.
- When loading data, R will try and figure out what is and is not a factor.
- Function `factor` will turn things into a factor
- Function `levels` will describe what levels are in a factor
- Function `is.factor` will test if something is a factor

- `site_category <- c("Y08","Y08","Y08","Y08","Y08","Y08","Y09","Y09","Y09","Y09","Y09","Y09","Y07","Y07","Y07","Y07","Y07","Y07")`
- `summary(site_category)`
- `is.factor(site_category)`
- `levels(site_category)`
- `levels(site_category)[1]`
Data (matrix) Management

- Function `subset` will divide a vector or matrix based on criteria given
  - `subset(matrix, criteria)`

- Only use sites where species 1 occurred
  - `community_sp1_present<-subset(community,Species1>0)`

- Only use sites where species 1 or 2 occurred
  - `community_sp1or2_present<-subset(community,Species1+Species2>0)`

Function `write.csv`

- Works similar to read.csv, except it writes files. Any matrix or data frame can be written to a file.
  - `write.csv(object, "filename.csv")`
- `sep` - specify if you want something besides commas separating fields
- `quote` - true/false to put each value in quotes
- `dec` – specify the number of decimal values

Assignment

- Download the Brier Creek dataset from the webpage
- Load data in R and get:
  - Dimensions of the matrix (dim function)
  - List variables in the data (names function)
- Get the five most abundant species (apply function)
- Create a presence/absence matrix (decostand function), save it as a new object
- Get number of species by site (apply function)
- Eliminate sites with 2 or fewer species, saving the result as a new object. From that new object, eliminate species that have a total abundance <10, saving the result as a new object (your final matrix).
- Get the dimensions of the final matrix.
- Log+1 transform the final matrix (log function).
- Save the final matrix (write.csv function).