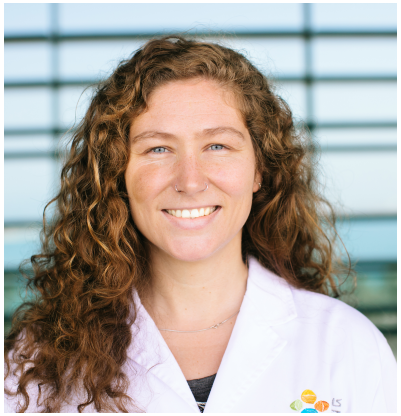
Data analysis in R applied to Marine Science

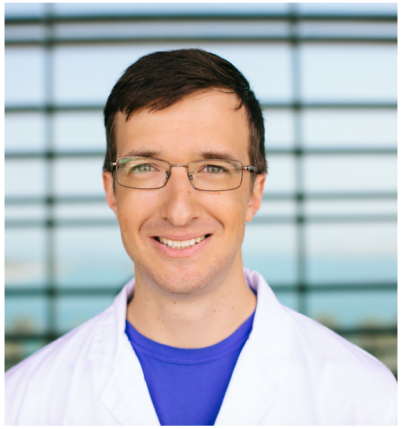
Day 4. Reproduceable workflow (Diversity)
30 January 2019

Tamara Huete-Stauffer - Grégoire Michoud - Daffne López-Sandoval - Malika Kheireddine

Instructors



Tamara Huete-Stauffer
Postdoctoral fellow
Prof. Xelu Morán
Marine microbial ecology
tamara.huetestauffer@kaust.edu.sa



Grégoire Michoud
Postdoctoral fellow
Prof. Daniele Daffonchio
Microbiology of brine pools
gregoire.michoud@kaust.edu.sa

Daffne López-Sandoval
Postdoctoral Fellow
Prof. Susana Agustí
Phytoplankton ecology
daffne.lopezsandoval@kaust.edu.sa

Reproduceable workflow (16S Diversity)

Folder: ~/R_Course_2019/Day4_Diversity/

Script: 16S.R (Create a new one)

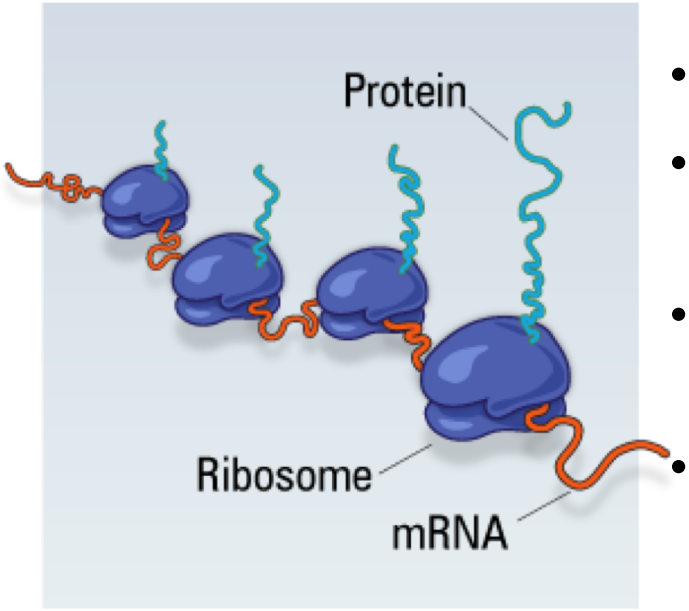
Reproduceable workflow

- Anybody with your data can reach the same result
- Step by step analysis
- Includes all the table modifications to your original data

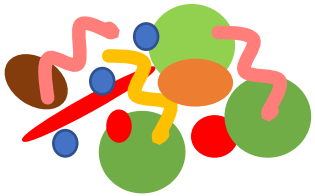
*We will work with 16S diversity data, but many of the tools can be applied to any **dataset with counts:**

- other amplicons (18S, amoA)
- metagenomes/metatrascriptomes
- flow cytometry
- microscopy

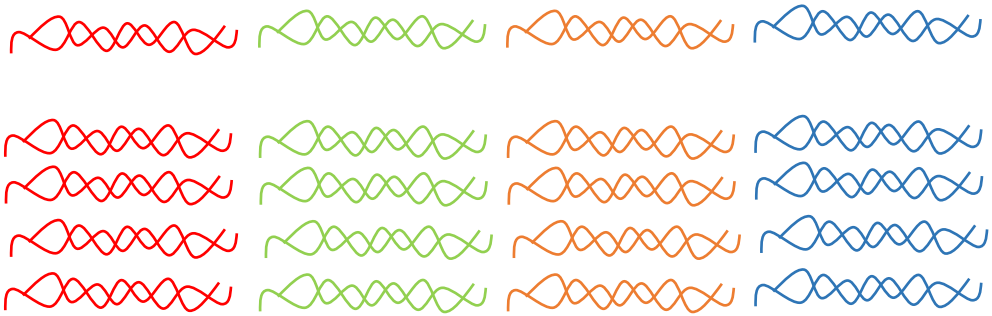
16S diversity



- 16S is the small subunit of the ribosome of prokaryotes
- Ribosomes are responsible for translation (mRNA to proteins)
- Ribosomes are very conserved in all organisms but have some variable regions
- We sequence the variable regions of the 16S gene and we can identify the microbial diversity



Sampling



DNA extraction

16S Amplification

ATCGTAATCC ATCGTAATCC CCGTAAGTCC GGATCGTCAC

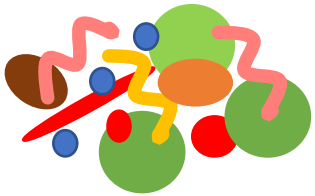
16S Sequencing

Group similar sequences in Operational Taxonomic Units (**OTU**) and count them

OTU 1 OTU 2 OTU 3 OTU 4

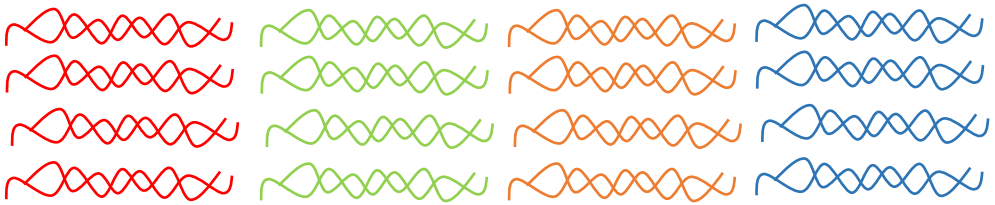
SAR86 Synechococcus SAR11 Roseobacter

Assign taxonomy



Sampling

DNA extraction



16S Amplification

ATCGTAATCC ATCGTAATCC CCGTAAGTCC GGATCGTCAC

16S Sequencing

OTU 1 OTU 2 OTU 3 OTU 4

Group similar sequences in Operational Taxonomic Units (**OTU**) and count them

SAR86 Synechococcus SAR11 Roseobacter

Assign taxonomy

Sample 1

OTU 1 — SAR86

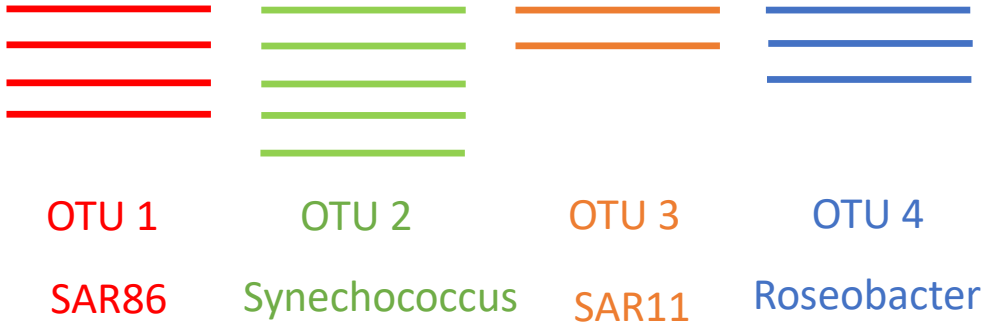
OTU 2 — Synechococcus

OTU 3 — SAR11

OTU 4 — Roseobacter

OTU	Sample 1	Taxonomy
OTU 1	4	SAR86
OTU 2	5	Synechococcus
OTU 3	2	SAR11
OTU 4	3	Roseobacter

Sample 1



OTU	Sample 1	Taxonomy
OTU 1	4	SAR86
OTU 2	5	Synechococcus
OTU 3	2	SAR11
OTU 4	3	Roseobacter

Sample 2

OTU	Sample 1	Sample 2	Taxonomy
OTU 1	4	2	SAR86
OTU 2	5	4	Synechococcus
OTU 3	2	5	SAR11
OTU 4	3	0	Roseobacter
OTU 5	0	3	NS4

16S tables

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

OTU table

- Columns: samples
- Rows: OTUs as rows
- Cells: Counts of OTUs

16S tables

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

	Tax Level 1	Tax Level 2	Tax Level 3	Tax Level 4
OTU 1	Bacteria	Cyanobacteria	Oxyphotobacteria	Synechococcales
OTU 2	Archaea	Thaumarchaeota	Nitrososphaeria	Nitrosopumilales
OTU 3	Bacteria	Alphaproteobacteria	SAR11_clade	Clade I

OTU table

- Columns: samples
- Rows: OTUs as rows
- Cells: Counts of OTUs

Taxonomy table:

- Columns: taxonomy levels
- Rows: OTUs
- Cells: Levels of each OTU

Sometimes they will be together, but we will need to separate them for some analyses

16S tables

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

OTU table

- Columns: samples
- Rows: OTUs as rows
- Cells: Counts of OTUs

	Tax Level 1	Tax Level 2	Tax Level 3	Tax Level 4
OTU 1	Bacteria	Cyanobacteria	Oxyphotobacteria	Synechococcales
OTU 2	Archaea	Thaumarchaeota	Nitrososphaeria	Nitrosopumilales
OTU 3	Bacteria	Alphaproteobacteria	SAR11_clade	Clade I

Taxonomy table:

- Columns: taxonomy levels
- Rows: OTUs
- Cells: Levels of each OTU

	Variable1	Variable 2	Variable 3
Sample 1	22	0.02	39.3
Sample 2	19	0.3	38.5
Sample 3	28	0.6	38.8
Sample 4	31	0.21	39.2

Metadata table:

- Columns: environmental variables
- Rows: samples
- Cells: value of the variable in each sample

Typical steps of 16S analysis

- Sequencing depth
  - Eliminate samples with low reads
  - Rarefy data (optional)
  - Rarefaction curve
- Alpha diversity
- Beta diversity
  - Transform data (optional)
  - Calculate distances
  - Multi-dimensional scaling (ordination plots)
- Relative abundance

Packages for 16S analysis

- **vegan** = classical, good calculations but ugly figures
- **ggvegan** = combines vegan and ggplot2 to help make the vegan plots a little nicer
- **phyloseq** = specific for 16S data, combines calculations and plots. Combines vegan, ggplot2 and other functions: https://joey711.github.io/phyloseq/
- **microbiomeSeq** = Beta version but very promising. Combines vegan, ggplot, phyloseq, and statistical tests http://userweb.eng.gla.ac.uk/umer.ijaz/projects/microbiomeSeq_Tutorial.html

Packages for today

```
library(vegan)
library(ggplot2)
library(ggvegan)
library(RColorBrewer)
```

```
install.packages("reshape2")
library(reshape2)
```

```
install.packages("ade4")
library(ade4)
```

Example dataset

- DNA
- Dissolved organic carbon (DOC)
- Total dissolved nitrogen (TDN)
- Temperature
- Salinity
- pH

1. Loading and preparing data

1.1. Read data

```
microbial16S <- read.csv("16S_OTU_TAX.csv",row.names=1)
```

```
str(microbial16S)
```

```
head(microbial16S)[1:4,1:4]
```

```
      St01_MN St01_SG St01_CR St01_SD
OTU_1    5811    8136    1179       0
OTU_2   43796    6968    1629     167
OTU_3   11141    3621    3302       0
OTU_4   14375    4650    3055      25
```

```
names(microbial16S)
```



```
 [1] "St01_MN" "St01_SG" "St01_CR" "St01_SD"
 [5] "St02_MN" "St02_SG" "St02_CR" "St02_SD"
 [9] "St03_MN" "St03_SG" "St03_CR" "St03_SD"
[13] "St04_MN" "St04_SG" "St04_CR" "St04_SD"
[17] "St05_MN" "St05_SG" "St05_CR" "St05_SD"
[21] "St06_MN" "St06_SG" "St06_CR" "St06_SD"
[25] "St07_MN" "St07_SG" "St07_CR" "St07_SD"
[29] "St08_MN" "St08_SG" "St08_CR" "St08_SD"
[33] "St09_MN" "St09_SG" "St09_CR" "St09_SD"
[37] "St10_SD" "Kingdom" "Phylum"  "Class"
[41] "Order"   "Family"  "Genus"   "Species"
```

1.2. Split microbial table, in OTU and TAX tables

```
OTU <-
```

you know how to do this ☺

```
dataframe[rows,columns]
```

```
names(OTU)
```

```
TAX <-
```

you know how to do this ☺

```
names(TAX)
```

1.3. Read the metadata

```
metadata <- read.csv("16S_metadata.csv",row.names=1)
```

```
names(metadata)
```

1.2. Split microbial table, in OTU and TAX tables

```
OTU <- microbial16S[,1:37]
names(OTU)
```

```
TAX <- microbial16S[,38:44]
names(TAX)
```

1.3. Read the metadata

`row.names()` will make a column in your dataset the row names, can be any column, just specify

```
metadata <- read.csv("16S_metadata.csv",row.names=1)
names(metadata)
```

1.4. Check that all rows and columns match

```
nrow(OTU) ==
row.names(OTU) ==
```

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

	Tax Level 1	Tax Level 2	Tax Level 3	Tax Level 4
OTU 1	Bacteria	Cyanobacteria	Oxyphotobacteria	Synechococcales
OTU 2	Archaea	Thaumarchaeota	Nitrososphaeria	Nitrosopumilales
OTU 3	Bacteria	Alphaproteobacteria	SAR11_clade	Clade I

```
names(OTU) ==
```

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

	Variable1	Variable 2	Variable 3
Sample 1	22	0.02	39.3
Sample 2	19	0.3	38.5
Sample 3	28	0.6	38.8
Sample 4	31	0.21	39.2

1.4. Check that all rows and columns match

```
nrow(OTU) == nrow(TAX)
row.names(OTU) == row.names(TAX)
```

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

	Tax Level 1	Tax Level 2	Tax Level 3	Tax Level 4
OTU 1	Bacteria	Cyanobacteria	Oxyphotobacteria	Synechococcales
OTU 2	Archaea	Thaumarchaeota	Nitrososphaeria	Nitrosopumilales
OTU 3	Bacteria	Alphaproteobacteria	SAR11_clade	Clade I

```
names(OTU) == row.names(metadata)
```

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

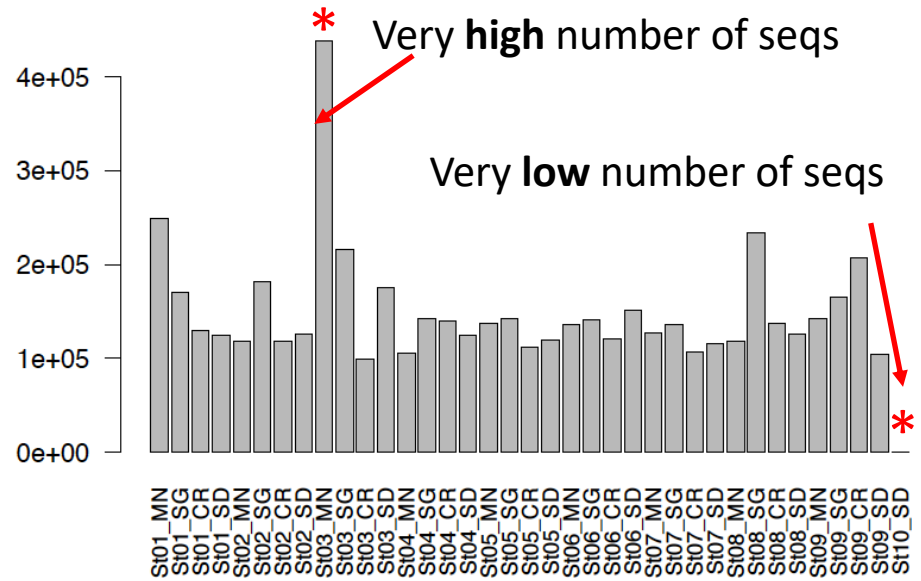
	Variable1	Variable 2	Variable 3
Sample 1	22	0.02	39.3
Sample 2	19	0.3	38.5
Sample 3	28	0.6	38.8
Sample 4	31	0.21	39.2

2. Sequencing depth

2.1 Check the sequences per sample

```
seqs <-
barplot(seqs,cex.names=0.8,las=2)
```

`colSums()`
returns the sum of each column

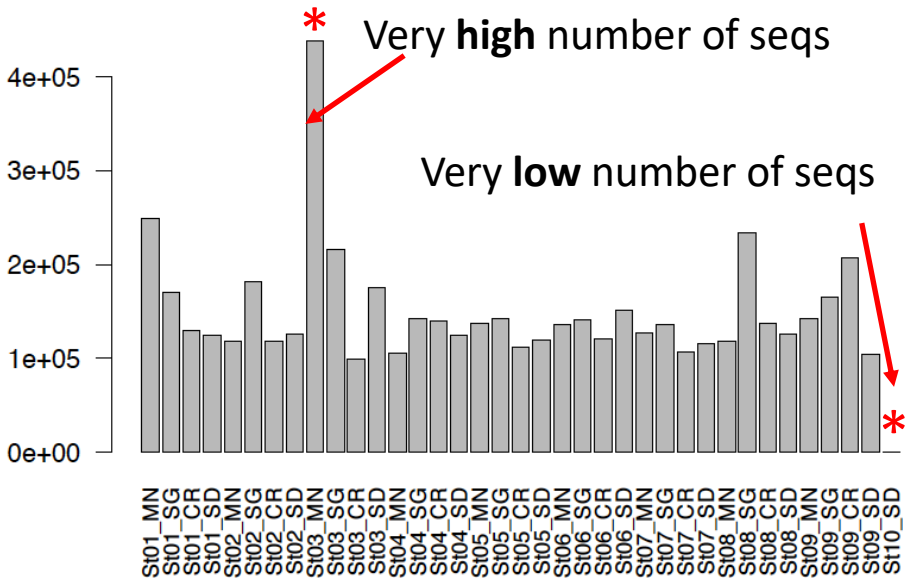


2. Sequencing depth

2.1 Check the sequences per sample

```
seqs <- colSums(OTU)
barplot(seqs,cex.names=0.8,las=2)
```

`colSums()`
returns the sum of each column



2.2 Find samples with low number of X reads (X depends on your samples)

```
colSums(OTU)<10000
```

```
which(colSums(OTU)<10000)
```

```
names(which(colSums(OTU)<10000))
```

`which()` gives the indices for which a condition is TRUE

2.2 Find samples with low number of X reads (X depends on your samples)

```
colSums(OTU)<10000
which(colSums(OTU)<10000)
names(which(colSums(OTU)<10000))
```

`which()` gives the indices for which a condition is TRUE

Delete bad samples

- from the metadata
  1. Find the row name in `metadata` that matches the name of the sample in the OTU table that has low reads: `which() row.names() == names()`
  2. Delete the bad row
- from the OTU table
  1. `dataframe[row, - column]`

2.2 Find samples with low number of X reads (X depends on your samples)

```
colSums(OTU)<10000
which(colSums(OTU)<10000)
names(which(colSums(OTU)<10000))
```

which() gives the indices for which a condition is TRUE

Delete bad samples

- from the metadata

```
bad <- which(row.names(metadata)==names(which(colSums(OTU)<10000)))
metadata <- metadata[-bad,]
```

- from the OTU table

```
OTU <- OTU[,colSums(OTU)>10000]
dim(OTU)
```

2.3. Transpose the table

```
OTUt <- t(sapply(OTU, as.numeric))
```

`t()` transposes the table
`sapply()` applies the function `as.numeric()` to all the elements in OTU

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

	OTU 1	OTU 2	OTU 3
Sample 1	1456	200	299
Sample 2	4900	30	3
Sample 3	2340	50	2000
Sample 4	7	867	1002

```
dim(OTUt)
```

```
OTUt[1:4,1:4]
```

```
colnames(OTUt) <-
```

get the column names for `OTUt` from `OTU`

2.3. Transpose the table

```
OTUt <- t(sapply(OTU, as.numeric))
```

	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	1456	4900	2340	7
OTU 2	200	30	50	867
OTU 3	299	3	2000	1002

	OTU 1	OTU 2	OTU 3
Sample 1	1456	200	299
Sample 2	4900	30	3
Sample 3	2340	50	2000
Sample 4	7	867	1002

`t()` transposes the table
`sapply()` applies the function `as.numeric()` to all the elements in OTU

```
dim(OTUt)
```

```
OTUt[1:4,1:4]
```

```
colnames(OTUt) <- row.names(OTU)
```

2.4. Rarefy samples to a minimum number of reads (Optional)

```
minReads <- min() + rowSums()
minReads
```

find samples with lower number of reads

	OTU 1	OTU 2	OTU 3	
Sample 1	1456	200	299	=1955
Sample 2	4900	30	3	=4933
Sample 3	2340	50	2000	=4390
Sample 4	7	867	1002	=1876

2.4. Rarefy samples to a minimum number of reads (Optional)

```
minReads <- min(rowSums(OTUt))
minReads
```

find samples with lower number of reads

	OTU 1	OTU 2	OTU 3	
Sample 1	1456	200	299	=1955
Sample 2	4900	30	3	=4933
Sample 3	2340	50	2000	=4390
Sample 4	7	867	1002	=1876

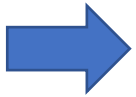
2.4. Rarefy samples to a minimum number of reads (Optional)

```
minReads <- min() + rowSums()
minReads
```

find samples with lower number of reads

	OTU 1	OTU 2	OTU 3	
Sample 1	1456	200	299	=1955
Sample 2	4900	30	3	=4933
Sample 3	2340	50	2000	=4390
Sample 4	7	867	1002	=1876

`rrarefy()`



	OTU 1	OTU 2	OTU 3
Sample 1	1397	190	289
Sample 2	1860	15	0
Sample 3	1007	21	848
Sample 4	7	867	1002

```
set.seed(1)
OTUt_rarefy<-rrarefy(OTUt, minReads)
rowSums(OTUt_rarefy)
```

standardize all samples to the same number of reads

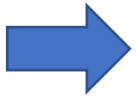
2.4. Rarefy samples to a minimum number of reads (Optional)

```
minReads <- min() + rowSums()
minReads
```

find samples with lower number of reads

	OTU 1	OTU 2	OTU 3	
Sample 1	1456	200	299	=1955
Sample 2	4900	30	3	=4933
Sample 3	2340	50	2000	=4390
Sample 4	7	867	1002	=1876

`rrarefy()`



	OTU 1	OTU 2	OTU 3
Sample 1	1397	190	289
Sample 2	1860	15	0
Sample 3	1007	21	848
Sample 4	7	867	1002

```
set.seed(1)
OTUt_rarefy<-rrarefy(OTUt, minReads)
rowSums(OTUt_rarefy)
```

standardize all samples to the same number of reads

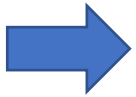
2.4. Rarefy samples to a minimum number of reads (Optional)

```
minReads <- min() + rowSums()
minReads
```

find samples with lower number of reads

	OTU 1	OTU 2	OTU 3	
Sample 1	1456	200	299	=1955
Sample 2	4900	30	3	=4933
Sample 3	2340	50	2000	=4390
Sample 4	7	867	1002	=1876

rrarefy()



	OTU 1	OTU 2	OTU 3
Sample 1	1397	190	289
Sample 2	1860	15	0
Sample 3	1007	21	848
Sample 4	7	867	1002

```
set.seed(1)
OTUt_rarefy<-rrarefy(OTUt, minReads)
rowSums(OTUt_rarefy)
```

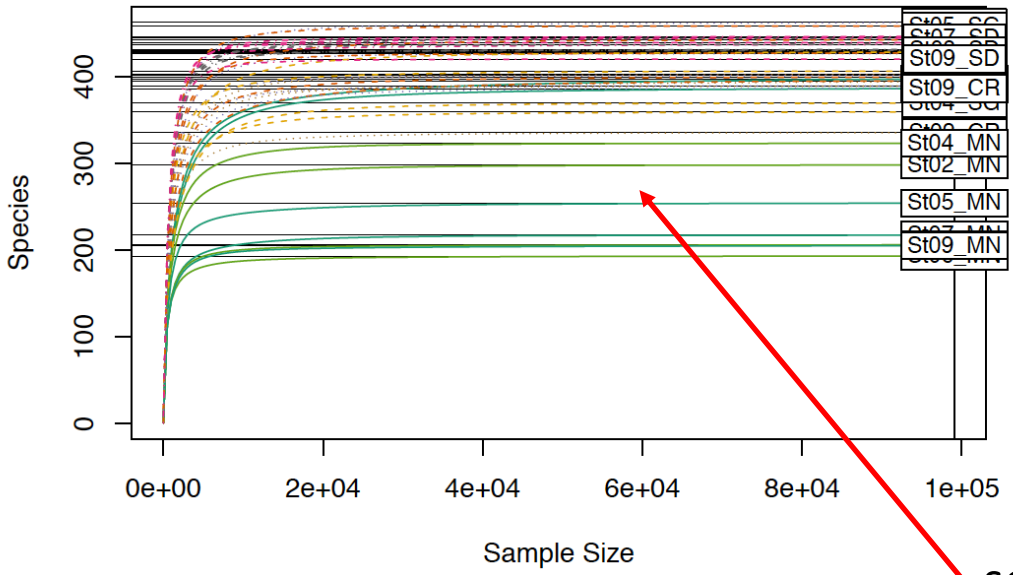
standardize all samples to the same number of reads

```
OTUt_rarefy_nozero <- OTUt_rarefy[,colSums(OTUt_rarefy)>0]
dim(OTUt_rarefy_nozero)
```

eliminate empty OTUs (the sum of the whole column is now zero)

2.5. Plot the rarefaction curve

```
rarecurve(OTUt_rarefy_nozero,step=500,sample=minReads,
col=brewer.pal(n = 8, name = "Dark2"),lty=c(1:4),label=TRUE)
```



sequencing more does not increase the number of species

3. Alpha diversity

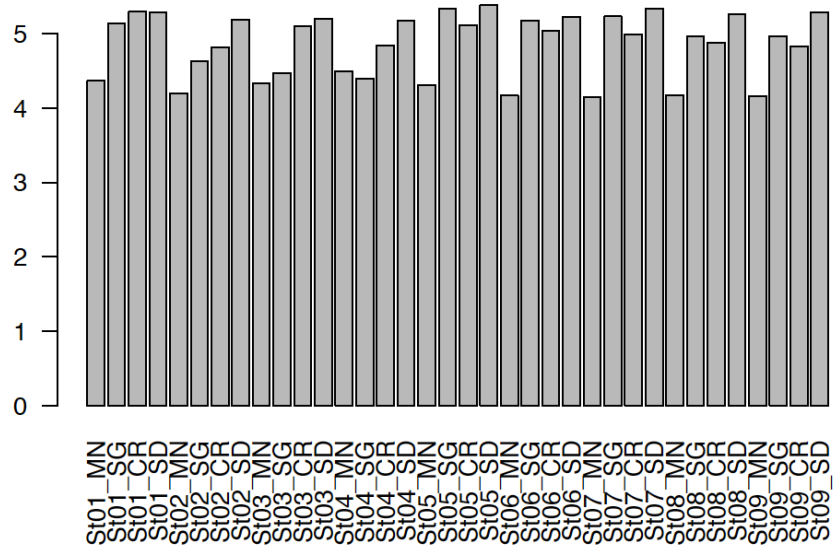
- Shannon, Simpson
  - Richness = number of species per sample
  - Evenness = how evenly are the species distributed

```
alpha <- diversity(OTUt_rarefy_nozero, index="shannon")
```

3. Alpha diversity

- Shannon, Simpson
  - Richness = number of species per sample
  - Evenness = how evenly are the species distributed

```
alpha <- diversity(OTUt_rarefy_nozero, index="shannon")
```



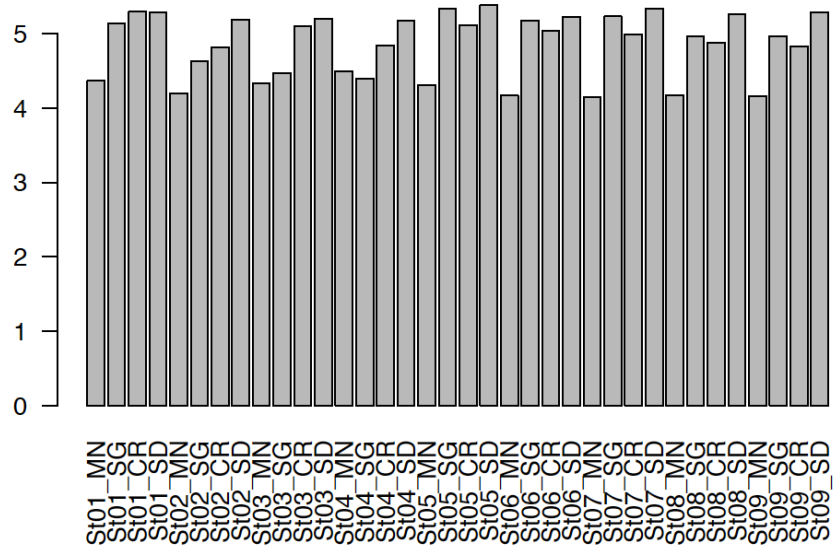
Shannon index of each sample separately

```
barplot()
```

3. Alpha diversity

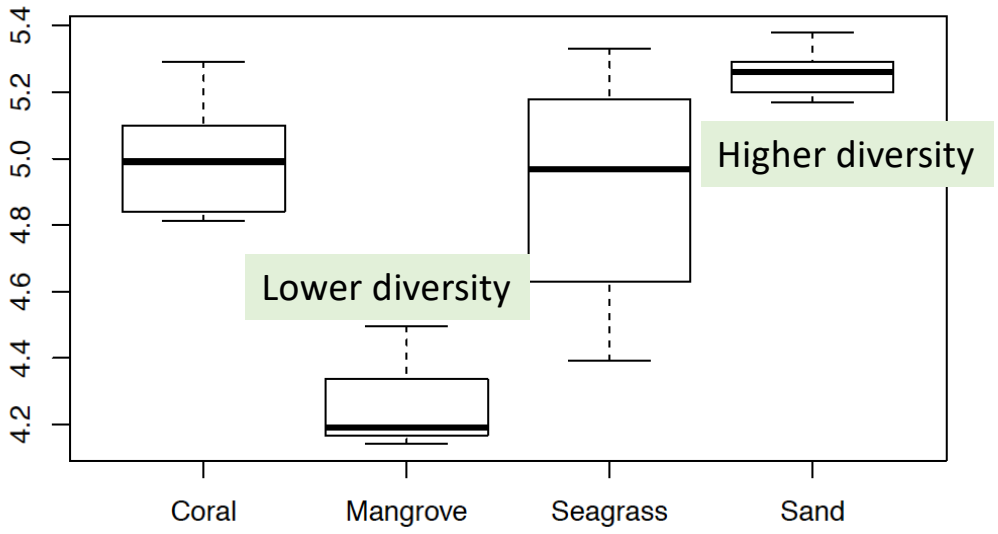
- Shannon, Simpson
  - Richness = number of species per sample
  - Evenness = how evenly are the species distributed

```
alpha <- diversity(OTUt_rarefy_nozero, index="shannon")
```



Shannon index of each sample separately

```
barplot(alpha, las=2)
```



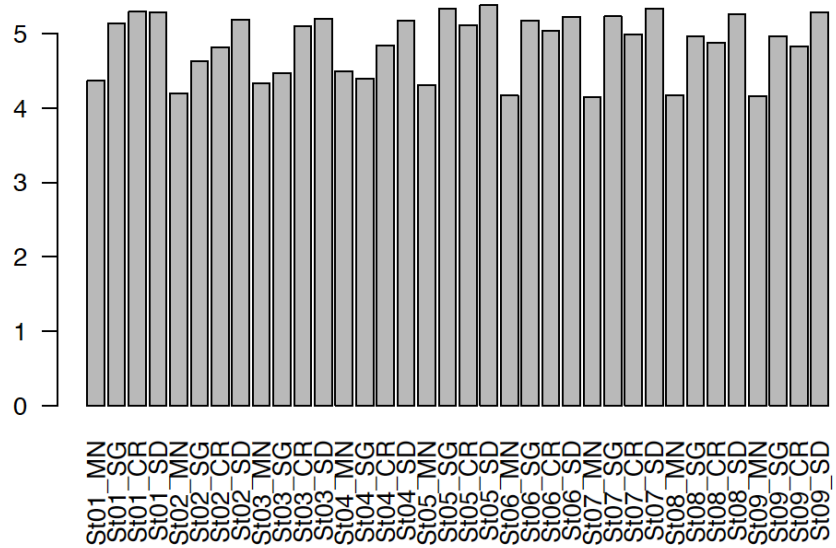
Shannon index of samples grouped by a metadata variable

```
boxplot()
```

3. Alpha diversity

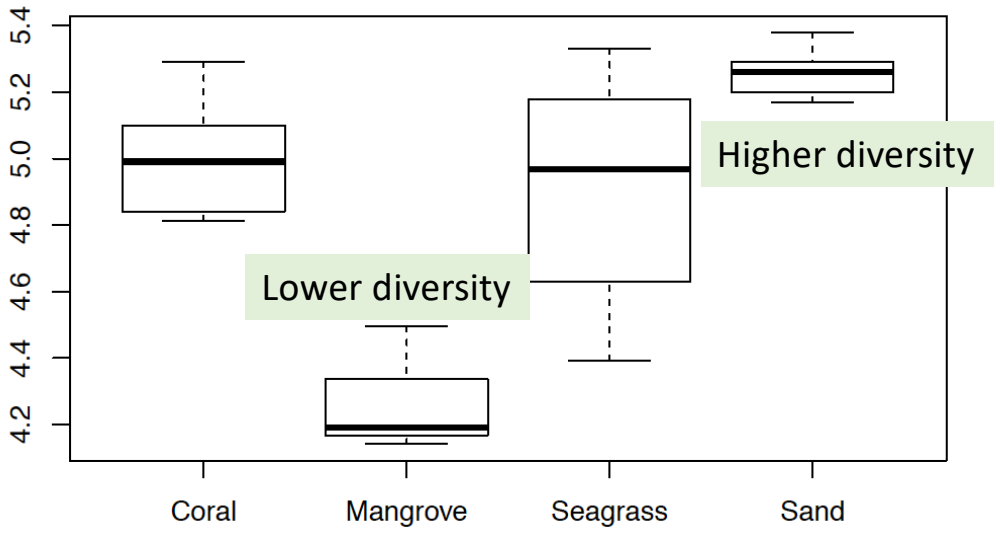
- Shannon, Simpson
  - Richness = number of species per sample
  - Evenness = how evenly are the species distributed

```
alpha <- diversity(OTUt_rarefy_nozero, index="shannon")
```



Shannon index of each sample separately

```
barplot(alpha, las=2)
```



Shannon index of samples grouped by a metadata variable

```
boxplot(alpha ~  metadata$habitat)
```

Exercise

1. Calculate the alpha diversity using `diversity()` and simpson's index
2. Do a boxplot of the simpson index according to habitat ordered as Coral, Mangrove, Seagrass, Sand

Exercise

1. Calculate the alpha diversity using `diversity()` and simpson's index
2. Do a boxplot of the simpson index according to habitat ordered as Coral, Mangrove, Seagrass, Sand

```
alphaS <- diversity(OTUt_rarefy_nozero, index="simpson")
```

```
metadata$habitat <-
factor(metadata$habitat,levels=c("Coral","Mangrove","Seagrass","Sand"))
```

```
boxplot(alphaS ~  metadata$habitat)
```

Interpret the results, is it the same as the previous?

Range 0 - 1

0: No diversity

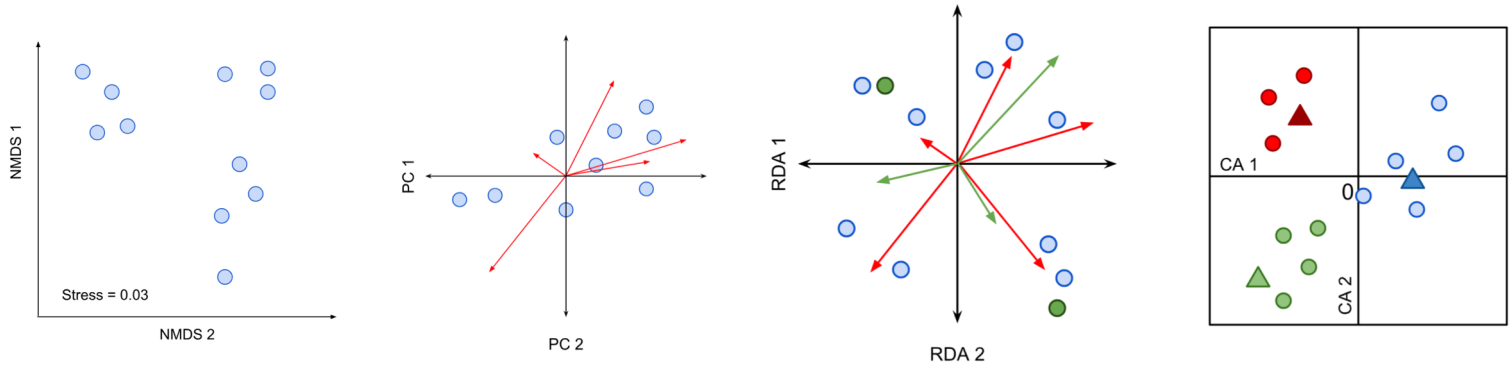
1: Infinite diversity

4. Beta diversity

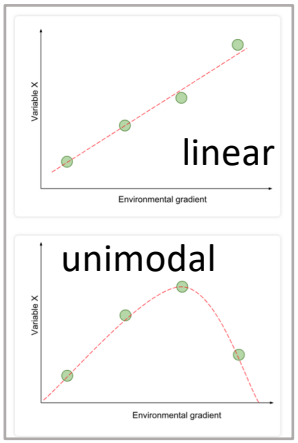
Differences between samples

- log transform data (Optional)
- Distance index (Bray-Curtis, Manhattan, Euclidean...)
- Check statistical differences between groups
- Visualize with multidimensional scaling methods (MDS)

  https://mb3is.megx.net/gustame
- Find environmental variables that explain the distribution



MULTIDIMENSIONAL SCALING METHODS



EXPLORATION:
Sites/species

ENVIRONMENTAL INTERPRETATION:
Sites/species
Sites/Variables

Distance

Raw data

Cluster

PCoA

NMDS

PCA
Euclidean
linear

CA
Chi-squared
unimodal

NMDS

RDA
Constrained PCA
linear

CCA
Constrained CA
unimodal

4.1 Transform data

- Count data (OTU table) `log10(x + 1)`
  - Avoid bias towards very abundant OTUs

```
logOTUt_rarefy_nozero <- log10(OTUt_rarefy_nozero + 1)
```

4.1 Transform data

- Count data (OTU table) `log10(x + 1)`
  - Avoid bias towards very abundant OTUs

```
logOTUt_rarefy_nozero <- log10(OTUt_rarefy_nozero + 1)
```

- Environmental variables (metadata) `scale()`
  - Avoid bias towards variables with bigger values
  - Only continuous variables

Z-score = (Value – Mean)/SD

```
metadata_scale <-
```
create a new dataframe with the scaled values only of the continuous variables

```
metadata_scale_all <-
```
Use `cbind ()` to put together the scaled continuous variables and the habitat column from the metadata table

4.1 Transform data

- Count data (OTU table) `log10(x + 1)`
  - Avoid bias towards very abundant OTUs

```
logOTUt_rarefy_nozero <- log10(OTUt_rarefy_nozero + 1)
```

- Environmental variables (metadata) `scale()`
  - Avoid bias towards variables with bigger values
  - Only continuous variables

Z-score = (Value – Mean)/SD

```
metadata_scale <-as.data.frame(scale(metadata[-1])) # only continuous variables
metadata_scale_all <-
```

Use `cbind ()` to put together the scaled continuous variables and the habitat column from the metadata table

4.1 Transform data

- Count data (OTU table) `log10(x + 1)`
  - Avoid bias towards very abundant OTUs

```
logOTUt_rarefy_nozero <- log10(OTUt_rarefy_nozero + 1)
```

- Environmental variables (metadata) `scale()`
  - Avoid bias towards variables with bigger values
  - Only continuous variables

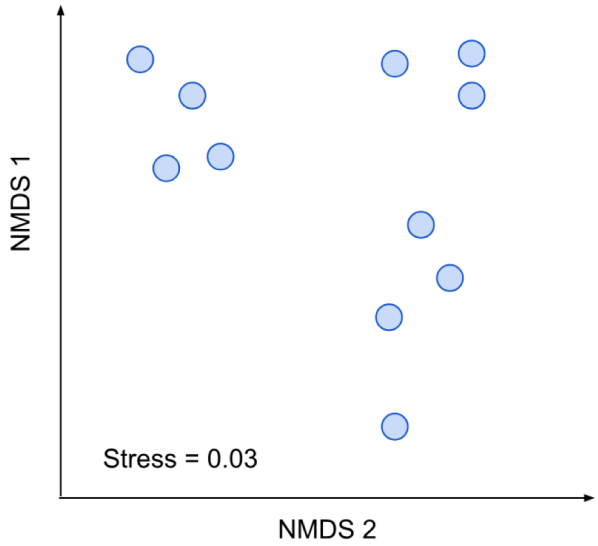
Z-score = (Value – Mean)/SD

```
metadata_scale <-as.data.frame(scale(metadata[-1])) # only continuous variables
metadata_scale_all <- cbind(habitat=metadata$habitat,metadata_scale)
```

4.2 Nonmetric Multidimensional scaling (NMDS)

- Aim: Find patterns among samples/sites from a distance matrix
- Plots shows **ranked distances** between objects distributed non linearly on a 2 dimension plot
- Closer objects are more similar
- Stress: over 0 and below 0.3
- Axes are arbitrary

  - Dots: Samples/sites (rows)
  - Crosses: Species/OTU (columns)
  - Arrows: Environmental parameters



will make a random calculation always the same

```
set.seed(100)
logOTUt.nmds <- metaMDS(logOTUt_rarefy_nozero, distance="bray")
logOTUt.nmds$stress
```

Calculates the distance matrix and NMDS

Stress should be > 0 and < 0.3

will make a random calculation always the same

```
set.seed(100)
logOTUt.nmds <- metaMDS(logOTUt_rarefy_nozero, distance="bray")
logOTUt.nmds$stress
```

Calculates the distance matrix and NMDS

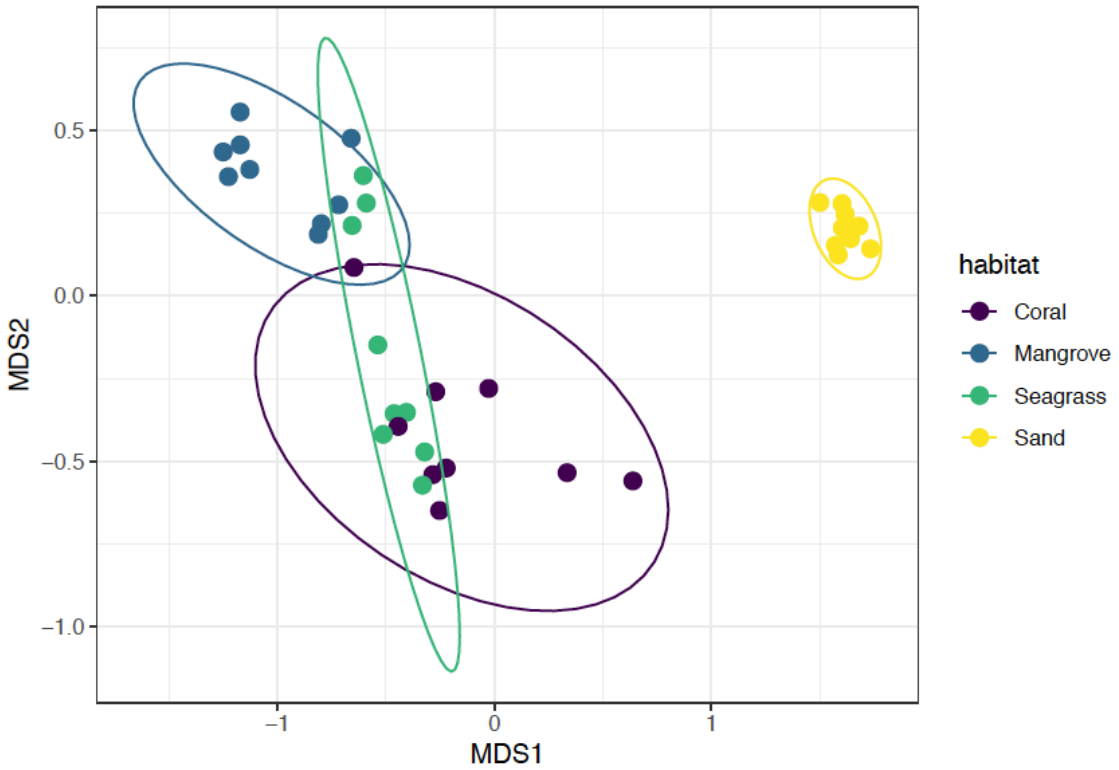
Stress should be > 0 and < 0.3

```
nmds.p <- as.data.frame(logOTUt.nmds$points)
```

access point coordinates from NMDS object

```
row.names(logOTUt.nmds$points)==row.names(metadata)
nmds.p.m <- cbind(nmds.p,metadata)
```

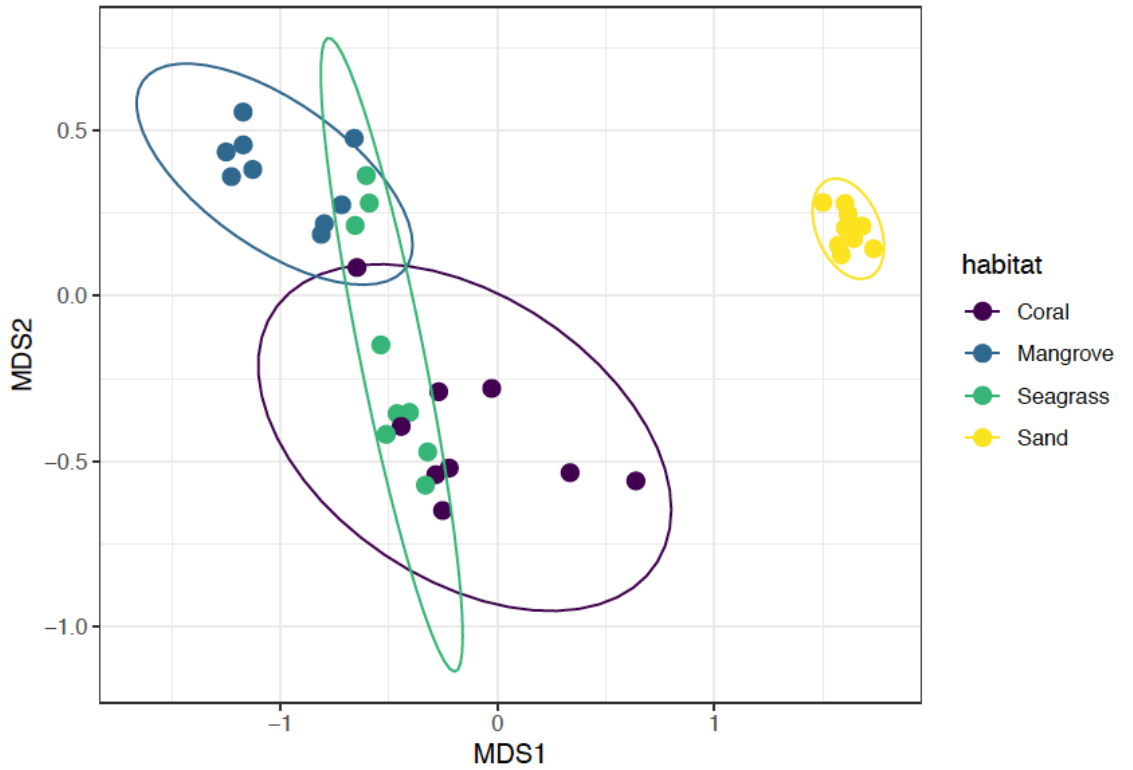
`cbind()` NMDS coordinates with metadata (Check that samples are in the same order as metadata)



Exercise

Try to get this NMDS plot using the data **`nmds.p.m`**, and the geom `stat_ellipse()`

*Remember to order your factor habitat



Exercise

Try to get this NMDS plot using the data **nmds.p.m**, and the geom `stat_ellipse()`

*Remember to order your factor habitat

reorder habitat levels

```
nmds.p.m$habitat <-
ordered(nmds.p.m$habitat,levels=c("Coral","Mangrove","Seagrass","Sand"))
```

```
nmds.ggplot <- ggplot(nmds.p.m, aes(x = MDS1, y = MDS2,color= habitat))+
  geom_point(size = 3)+
  stat_ellipse(level = 0.95)+
  theme_bw()
```

get nmds plot with 95% confidence level

will make a random calculation always the same

```
set.seed(100)
logOTUt.nmds <- metaMDS(logOTUt_rarefy_nozero, distance="bray")
logOTUt.nmds$stress
```

Calculates the distance matrix and NMDS

Stress should be > 0 and < 0.3

```
nmds.p <- as.data.frame(logOTUt.nmds$points)
```

access point coordinates from NMDS object

```
row.names(logOTUt.nmds$points)==row.names(metadata)
nmds.p.m <- cbind(nmds.p,metadata)
```

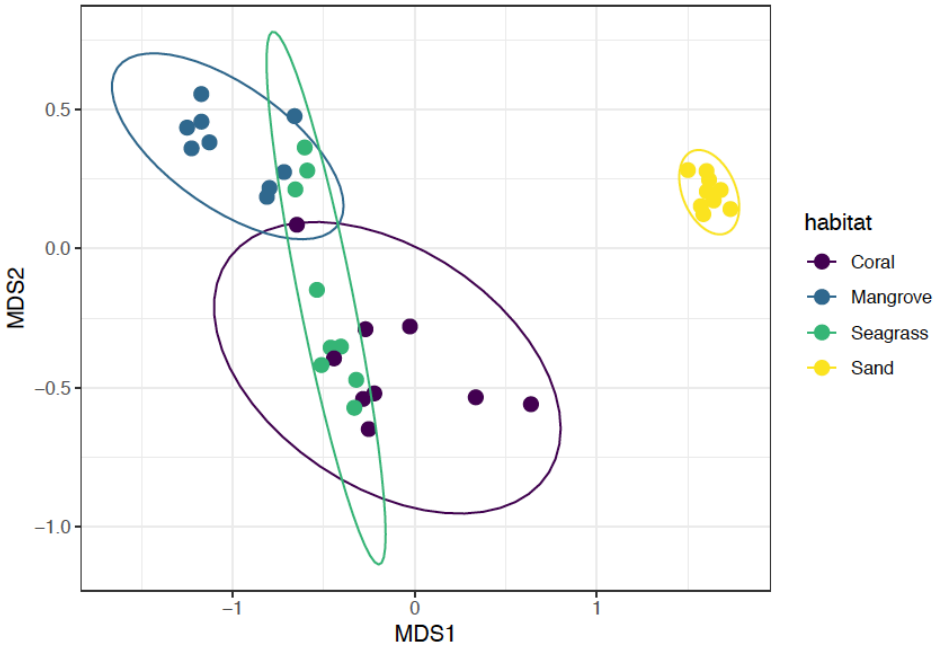
cbind() NMDS coordinates with metadata (Check that samples are in the same order as metadata)

```
nmds.p.m$habitat <-
ordered(nmds.p.m$habitat,levels=c("Coral","Mangrove","Seagrass","Sand"))
```

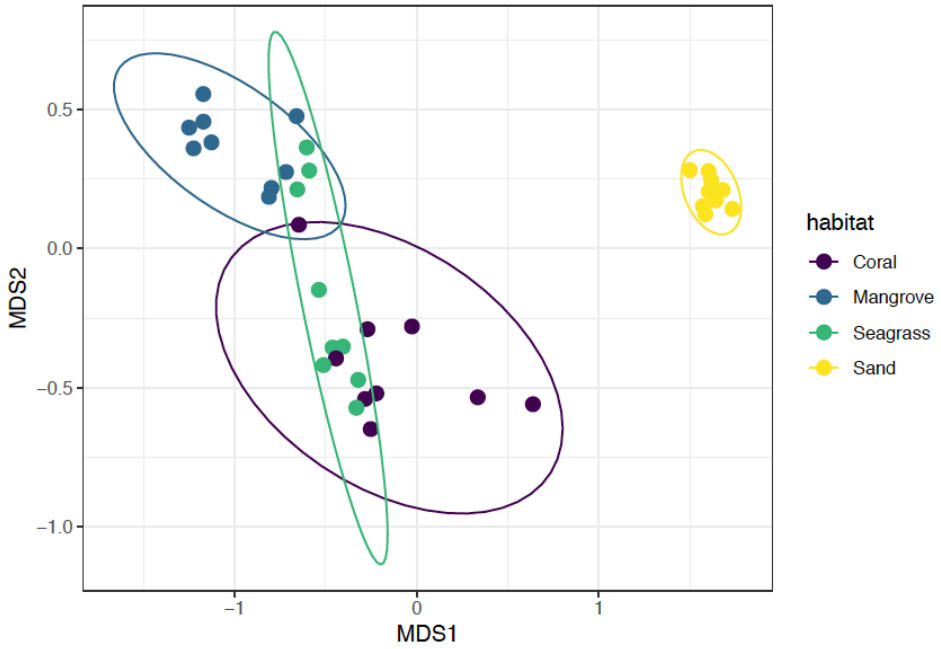
reorder habitat levels

```
nmds.ggplot <- ggplot(nmds.p.m, aes(x = MDS1, y = MDS2,color= habitat))+
  geom_point(size = 3)+
  stat_ellipse(level = 0.95)+
  theme_bw()
```

get nmds plot with 95% confidence level



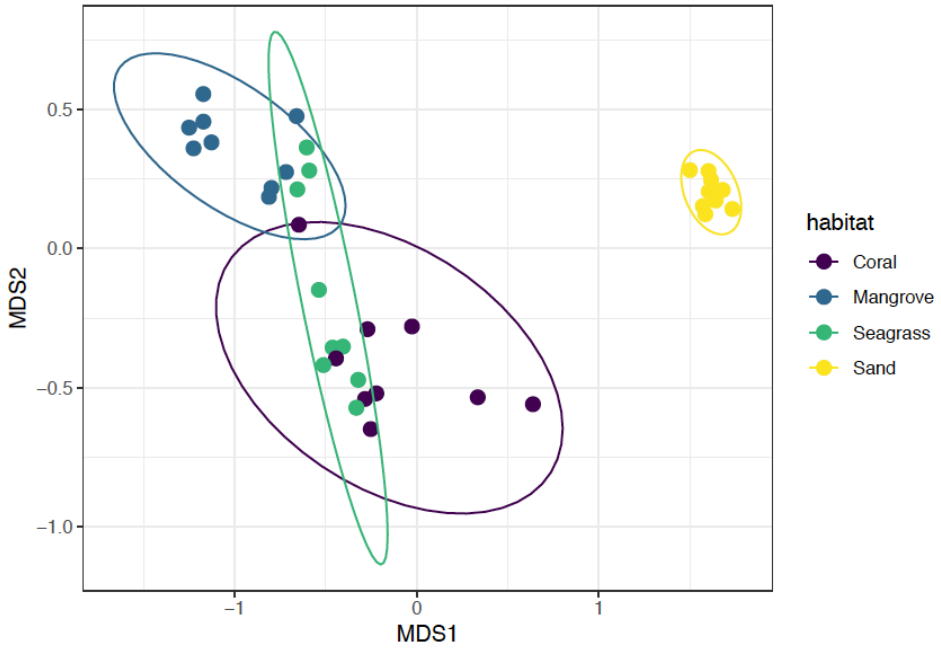
Are the differences between habitats significant?



Are the differences between habitats significant?

```
library(devtools)
install_github("pmartinezarbizu/pairwiseAdonis/pairwiseAdonis")
library(pairwiseAdonis)
```

new package from github



Are the differences between habitats significant?

```
library(devtools)
install_github("pmartinezarbizu/pairwiseAdonis/pairwiseAdonis")
library(pairwiseAdonis)
```

new package from github

```
adonis(logOTUt_rarefy_nozero  ~ metadata$habitat)
```

are differences in habitat significant?

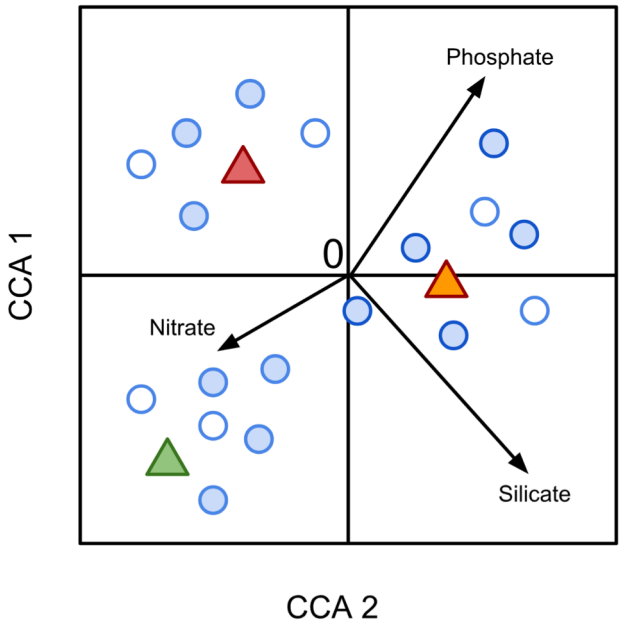
```
pairwise.adonis(logOTUt_rarefy_nozero,metadata$habitat)
```

Pairwise comparisons of habitat

4.3 Canonical/Constrained Correspondence Analysis (CCA)

- Aim: Describe count data reducing the number of dimensions. Find the maximum **correspondence** between sites and species.
- Uses $\mathcal{X}^2$ (Chi-square) distances and the linear combination of exploratory variables for the axes

- Best when:
  1. Species differ among sites (Count data or presence-absence)
  2. Species show unimodal response to environmental gradients
- Species in the center are closest to their optimum and are present in most samples, species on the edges are rare

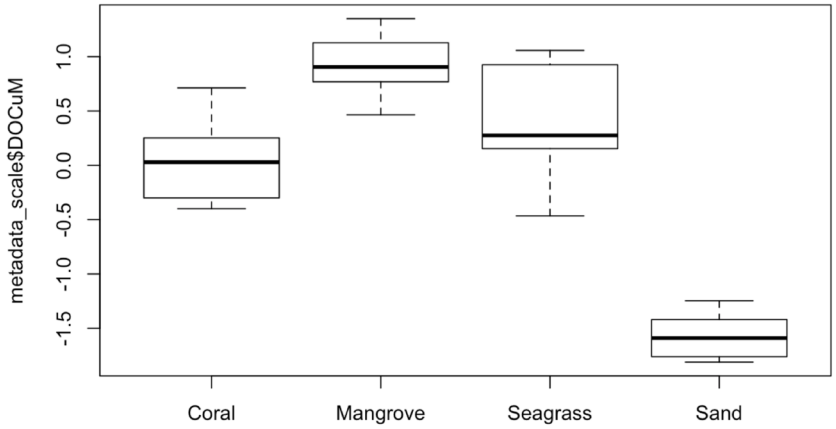


*What to do with NAs in your data?

- Few missing values
- Low variability
- Predictable behavior

```
complete.cases(metadata_scale) # row 8 has missing value
is.na(metadata_scale) # St02_SD in the DOCuM column
```

```
plot(metadata_scale$DOCuM ~ metadata$habitat)
```

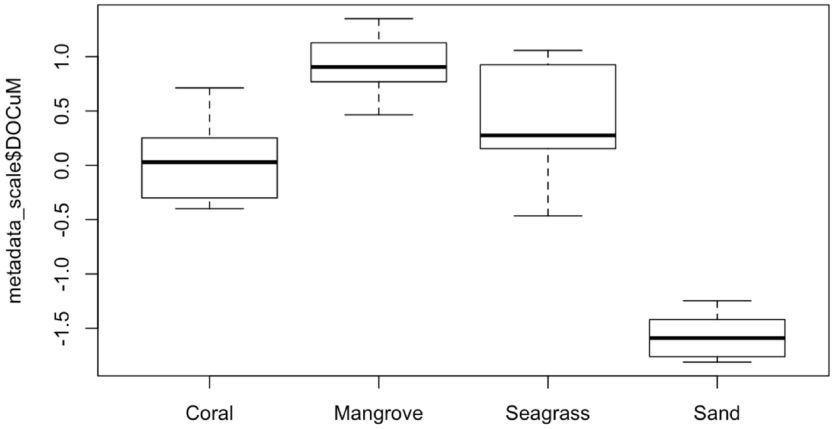


*What to do with NAs in your data?

- Few missing values
- Low variability
- Predictable behavior

```
complete.cases(metadata_scale) # row 8 has missing value
is.na(metadata_scale) # St02_SD in the DOCuM column
```

```
plot(metadata_scale$DOCuM ~ metadata$habitat)
```

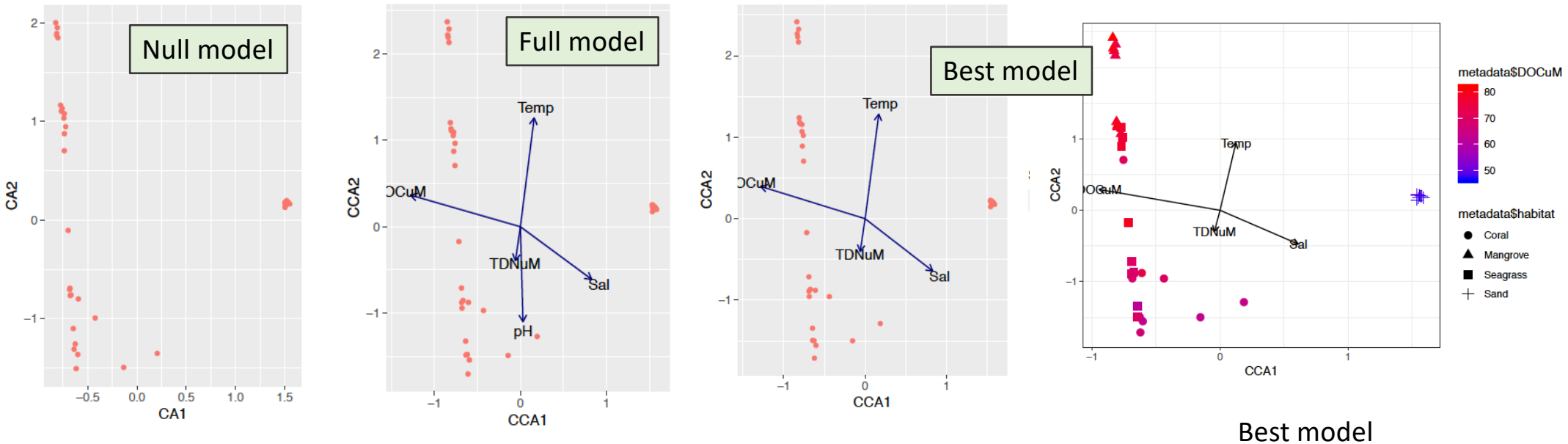



```
mean(metadata_scale[metadata$habitat=="Sand",]$
DOCuM,na.rm=TRUE) #-1.57
```

```
sd(metadata_scale[metadata$habitat == "Sand",]$
DOCuM,na.rm=TRUE) #0.21
```

Replace NA with mean value

```
metadata_scale[which(is.na(metadata_scale$DOCuM)==TRUE),]$DOCuM <-
mean(metadata_scale[metadata$habitat=="Sand",]$DOCuM,na.rm=TRUE)
```



Best model

Automatic model selection:

1. Null Model = no environmental variables
2. Full Model = all the environmental variables
3. Stepwise model = add 1 by 1 the variables and compare to the Null and Full models to decide which explain better the data

```
mod0 <- cca(logOTUt_rarefy_nozero ~ 1, metadata_scale)
mod0
anova(mod0)
autoplot(mod0, layers = c("sites", "biplot"))
```

Null model

No environmental variables

```r
mod0 <- cca(logOTUt_rarefy_nozero ~ 1, metadata_scale)
mod0
anova(mod0)
autoplot(mod0, layers = c("sites", "biplot"))
```

Null model

No environmental variables

```
## Call: cca(formula = logOTUt_rarefy_nozero ~ 1, data =
## metadata_scale, na.action = "na.omit")
##
##                Inertia Rank
## Total            1.643
## Unconstrained    1.643   35
## Inertia is scaled Chi-square
##
## Eigenvalues for unconstrained axes:
##    CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
## 0.8415 0.3445 0.1056 0.0831 0.0314 0.0239 0.0201 0.0180
## (Showing 8 of 35 unconstrained eigenvalues)
```

All Inertia is unconstrained , we don't have any variables to explain it

```r
mod0 <- cca(logOTUt_rarefy_nozero ~ 1, metadata_scale)
mod0
anova(mod0)
autoplot(mod0, layers = c("sites", "biplot"))
```

Null model

No environmental variables

```r
mod1 <- cca(logOTUt_rarefy_nozero ~ ., metadata_scale)
mod1
anova(mod1)
autoplot(mod1, layers = c("sites", "biplot"))
```

Full model

All environmental variables

```
## Call: cca(formula = logOTUt_rarefy_nozero ~ DOCuM + TDNuM + Temp +
## Sal + pH, data = metadata_scale, na.action = "na.omit")
##
##               Inertia Proportion Rank
## Total          1.6431     1.0000
## Constrained    1.2068     0.7345    5
## Unconstrained  0.4363     0.2655   30
## Inertia is scaled Chi-square
##
## Eigenvalues for constrained axes:
##   CCA1   CCA2   CCA3   CCA4   CCA5
## 0.8044 0.2734 0.0937 0.0258 0.0096
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
## 0.11275 0.07325 0.03789 0.02033 0.01962 0.01708 0.01530 0.01414
## (Showing 8 of 30 unconstrained eigenvalues)
```

We can explain up to 73% of the Inertia with all our variables

```
mod0 <- cca(logOTUt_rarefy_nozero ~ 1, metadata_scale)
mod0
anova(mod0)
autoplot(mod0, layers = c("sites", "biplot"))
```

Null model
No environmental variables

```
mod1 <- cca(logOTUt_rarefy_nozero ~ ., metadata_scale)
mod1
anova(mod1)
autoplot(mod1, layers = c("sites", "biplot"))
```

Full model
All environmental variables

```
mod.p <- ordistep(mod0, scope=formula(mod1),direction
= "both", steps = 1000)
mod.p
anova(mod.p)
autoplot(mod.p, layers = c("sites", "biplot"))
```

Best model
step by step addition of environmental variables

* `ordiR2step()` best model based on R^2 instead on p-value

Best model

mod.p

```
## Call: cca(formula = logOTUt_rarefy_nozero ~ DOCuM + Temp + TDNuM +
## Sal, data = metadata_scale, na.action = "na.omit")
##
##                Inertia Proportion Rank
## Total           1.6431     1.0000
## Constrained     1.1836     0.7204    4
## Unconstrained   0.4595     0.2796   31
## Inertia is scaled Chi-square
##
## Eigenvalues for constrained axes:
##    CCA1    CCA2    CCA3    CCA4
## 0.8008 0.2651 0.0932 0.0245
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
## 0.11939 0.07325 0.04235 0.02304 0.01984 0.01929 0.01535 0.01432
## (Showing 8 of 31 unconstrained eigenvalues)
```

Inertia explained by the environmental variables (72%)

Residual Inertia (not explained) (23%)

Eigenvalues are a fraction of the total constrained inertia
CCA1: 0.8/1.18 = **0.68**
CCA2: 0.26/1.18 = **0.22**

constrained axes are linear combinations of the explanatory variables

```
smy <- summary(mod.p)
```

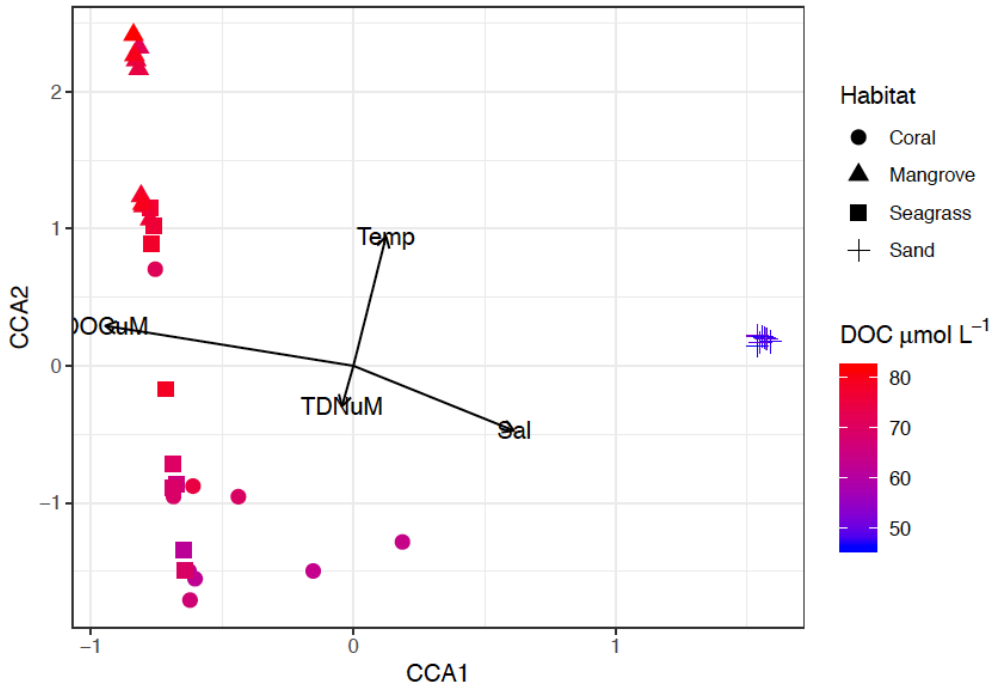
Access the data inside the object **mod.p**

```
df1 <- as.data.frame(smy$sites[,1:2])
```

Get the coordinates of the samples on the first 2 axes of the CCA

```
df2 <- as.data.frame(smy$biplot[,1:2])
```

Get the coordinates of the environmental variables



Exercise

Try to do this plot with **df1** and **df2**

- Use `geom_segment()` for the arrows of the environmental data
- Use `geom_text()` for the labels of the arrows of the environmental data
- Don't forget the legend!

```
p <- ggplot(df1, aes(x = CCA1, y = CCA2))+
  geom_point( )+
  geom_segment(data = df2, ) +
  geom_text(data = df2, ) +
  scale_color_gradient() +
  labs() +
  theme_bw()
p
```

```
p <- ggplot(df1, aes(x = CCA1, y = CCA2))+

  geom_point(size = 3,aes(color= metadata$DOCuM, shape =
metadata$habitat))+

  geom_segment(data = df2, aes(x=0, y = 0, xend = CCA1,
yend= CCA2), arrow = arrow(length = unit(0.02, "npc")))+

  geom_text(data = df2, aes(x = CCA1, y = CCA2, label =
row.names(df2))) +

  scale_color_gradient(low='blue',high='red') +

  labs(color=expression(paste("DOC ",mu,"mol ",L^-
1)),shape="Habitat")+

  theme_bw()

p
```

5. Relative abundance

	OTU 1	OTU 2	OTU 3	**Sum**
Sample 1	1456	200	299	= 1955
Sample 2	4900	30	3	= 4933
Sample 3	2340	50	2000	= 4390
Sample 4	7	867	1002	= 1876

**If the table is rarefied all the rowSums should give the same number

5. Relative abundance

	OTU 1	OTU 2	OTU 3	Sum
Sample 1	1456	200	299	= 1955
Sample 2	4900	30	3	= 4933
Sample 3	2340	50	2000	= 4390
Sample 4	7	867	1002	= 1876

**If the table is rarefied all the rowSums should give the same number

```
sums <- rowSums()
```

5. Relative abundance

	OTU 1	OTU 2	OTU 3	Sum
Sample 1	1456	200	299	= 1955
Sample 2	4900	30	3	= 4933
Sample 3	2340	50	2000	= 4390
Sample 4	7	867	1002	= 1876

**If the table is rarefied all the rowSums should give the same number

```
sums <- rowSums(OTUt_rarefy_nozero)
```

5. Relative abundance

	OTU 1	OTU 2	OTU 3	Sum
Sample 1	1456	200	299	= 1955
Sample 2	4900	30	3	= 4933
Sample 3	2340	50	2000	= 4390
Sample 4	7	867	1002	= 1876

**If the table is rarefied all the rowSums should give the same number

$$\text{Rel Ab} = \frac{\text{OTU counts}}{\text{Total}}$$

OTU 1 = 1456/1955 = 0.75
OTU 2 = 200/1955 = 0.1
OTU 3 = 299/1955 = 0.15

```
sums <- rowSums(OTUt_rarefy_nozero)
```

5. Relative abundance

	OTU 1	OTU 2	OTU 3	Sum
Sample 1	1456	200	299	= 1955
Sample 2	4900	30	3	= 4933
Sample 3	2340	50	2000	= 4390
Sample 4	7	867	1002	= 1876

**If the table is rarefied all the rowSums should give the same number

$$\text{Rel Ab} = \frac{\text{OTU counts}}{\text{Total}}$$

OTU 1 = 1456/1955 = 0.75
OTU 2 = 200/1955 = 0.1
OTU 3 = 299/1955 = 0.15

```
sums <- rowSums(OTUt_rarefy_nozero)
relABt <- sweep(OTUt_rarefy_nozero, MARGIN=1,sums ,"/")
```

5. Relative abundance

	OTU 1	OTU 2	OTU 3	Sum
Sample 1	1456	200	299	= 1955
Sample 2	4900	30	3	= 4933
Sample 3	2340	50	2000	= 4390
Sample 4	7	867	1002	= 1876

**If the table is rarefied all the rowSums should give the same number

$$\text{Rel Ab} = \frac{\text{OTU counts}}{\text{Total}}$$

OTU 1 = 1456/1955 = 0.75
OTU 2 = 200/1955 = 0.1
OTU 3 = 299/1955 = 0.15

The sum of all the relative abundances in one sample is 1
0.75+0.1+0.15=1

```
sums <- rowSums(OTUt_rarefy_nozero)
relABt <- sweep(OTUt_rarefy_nozero, MARGIN=1,sums ,"/")
```

5. Relative abundance

	OTU 1	OTU 2	OTU 3	Sum
Sample 1	1456	200	299	= 1955
Sample 2	4900	30	3	= 4933
Sample 3	2340	50	2000	= 4390
Sample 4	7	867	1002	= 1876

**If the table is rarefied all the rowSums should give the same number

$$\text{Rel Ab} = \frac{\text{OTU counts}}{\text{Total}}$$

OTU 1 = 1456/1955 = 0.75
OTU 2 = 200/1955 = 0.1
OTU 3 = 299/1955 = 0.15

The sum of all the relative abundances in one sample is 1
0.75+0.1+0.15=1

```
sums <- rowSums(OTUt_rarefy_nozero)
relABt <- sweep(OTUt_rarefy_nozero, MARGIN=1,sums ,"/")
relABt <- relABt*100
rowSums(relABt)
```

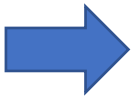
Get 0-100 range instead of 0-1

Merge relAB + TAX + metadata

1. Transpose relAb back to the original format (samples as columns)

```
relAB <- t(relABt)
relAB[1:4,1:4]
names(relAB)
row.names(relAB)
```

	OTU 1	OTU 2	OTU 3
Sample 1	75	10	15
Sample 2	60	30	20
Sample 3	35	5	15
Sample 4	2	15	80



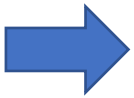
	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	75	60	35	2
OTU 2	10	30	5	15
OTU 3	15	20	60	80

Merge relAB + TAX + metadata

1. Transpose relAb back to the original format (samples as columns)

```
relAB <- t(relABt)
relAB[1:4,1:4]
names(relAB) <- row.names(relABt)
row.names(relAB) <- names(relABt)
```

	OTU 1	OTU 2	OTU 3
Sample 1	75	10	15
Sample 2	60	30	20
Sample 3	35	5	15
Sample 4	2	15	80



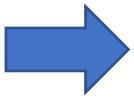
	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	75	60	35	2
OTU 2	10	30	5	15
OTU 3	15	20	60	80

Merge relAB + TAX + metadata

1. Transpose relAb back to the original format (samples as columns)

```
relAB <- t(relABt)
relAB[1:4,1:4]
names(relAB) <- row.names(relABt)
row.names(relAB) <- names(relABt)
```

	OTU 1	OTU 2	OTU 3
Sample 1	75	10	15
Sample 2	60	30	20
Sample 3	35	5	15
Sample 4	2	15	80



	Sample 1	Sample 2	Sample 3	Sample 4
OTU 1	75	60	35	2
OTU 2	10	30	5	15
OTU 3	15	20	60	80

2. cbind() relAB table and TAX (only if they have the same number of rows, in same order)

```
nrow(relAb)==nrow(TAX)
rAbTAX <- cbind(relAB,TAX)
rAbTAX$OTU <- row.names(rAbTAX)
```

Create also a new column with the OTU names because we will loose them in the next step

	Sample 1	Sample 2	Sample 3	Sample 4	Tax Level 1	Tax Level 2	OTU
OTU 1	75	60	35	2	Bacteria	Cyanobacteria	OTU 1
OTU 2	10	30	5	15	Archaea	Thaumarchaeota	OTU2
OTU 3	15	20	60	80	Bacteria	Alphaproteobacteria	OTU 3

Merge relAB + TAX + metadata

3. Covert table to long format

```
long.rAbTAX <- melt(rAbTAX,id=c("OTU", "Kingdom", "Phylum", "Class", "Order",
"Family", "Genus", "Species" ))
head(long.rAbTAX)
```

relAb + TAX
Long format

	Sample 1	Sample 2	Sample 3	Sample 4	Tax Level 1	Tax Level 2	OTU
OTU 1	75	60	35	2	Bacteria	Cyanobacteria	OTU 1
OTU 2	10	30	5	15	Archaea	Thaumarchaeota	OTU2
OTU 3	15	20	60	80	Bacteria	Alphaproteobacteria	OTU 3

	Variable	Tax Level 1	Tax Level 2	OTU	Value
1	Sample 1	Bacteria	Cyanobacteria	OTU 1	75
2	Sample 1	Archaea	Thaumarchaeota	OTU2	10
3	Sample 1	Bacteria	Alphaproteobacteria	OTU 3	15
4	Sample 2	Bacteria	Cyanobacteria	OTU 1	60
5	Sample 2	Archaea	Thaumarchaeota	OTU2	30
6	Sample 2	Bacteria	Alphaproteobacteria	OTU 3	20

Merge relAB + TAX + metadata

3. Covert table to long format

```
long.rAbTAX <- melt(rAbTAX,id=c("OTU", "Kingdom", "Phylum", "Class", "Order",
"Family", "Genus", "Species" ))
head(long.rAbTAX)
```

relAb + TAX
Long format

	Sample 1	Sample 2	Sample 3	Sample 4	Tax Level 1	Tax Level 2	OTU
OTU 1	75	60	35	2	Bacteria	Cyanobacteria	OTU 1
OTU 2	10	30	5	15	Archaea	Thaumarchaeota	OTU2
OTU 3	15	20	60	80	Bacteria	Alphaproteobacteria	OTU 3

	Variable	Tax Level 1	Tax Level 2	OTU	Value
1	Sample 1	Bacteria	Cyanobacteria	OTU 1	75
2	Sample 1	Archaea	Thaumarchaeota	OTU2	10
3	Sample 1	Bacteria	Alphaproteobacteria	OTU 3	15
4	Sample 2	Bacteria	Cyanobacteria	OTU 1	60
5	Sample 2	Archaea	Thaumarchaeota	OTU2	30
6	Sample 2	Bacteria	Alphaproteobacteria	OTU 3	20

```
names(long.rAbTAX)[9] <- "station" #variable
names(long.rAbTAX)[10] <- "relAb"   #value
```

Rename value and variable to something understandable

Merge relAB + TAX + metadata

3. Covert table to long format

```
long.rAbTAX <- melt(rAbTAX,id=c("OTU", "Kingdom", "Phylum", "Class", "Order",
"Family", "Genus", "Species" ))
head(long.rAbTAX)
```

relAb + TAX
Long format

	Sample 1	Sample 2	Sample 3	Sample 4	Tax Level 1	Tax Level 2	OTU
OTU 1	75	60	35	2	Bacteria	Cyanobacteria	OTU 1
OTU 2	10	30	5	15	Archaea	Thaumarchaeota	OTU2
OTU 3	15	20	60	80	Bacteria	Alphaproteobacteria	OTU 3

	Variable	Tax Level 1	Tax Level 2	OTU	Value
1	Sample 1	Bacteria	Cyanobacteria	OTU 1	75
2	Sample 1	Archaea	Thaumarchaeota	OTU2	10
3	Sample 1	Bacteria	Alphaproteobacteria	OTU 3	15
4	Sample 2	Bacteria	Cyanobacteria	OTU 1	60
5	Sample 2	Archaea	Thaumarchaeota	OTU2	30
6	Sample 2	Bacteria	Alphaproteobacteria	OTU 3	20

```
names(long.rAbTAX)[9] <- "station" #variable
names(long.rAbTAX)[10] <- "relAb"   #value
```

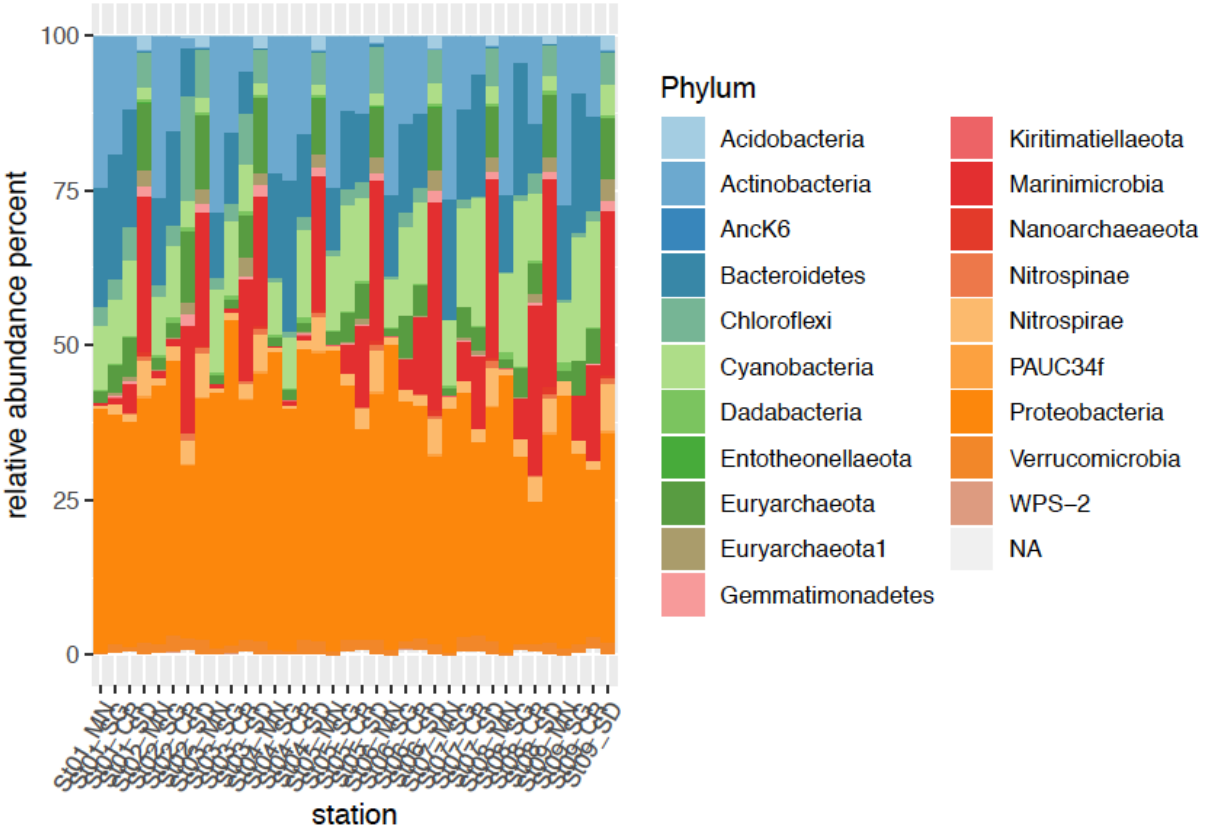
Rename value and variable to something understandable

4. Merge with metadata by value of a column (**same in both datasets**)

relAb +TAX + metadata
Long format

```
metadata$Label <- row.names(metadata)
long.all <- merge(x=long.rAbTAX, y=metadata, by.x= "station", by.y= "Label" )
```

Exercise



Make a barplot of the relative abundance of each sample filled by Phylum

```
geom_bar()
```

```
ggplot(data=long.all, aes(x=station,y=relAb, fill=Phylum)) +
    geom_bar(stat="identity",position="stack") +
    guides() +
    ylab() +
    scale_fill_manual() +
    theme()
  )
```

```
ggplot(data=long.all, aes(x=station,y=relAb, fill=Phylum)) +
    geom_bar(stat="identity",position="stack") +
    guides(fill=guide_legend(title="Phylum"),colour=FALSE) +
    ylab(label= "relative abundance percent") +
    scale_fill_manual(values =
colorRampPalette(brewer.pal(9,"Paired"))(length(unique(long.al
l$Phylum)))) +
    theme(axis.text.x = element_text(angle = 60, hjust = 1)
  )
```

Get the top 10 most abundant Phyla

You can do this for any Taxonomic level!

1. Make the column relAb numeric

```
long.all$relAb <- as.numeric(long.all$relAb)
```

Get the top 10 most abundant Phyla

You can do this for any Taxonomic level!

1. Make the column relAb numeric

```
long.all$relAb <- as.numeric(long.all$relAb)
```

2. Delete values where the relAb is 0, because it will mess up the mean we have to calculate later

```
long.all <- long.all[-as.numeric(row.names(long.all[long.all$relAb==0,])),]
```

Get the top 10 most abundant Phyla

You can do this for any Taxonomic level!

1. Make the column relAb numeric

```
long.all$relAb <- as.numeric(long.all$relAb)
```

2. Delete values where the relAb is 0, because it will mess up the mean we have to calculate later

```
long.all <- long.all[-as.numeric(row.names(long.all[long.all$relAb==0,])),]
```

3. Get the sum of the relative abundance per Phylum and Sample

```
sumPhyla <- tapply(long.all$relAb,
list(long.all$Phylum,long.all$station),sum,na.rm=TRUE)
```

	station	Phylum	relAb
1	Sample 1	Cyanobacteria	75
2	Sample 1	Cyanobacteria	10
3	Sample 1	Thaumarchaeota	15
4	Sample 2	Cyanobacteria	60
5	Sample 2	Cyanobacteria	30
6	Sample 2	Thaumarchaeota	20

Get the top 10 most abundant Phyla

You can do this for any Taxonomic level!

1. Make the column relAb numeric

```
long.all$relAb <- as.numeric(long.all$relAb)
```

2. Delete values where the relAb is 0, because it will mess up the mean we have to calculate later

```
long.all <- long.all[-as.numeric(row.names(long.all[long.all$relAb==0,])),]
```

3. Get the sum of the relative abundance per Phylum and Sample

```
sumPhyla <- tapply(long.all$relAb,
list(long.all$Phylum,long.all$station),sum,na.rm=TRUE)
```

	station	Phylum	relAb
1	Sample 1	Cyanobacteria	75
2	Sample 1	Cyanobacteria	10
3	Sample 1	Thaumarchaeota	15
4	Sample 2	Cyanobacteria	60
5	Sample 2	Cyanobacteria	30
6	Sample 2	Thaumarchaeota	20

Get the top 10 most abundant Phyla

You can do this for any Taxonomic level!

1. Make the column relAb numeric

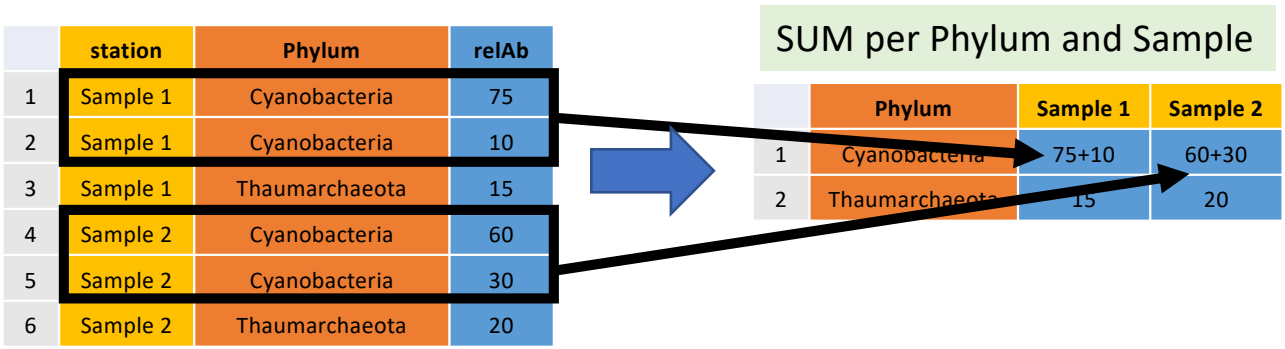
```
long.all$relAb <- as.numeric(long.all$relAb)
```

2. Delete values where the relAb is 0, because it will mess up the mean we have to calculate later

```
long.all <- long.all[-as.numeric(row.names(long.all[long.all$relAb==0,])),]
```

3. Get the sum of the relative abundance per Phylum and Sample

```
sumPhyla <- tapply(long.all$relAb,
list(long.all$Phylum,long.all$station),sum,na.rm=TRUE)
```



Get the top 10 most abundant Phyla

You can do this for any Taxonomic level!

1. Make the column relAb numeric

```
long.all$relAb <- as.numeric(long.all$relAb)
```

2. Delete values where the relAb is 0, because it will mess up the mean we have to calculate later

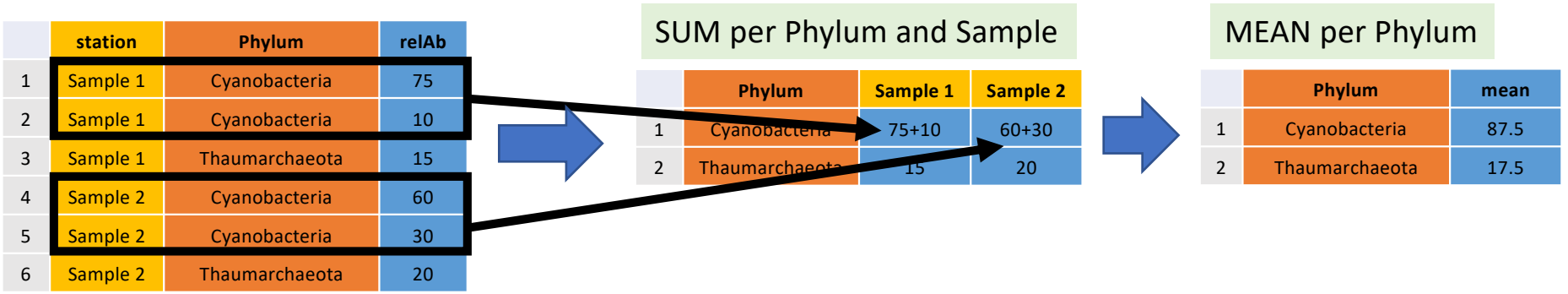
```
long.all <- long.all[-as.numeric(row.names(long.all[long.all$relAb==0,])),]
```

3. Get the sum of the relative abundance per Phylum and Sample

```
sumPhyla <- tapply(long.all$relAb,
list(long.all$Phylum,long.all$station),sum,na.rm=TRUE)
```

4. Get the mean per Phylum

```
msumPhyla <- rowMeans(sumPhyla,na.rm=TRUE)
```



5. Order Phyla starting with the ones that are more abundant

```
top10 <- names(msumPhyla[order(msumPhyla,decreasing=TRUE)[1:10]])
```

	Phylum	mean
1	Cyanobacteria	15
2	Thaumarchaeota	5
3	Proteobacteria	30
4	Chloroflexi	3

	Phylum	mean
1	Proteobacteria	30
2	Cyanobacteria	15
3	Thaumarchaeota	5
4	Chloroflexi	3

Order from most abundant to least abundant and choose the first 10 names

5. Order Phyla starting with the ones that are more abundant

```
top10 <- names(msumPhyla[order(msumPhyla,decreasing=TRUE)[1:10]])
```

	Phylum	mean
1	Cyanobacteria	15
2	Thaumarchaeota	5
3	Proteobacteria	30
4	Chloroflexi	3

	Phylum	mean
1	Proteobacteria	30
2	Cyanobacteria	15
3	Thaumarchaeota	5
4	Chloroflexi	3

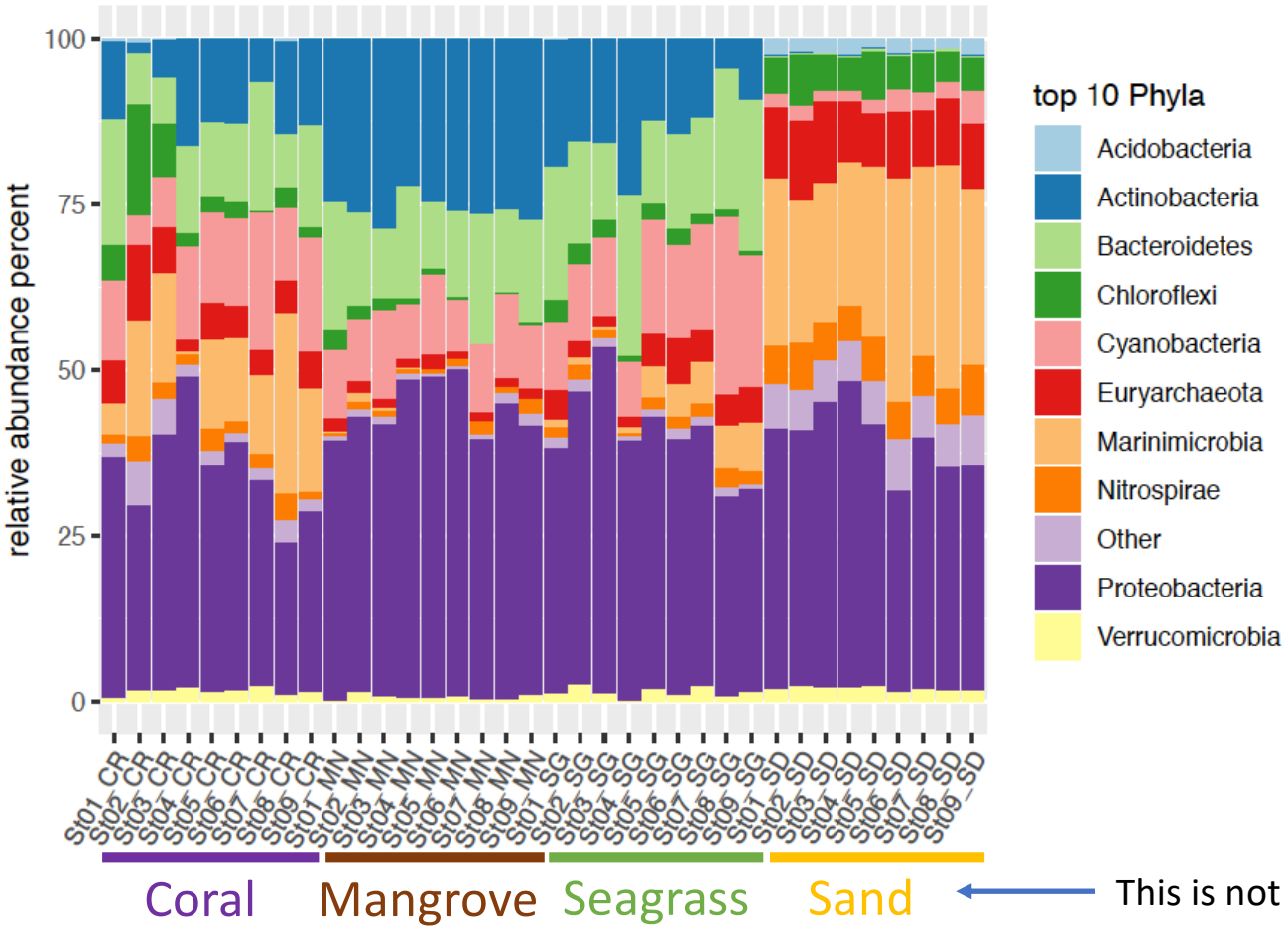
Order from most abundant to least abundant and choose the first 10 names

6. Back to our long table, keep top 10 and mark rest as “Other”

```
long.all$newPhyla <- ifelse(long.all$Phylum %in% top10,
as.character(long.all$Phylum), "Other")
```

	station	Phylum	relAb	newPhyla
1	Sample 1	Cyanobacteria	75	Cyanobacteria
2	Sample 1	Cyanobacteria	10	Cyanobacteria
	....	.....	....	.....
15	Sample 5	Dadabacteria	3	Other
16	Sample 5	Cyanobacteria	30	Cyanobacteria
17	Sample 5	Thaumarchaeota	20	Thaumarchaeota

*ifelse(condition,TRUE,FALSE)



This is not from ggplot, don't go nuts!

Exercise

Barplot of the 10 most abundant Phyla (*Other* includes all the rest) by station and habitat.

Try to get this one!
tip: order the factors habitat and station before

```
long.all$station <- factor(long.all$station,
levels=unique(long.all$station[order(long.all$habitat)]), ordered=TRUE)
```

```
long.all$habitat <- ordered(long.all$habitat,
levels=c("Coral","Mangrove","Seagrass","Sand"))

long.all$station <- factor(long.all$station,
levels=unique(long.all$station[order(long.all$habitat)]),
ordered=TRUE)
```

Order the levels of the factors first habitat and variable (Station)

```
ggplot(data=long.all, aes(x=station,y=relAb,
fill=newPhyla))+
    geom_bar(stat="identity",position="stack")+
    guides(fill=guide_legend(title="top 10
Phyla"),colour=FALSE)+
    ylab(label= "relative abundance percent")+
    scale_fill_brewer(palette = 'Paired') +
    theme(axis.text.x = element_text(angle = 60, hjust = 1)
```