# Community ecology- Computer lab I - AB332

Ramiro Logares (ICM), with additions by Anders K. Krabberød (UiO)

October 2021

> This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.
>
> Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.
>
> When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).
>
> The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

It's time for you to try and do the same analysis as was shown in the lecture but using a different dataset: Make sure you have installed all packages!

### Load Packages

## Starting community ecology analyses

Read the data from the github page:

```
otu.tab<-read_tsv("https://raw.githubusercontent.com/krabberod/UNIS_AB332_2021/main/c
omputer_lab/data/AB332_otutab_reduc3.txt")

## Rows: 3697 Columns: 83

## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (1): OTUNumber
## dbl (82): Isa_111214, Isa_120117, Isa_120128, Isa_120209, Isa_120216, Isa_12...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

First, get to know the data: How many samples and how many OTUs are in the dataset? What do the numbers in the sample names mean?

```
head(otu.tab)

## # A tibble: 6 x 83
##   OTUNumber Isa_111214 Isa_120117 Isa_120128 Isa_120209 Isa_120216 Isa_120223
##   <chr>          <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 OTU1             436        348        236        139        260        446
## 2 OTU10           2308       2537       1599       1956        328        668
## 3 OTU100             83          36          21          30           1          21
## 4 OTU1000             0           0           0           0           0           0
## 5 OTU1001             1           0           0           0           0           2
## 6 OTU1002             0           2           0           1           0           0
## # ... with 76 more variables: Isa_120301 <dbl>, Isa_120308 <dbl>,
## #   Isa_120320 <dbl>, Isa_120321 <dbl>, Isa_120322 <dbl>, Isa_120323 <dbl>,
## #   Isa_120329 <dbl>, Isa_120403 <dbl>, Isa_120411 <dbl>, Isa_120416 <dbl>,
## #   Isa_120419 <dbl>, Isa_120423 <dbl>, Isa_120426 <dbl>, Isa_120430 <dbl>,
## #   Isa_120503 <dbl>, Isa_120507 <dbl>, Isa_120508 <dbl>, Isa_120509 <dbl>,
## #   Isa_120510 <dbl>, Isa_120516 <dbl>, Isa_120524 <dbl>, Isa_120621 <dbl>,
## #   Isa_120706 <dbl>, Isa_120806 <dbl>, Isa_120823 <dbl>, Isa_120906 <dbl>, ...

dim(otu.tab)

## [1] 3697   83
```

You can look at a given selection of the table by specifying a range of rows and columns:

```
otu.tab[5:15,1:5] # The first 10 rows, and the first 5 columns

## # A tibble: 11 x 5
##    OTUNumber Isa_111214 Isa_120117 Isa_120128 Isa_120209
##    <chr>          <dbl>      <dbl>      <dbl>      <dbl>
##  1 OTU1001            1          0          0          0
##  2 OTU1002            0          2          0          1
##  3 OTU1003            3          0          0          0
##  4 OTU1004            9          2          5          7
##  5 OTU1005            1          0          2          0
##  6 OTU1006            0          2          0          4
##  7 OTU1007            0          0          0          0
##  8 OTU1008            0          0          0          0
##  9 OTU1009            0          0          0          1
## 10 OTU101             3          0          0          0
## 11 OTU1010            1          3          0          3
```

You can also see the entire table withe View() function:

```
View(otu.tab)
```

See if you can choose a different subset. For instance samples 6-12 and Otus 20-26:

We assign OTUnumbers as rownames

```
otu.tab <- column_to_rownames(otu.tab, var = "OTUNumber")
```

Let’s check the names

```
head(rownames(otu.tab))

## [1] "OTU1"    "OTU10"   "OTU100"  "OTU1000" "OTU1001" "OTU1002"

dim(otu.tab)

## [1] 3697   82
```

For simplicity, I have included only the 25 samples in the rest of the tutorial. As an exercise, you should redo the analysis with the full dataset. I.e. remove the part of the code that selects samples

1:15 in the following chunk. (This way your numbers will differ from the pdf, and you can also see the effect of a different dataset).

```
otu.tab.red<-otu.tab[,6:30]
```

The data needs to be transposed since this is how Vegan likes it.

```
otu.tab.simple<-t(otu.tab.red)
otu.tab.simple[1:5,1:5]

##            OTU1 OTU10 OTU100 OTU1000 OTU1001
## Isa_120223  446   668     21       0       2
## Isa_120301  149   551     23       1       1
## Isa_120308  321  1462     99      11       0
## Isa_120320  204   896     75       1       0
## Isa_120321  442   646     61       4       0
```

You can get the total number of reads for each sample using rowSums(), and the total reads per OTU with colSums()

```
rowSums(otu.tab.simple)

## Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321 Isa_120322 Isa_120323
##      13682      15783      32833      19361      17110      13658      16251
## Isa_120329 Isa_120403 Isa_120411 Isa_120416 Isa_120419 Isa_120423 Isa_120426
##      17002      13551      25606      29877      16194      17161      22305
## Isa_120430 Isa_120503 Isa_120507 Isa_120508 Isa_120509 Isa_120510 Isa_120516
##      19942      25237      27389      19969      34666      17682      25256
## Isa_120524 Isa_120621 Isa_120706 Isa_120806
##      40267      24636      23156      20390

head(colSums(otu.tab.simple)) # Too many to show them all.

##    OTU1   OTU10  OTU100 OTU1000 OTU1001 OTU1002
##  100420    7814     786      54       4      28
```

Since I have selected only a few of the samples is possible that some of the OTU's are left with a total abundance of zero. In R it is possible to have functions within functions so the following will print the number of columns in the data set that has a sum equal to 0:

```
length(which(colSums(otu.tab.simple)==0))

## [1] 1163
```

We can use the same idea of a function within a function to exclude the OTUs with a total number of 0.

```
otu.tab.simple<-otu.tab.simple[,-(which(colSums(otu.tab.simple)==0))]
```

Now how many are 0?

```
length(which(colSums(otu.tab.simple)==0))

## [1] 0
```

How many have more than 0 reads?

```
length(which(colSums(otu.tab.simple)>0))

## [1] 2534
```

Can you find how many OTU's that have more than 10 reads (in total)?

## Common metrics and methods

The following calculations make use of functions in the vegan package written by Jari Oksanen. *Vegan is an R package for community ecologists. It contains the most popular methods of multivariate analysis needed in analysing ecological communities, and tools for diversity analysis, and other potentially useful functions*. If you want to learn more about the vegan you can run `browseVignettes("vegan")`

### Richness estimations

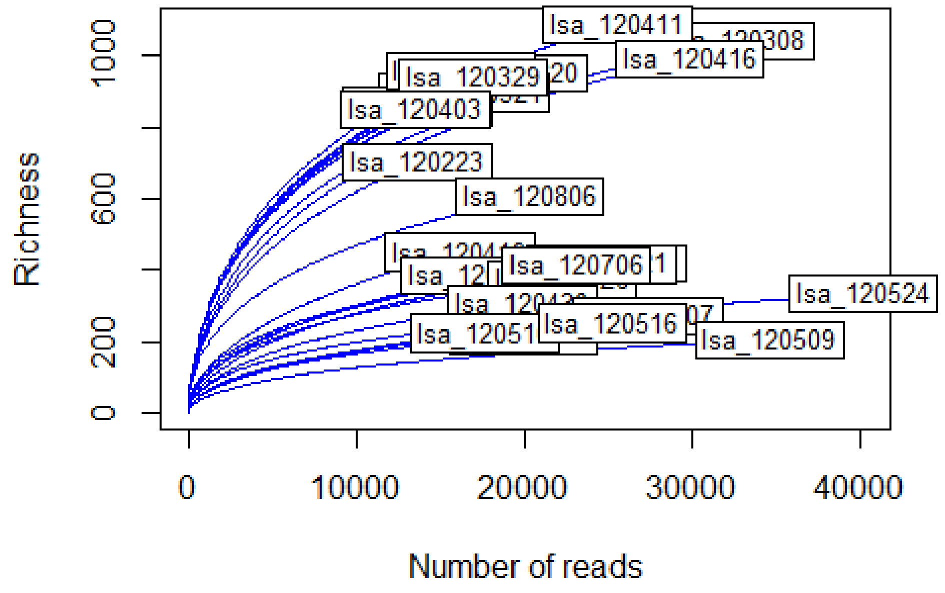
```
richness<-estimateR(otu.tab.simple)
richness

##           Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321 Isa_120322
## S.obs      701.00000  902.00000 1042.00000  952.00000  897.00000  858.00000
## S.chao1   1100.22581 1260.69343 1355.30612 1316.81633 1200.76642 1337.40517
## se.chao1    66.62128   54.17742   47.66939   53.77784   47.47591   72.14089
## S.ACE     1036.24030 1249.23447 1337.63473 1341.50902 1197.73116 1278.33686
## se.ACE      17.06514   18.59112   18.50405   19.72472   17.80976   19.74198
##           Isa_120323 Isa_120329 Isa_120403 Isa_120411 Isa_120416 Isa_120419
## S.obs      959.00000  942.00000  848.00000 1088.00000  992.00000  450.00000
## S.chao1   1377.95000 1397.55469 1155.65000 1517.36486 1287.23077  731.44286
## se.chao1    60.96351   67.07142   47.65508   61.13897   44.73270   54.74508
## S.ACE     1365.23150 1356.87666 1181.99016 1480.20490 1295.47022  759.36594
## se.ACE      19.72705   19.88369   18.05970   20.02279   18.55676   16.47752
##           Isa_120423 Isa_120426 Isa_120430 Isa_120503 Isa_120507 Isa_120508
## S.obs      384.00000  374.00000  304.00000  420.00000  272.00000 219.000000
## S.chao1    693.25532  511.60000  438.63830  654.23077  447.77778 324.636364
## se.chao1    67.46164   32.24236   34.05155   48.23404   45.97426  31.165443
## S.ACE      652.06713  539.26272  435.54374  684.48661  432.26064 329.851638
## se.ACE      14.78726   12.98133   10.86998   15.37044   11.73649   9.569067
##           Isa_120509 Isa_120510 Isa_120516 Isa_120524 Isa_120621 Isa_120706
## S.obs     205.000000  224.00000  252.00000  335.00000  417.00000  414.00000
## S.chao1   363.052632  371.17143  369.00000  528.75000  618.88235  582.52174
## se.chao1   51.971891   40.07574   33.24605   48.16268   46.10172   36.35228
## S.ACE     304.844016  395.87277  364.99135  514.52628  590.13947  615.28756
## se.ACE      9.305907   12.00009   10.37553   12.39624   13.01145   13.57143
##           Isa_120806
## S.obs      605.00000
## S.chao1    873.07692
## se.chao1    50.85980
## S.ACE      851.28562
## se.ACE      15.83698
```

Above we have the estimators Chao and ACE as well as the species number. What do the numbers mean?

### Rarefaction

Let's calculate the number of reads per sample.

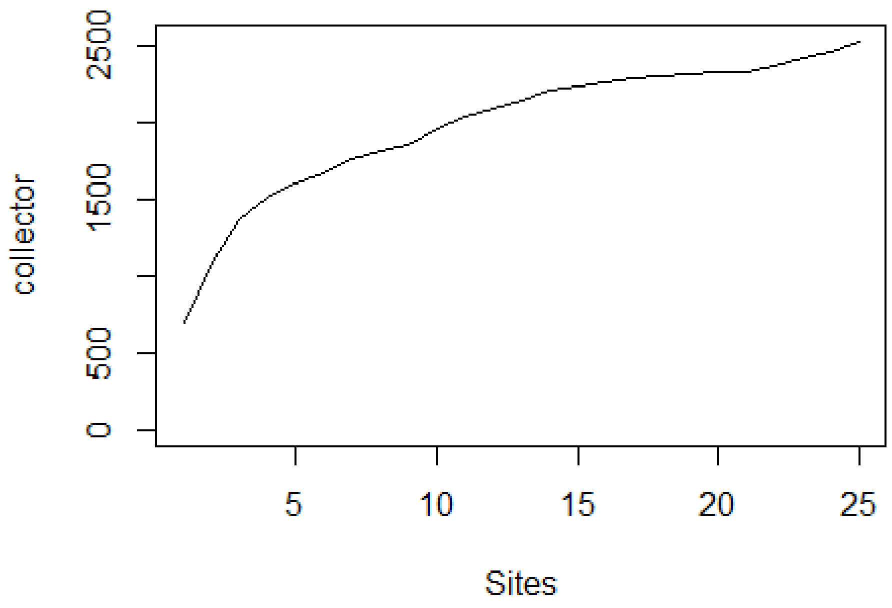
```
rowSums(otu.tab.simple)

## Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321 Isa_120322 Isa_120323
##      13682      15783      32833      19361      17110      13658      16251
## Isa_120329 Isa_120403 Isa_120411 Isa_120416 Isa_120419 Isa_120423 Isa_120426
##      17002      13551      25606      29877      16194      17161      22305
## Isa_120430 Isa_120503 Isa_120507 Isa_120508 Isa_120509 Isa_120510 Isa_120516
##      19942      25237      27389      19969      34666      17682      25256
## Isa_120524 Isa_120621 Isa_120706 Isa_120806
##      40267      24636      23156      20390

rarecurve (otu.tab.simple, step=100, xlab= "Number of reads", ylab="Richness", col="b
lue")
```



How do you interpret these curves? Which samples have the lowest number of total reads? Which are the highest?

## Accumulation curves

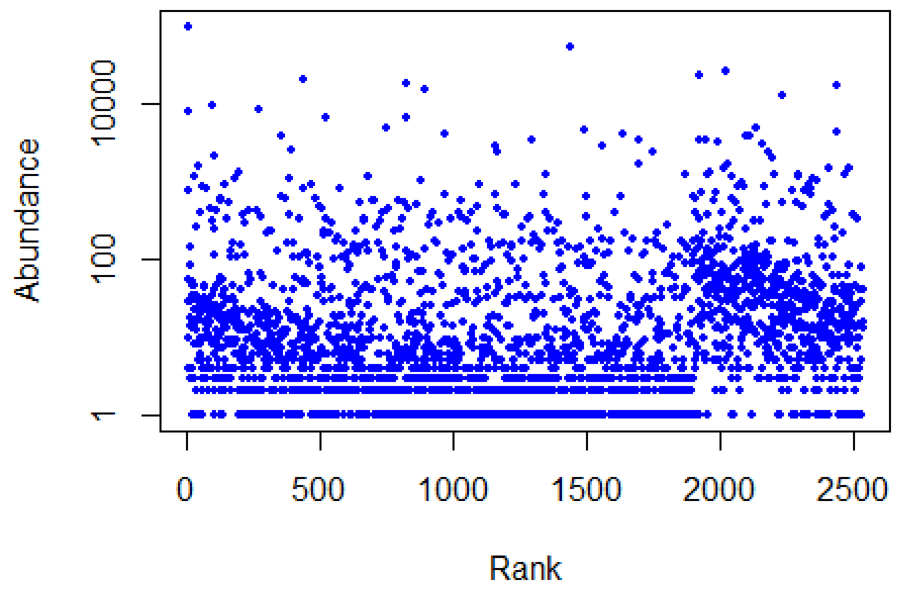
```
accum.curve<-specaccum(otu.tab.simple, method="collector")
plot(accum.curve)
```



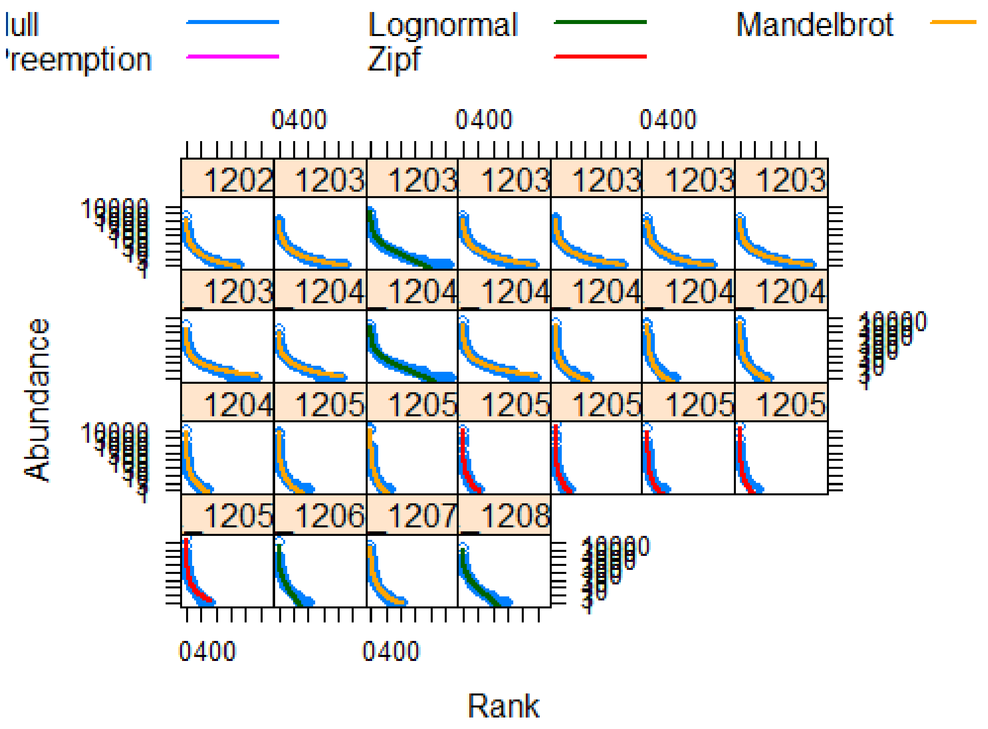
What does this curve represent? How do you interpret it?

## Evenness

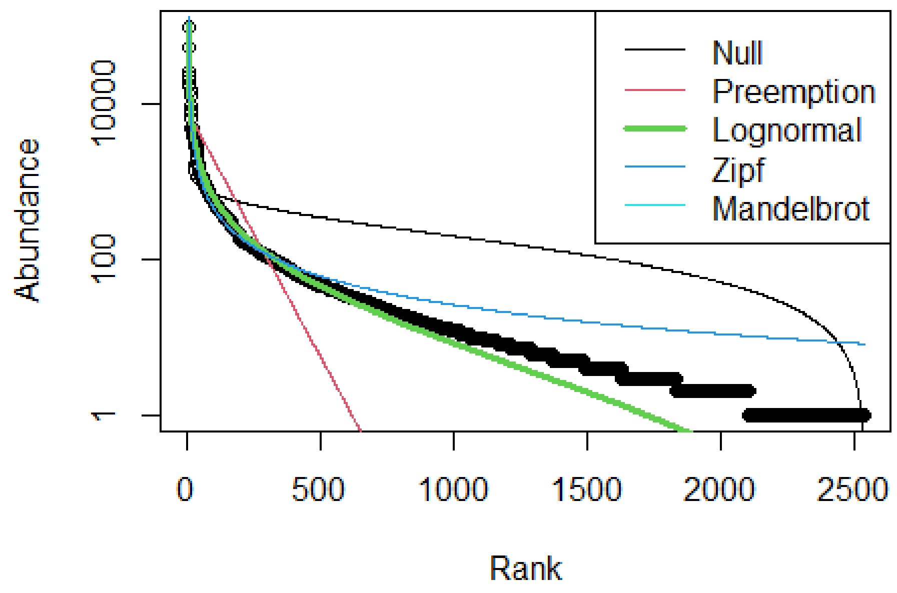
```
plot(colSums(otu.tab.simple),log="y",xlab="Rank", ylab="Abundance", pch=19, cex=0.5,
col="blue")
```



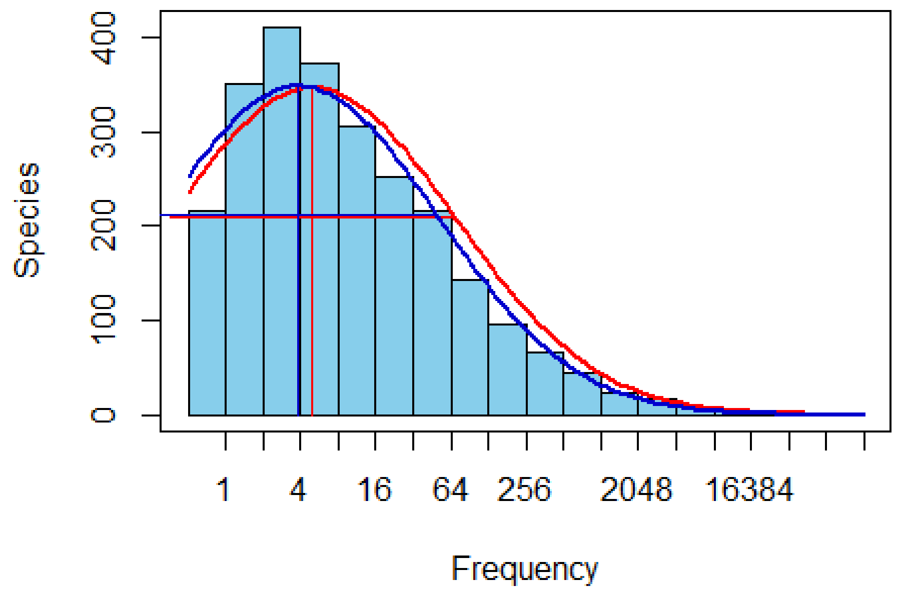
```
#Fitting rank-abundance distribution models to the data
mod<-radfit(otu.tab.simple)

## Error in glm.fit(x = structure(c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  :
##   NA/NaN/Inf in 'x'

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: algorithm did not converge

plot(mod)
```



```
mod.all<-radfit(colSums(otu.tab.simple))

## Error in glm.fit(x = structure(c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  :
##   NA/NaN/Inf in 'x'

plot(mod.all)
```



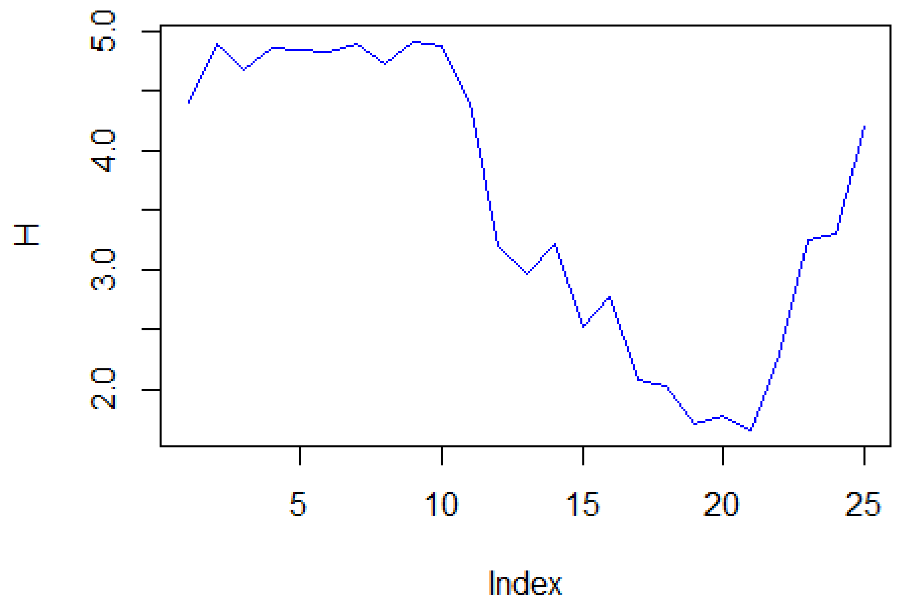
```
#Fitting data to the Preston model
preston<-prestonfit(colSums(otu.tab.simple))
preston.dist<-prestondistr(colSums(otu.tab.simple))
plot(preston)
lines(preston.dist, line.col="blue3")
```



```
## Extrapolated richness
veiledspec(preston)

## Extrapolated      Observed       Veiled
##    3278.6837     2534.0000     744.6837

veiledspec(preston.dist)

## Extrapolated      Observed       Veiled
##    3211.9212     2534.0000     677.9212
```

**Shannon H index (considers richness and evenness)**

```
H<-diversity(otu.tab.simple, index="shannon")
H

## Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321 Isa_120322 Isa_120323
##   4.407409   4.902868   4.679546   4.859799   4.844499   4.826550   4.900137
## Isa_120329 Isa_120403 Isa_120411 Isa_120416 Isa_120419 Isa_120423 Isa_120426
##   4.725430   4.914670   4.872817   4.400562   3.203907   2.970289   3.211268
## Isa_120430 Isa_120503 Isa_120507 Isa_120508 Isa_120509 Isa_120510 Isa_120516
##   2.520753   2.781324   2.086866   2.028217   1.709035   1.784380   1.662585
## Isa_120524 Isa_120621 Isa_120706 Isa_120806
##   2.295421   3.245357   3.311757   4.199123

plot(H, type="l", col="blue")
```



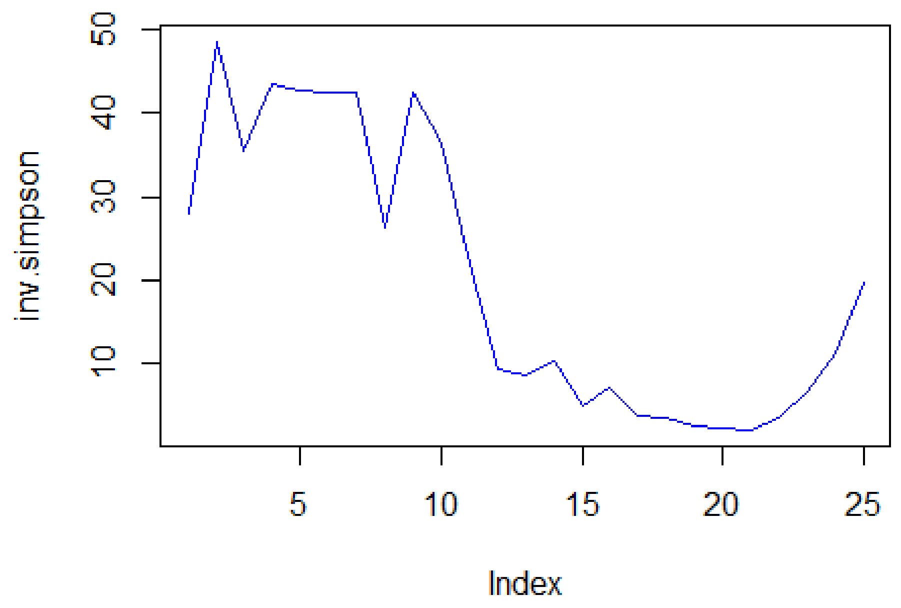
**Pielou's index of evenness (range 0-1, 1 = maximum evenness)**

```
J=H/Hmax
J=Shannon (H) / log(S=species richness)
```

```
J <- H/log(rowSums(otu.tab.simple>0))
```

**Inverse Simpson's D index (richness+evenness. Larger values, larger diversity)**

```
inv.simpson<-diversity(otu.tab.simple, "invsimpson")
plot(inv.simpson, type="l", col="blue")
```



## Beta diversity

We rarefy all samples to the same sequencing depth, to reduce biases

```
min(rowSums(otu.tab.simple)) # We calculate the sample with the minimum amount of rea
ds

## [1] 13551

otu.tab.simple.ss<-rrarefy(otu.tab.simple, min(rowSums(otu.tab.simple))) #Samples are
rarefied to lowest number of reads
rowSums(otu.tab.simple.ss)

## Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321 Isa_120322 Isa_120323
##      13551      13551      13551      13551      13551      13551      13551
## Isa_120329 Isa_120403 Isa_120411 Isa_120416 Isa_120419 Isa_120423 Isa_120426
##      13551      13551      13551      13551      13551      13551      13551
## Isa_120430 Isa_120503 Isa_120507 Isa_120508 Isa_120509 Isa_120510 Isa_120516
##      13551      13551      13551      13551      13551      13551      13551
## Isa_120524 Isa_120621 Isa_120706 Isa_120806
##      13551      13551      13551      13551
```

What is the number of reads these samples have been rarified to? What does it imply, do you understand how it is done?

Check that the number of OTUs are the same in the new table

```
dim(otu.tab.simple)
```

```
## [1]   25 2534

dim(otu.tab.simple.ss)

## [1]   25 2534
```

The tables have the same size, but, after removing reads, several OTUs might be left with zero read abundance.

```
length(which(colSums(otu.tab.simple)==0))

## [1] 0

length(which(colSums(otu.tab.simple.ss)==0))

## [1] 198

head(which(colSums(otu.tab.simple.ss)==0)) # Show the OTUs and the position in the ta
ble that have 0 abundance for the first OTUs

## OTU1016 OTU1032 OTU1043 OTU1075 OTU1103 OTU1221
##      21      37      45      71     100     199
```

We can compare the number of reads for one of the OTUs:

```
colnames(otu.tab.simple)[13]

## [1] "OTU1009"

otu.tab.simple[,13] # This gives the abundance of the OTU1009  across the different s
amples in the table that is NOT subsampled

## Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321 Isa_120322 Isa_120323
##          0          0          0          0          0          0          0
## Isa_120329 Isa_120403 Isa_120411 Isa_120416 Isa_120419 Isa_120423 Isa_120426
##          0          0          0          0          0          0          0
## Isa_120430 Isa_120503 Isa_120507 Isa_120508 Isa_120509 Isa_120510 Isa_120516
##          0          0          0          0          0          0          0
## Isa_120524 Isa_120621 Isa_120706 Isa_120806
##          0          0          0          3

otu.tab.simple.ss[,13] # # This gives the abundance of the OTU1009  across the differ
ent samples in the table that IS subsampled

## Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321 Isa_120322 Isa_120323
##          0          0          0          0          0          0          0
## Isa_120329 Isa_120403 Isa_120411 Isa_120416 Isa_120419 Isa_120423 Isa_120426
##          0          0          0          0          0          0          0
## Isa_120430 Isa_120503 Isa_120507 Isa_120508 Isa_120509 Isa_120510 Isa_120516
##          0          0          0          0          0          0          0
## Isa_120524 Isa_120621 Isa_120706 Isa_120806
##          0          0          0          3
```

We can remove the OTUs with zero abundance with a similar command as we used at the beginning of the lab:

```
otu.tab.simple.ss.nozero<-otu.tab.simple.ss[,-(which(colSums(otu.tab.simple.ss)==0))]
# Removes OTUs with zero abundance
length(which(colSums(otu.tab.simple.ss.nozero)==0)) # Check that no zero abundance OT
Us are left

## [1] 0
```

Let’s check dimensions of the tables:

```
dim(otu.tab.simple.ss)

## [1]   25 2534

dim(otu.tab.simple.ss.nozero)

## [1]   25 2336
```

2548-2226 = 322 , This is the number of OTUs that we expected to be removed. ## Compositional data analyses Replace zeros (problems with log calculations) with pseudo-counts

```
otu.tab.simple.gbm<-cmultRepl(t(otu.tab.simple), output = "p-counts")

## No. corrected values:  48248

otu.tab.simple.gbm[1:5,1:5] # We have a look to the replaced values

##            Isa_120223 Isa_120301   Isa_120308   Isa_120320   Isa_120321
## OTU1     4.460000e+02        149 3.210000e+02 2.040000e+02 4.420000e+02
## OTU10    6.680000e+02        551 1.462000e+03 8.960000e+02 6.460000e+02
## OTU100   2.100000e+01         23 9.900000e+01 7.500000e+01 6.100000e+01
## OTU1000 7.334094e-03          1 1.100000e+01 1.000000e+00 4.000000e+00
## OTU1001 2.000000e+00          1 1.024907e-03 1.230456e-03 3.803261e-04
```

### centered log-ratio (clr) transformation

```
otu.tab.simple.gbm.clr<-clr(otu.tab.simple.gbm) # We apply a centered log-ratio (clr)
transformation
otu.tab.simple.gbm.clr[1:5,1:5] #Values now look different than counts.

##         Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321
## OTU1     -1.084952  -2.181324  -1.413829 -1.8671505  -1.093961
## OTU10     2.667074   2.474521   3.450347  2.9607266   2.633586
## OTU100    1.549500   1.640472   3.100098  2.8224660   2.615852
## OTU1000  -3.370798   1.544424   3.942319  1.5444236   2.930718
## OTU1001   7.237552   6.544405  -0.338748 -0.1559649  -1.330076
## attr(,"class")
## [1] "rmult"
```

## Distance metrics

First calculate the Bray Curtis dissimilarities for the rarefied dataset

```
otu.tab.simple.ss.nozero.bray<-vegdist(otu.tab.simple.ss.nozero, method="bray")
as.matrix(otu.tab.simple.ss.nozero.bray)[1:5,1:5]

##            Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321
## Isa_120223  0.0000000  0.3075788  0.2680245  0.2911224  0.2962143
## Isa_120301  0.3075788  0.0000000  0.2427865  0.2030846  0.2073648
## Isa_120308  0.2680245  0.2427865  0.0000000  0.2611615  0.2900155
## Isa_120320  0.2911224  0.2030846  0.2611615  0.0000000  0.1329053
## Isa_120321  0.2962143  0.2073648  0.2900155  0.1329053  0.0000000
```

Then calculate the Euclidean distance based on the clr data (also known as Aitchison distance)

```
otu.tab.simple.gbm.clr.euclidean<-dist(t(otu.tab.simple.gbm.clr), method = "euclidean
")
as.matrix(otu.tab.simple.gbm.clr.euclidean)[1:5,1:5]
```

```
##             Isa_120223 Isa_120301 Isa_120308 Isa_120320 Isa_120321
## Isa_120223      0.0000   172.3006   181.9140   170.5338   172.2131
## Isa_120301    172.3006     0.0000   178.9638   167.7560   175.3762
## Isa_120308    181.9140   178.9638     0.0000   169.8122   187.4563
## Isa_120320    170.5338   167.7560   169.8122     0.0000   163.9084
## Isa_120321    172.2131   175.3762   187.4563   163.9084     0.0000
```

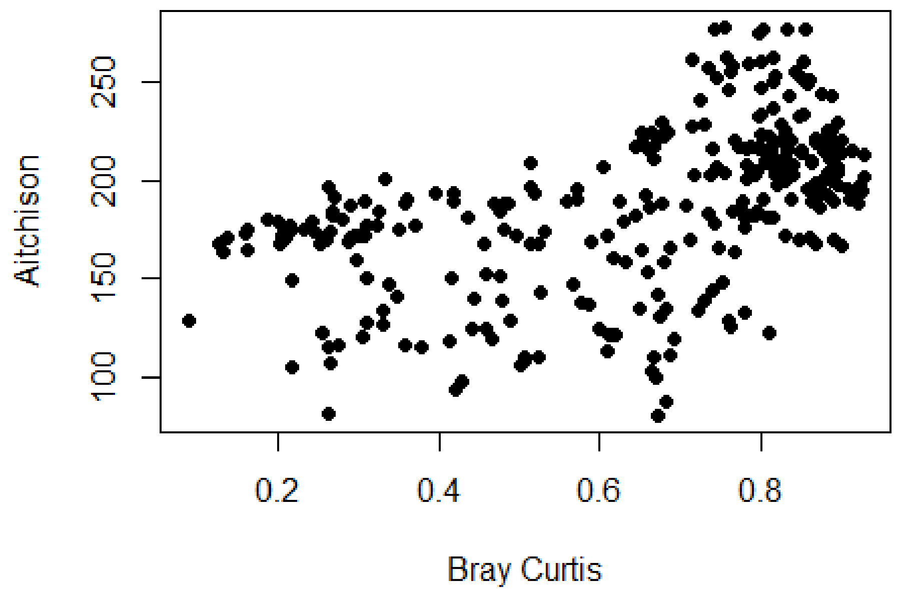
Let's compare the distance matrices:

```
identical(rownames(as.matrix(otu.tab.simple.ss.nozero.bray)),rownames(as.matrix(otu.t
ab.simple.gbm.clr.euclidean)))
```

```
## [1] TRUE
```

Generate a simple x-y plot, and fit the linear model (i.e. the regression)

```
plot(otu.tab.simple.ss.nozero.bray, otu.tab.simple.gbm.clr.euclidean, pch=19, xlab="B
ray Curtis", ylab="Aitchison")
```



```
#lm<-lm(otu.tab.simple.gbm.clr.euclidean~otu.tab.simple.ss.nozero.bray)
#abline(lm, col="red")
```

The correlation between distance matrices is tested with a Mantel test.

```
mantel(otu.tab.simple.ss.nozero.bray, otu.tab.simple.gbm.clr.euclidean)

##
## Mantel statistic based on Pearson's product-moment correlation
##
## Call:
## mantel(xdis = otu.tab.simple.ss.nozero.bray, ydis = otu.tab.simple.gbm.clr.euclide
an)
##
## Mantel statistic r: 0.5112
##       Significance: 0.001
##
## Upper quantiles of permutations (null model):
##    90%    95%  97.5%    99%
## 0.0722 0.1008 0.1386 0.1941
## Permutation: free
## Number of permutations: 999
```

*Phew* That was Part I. Now before you have a break save the data so it can be loaded if you want to use some of the same data.

```
save.image("AB332_lab_I.RData")
```