Data (matrix) Management

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

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

Row names

site	Species1	Species2	Species3	Species4	Species5	Species6	Species7	Species8	Species9	Species10
A_08	0	85	0	20	25	10	0	10	0	190
B_08	0	53	0	12	5	0	12	0	0	43
C_08	1	38	1	16	2	20	35	2	1	16
D_08	1	45	2	2	0	8	31	0	0	49
E_08	0	2	0	5	5	9	9	3	0	6
F_08	0	4	1	4	0	16	34	2	0	38
A_09	0	250	0	165	80	15	0	10	0	110
B_09	0	14	0	29	0	6	0	0	0	13
C_09	0	48	0	18	4	11	19	3	0	20
D_09	0	21	1	12	15	7	13	0	0	18
E_09	2	22	0	30	8	6	7	1	1	9
F_09	1	35	1	17	19	21	15	3	0	45
A_07	0	26	0	20	32	20	0	0	0	12
B_07	0	27	0	4	0	0	0	0	0	9
C_07	1	27	1	12	11	7	3	3	1	45
D_07	3	26	1	15	11	2	3	2	0	26
E_07	0	36	0	11	36	1	1	0	0	13
F_07	0	49	2	4	21	13	3	2	1	10

MARGIN

MARGIN

Data (matrix) Management

- 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

```
> 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
```

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)

site	Species1	Species2	Species3	Species4	Species5	Species6	Species7	Species8	Species9	Species10
A 08	0	85	0	20	25	10	0	10	0	190
B 08	0	53	0	12	5	0	12	0	0	43
C 08	1	38	1	16	2	20	35	2	1	16
D 08	1	45	2	2	0	8	31	0	0	49
E 08	0	2	0	5	5	9	9	3	0	6
F 08	0	4	1	4	0	16	34	2	0	38
A 09	0	250	0	165	80	15	0	10	0	110
B 09	0	14	0	29	0	6	0	0	0	13
C 09	0	48	0	18	4	11	19	3	0	20
D 09	0	21	1	12	15	7	13	0	0	18
E 09	2	22	0	30	8	6	7	1	1	9
F 09	1	35	1	17	19	21	15	3	0	45
A 07	0	26	0	20	32	20	0	0	0	12
B 07	0	27	0	4	0	0	0	0	0	9
C 07	1	27	1	12	11	7	3	3	1	45
D 07	3	26	1	15	11	2	3	2	0	26
E 07	0	36	0	11	36	1	1	0	0	13
F 07	0	49	2	4	21	13	3	2	1	10

34.0
12.5
…
Etc.

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)

Data (matrix) Management

- You can also directly exclude portions of a matrix in these functions
  - community>10 – this will return a TRUE/FALSE matrix based on whether a species abundance is above or below 10
  - Define rare species as <2
    - community_norare<-community>2
  - Diversity by site
    - apply(community_pa,1,sum)
  - Mean abundance by site excluding rare species 2
    - apply(community_pa*community_norare,1,sum)

Data (matrix) Management

- Exclude rows (sites) based on abundance – drop the three sites with the lowest abundance
  - row_abundance<-apply(community,1,sum)
  - cutoff<-sort(row_abundance, decreasing=FALSE)[3]
  - community_dropsites<-community[row_abundance>cutoff,]
- Exclude matrix columns (species) based on abundance – drop the three species with the lowest abundance
  - col_abundance<-apply(community,2,sum)
  - cutoff<-sort(col_abundance, decreasing=FALSE)[3]
  - community_ dropsp<-community[,col_abundance>cutoff]

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","Y09","Y09","Y09","Y09","Y09","Y09","Y07","Y07","Y07","Y07
","Y07","Y07")
> summary(site_category) Length Class Mode 18 character character
> is.factor(site_category) [1] FALSE
> site_category<-factor(site_category)
> summary(site_category) Y07 Y08 Y09
                         6 6 6
> is.factor(site_category) [1] TRUE
> levels(site_category) [1] "Y07" "Y08" "Y09"
```

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).