Homework 1

Dmitry Panteleev

2024-04-03

## Homework 1

1. Make EDA for data (5 points).
2. Build an ordination of objects using NMDS methods (descriptions, samples, etc.) (5 points).
3. Visualise the relationship between the resulting ordination and environmental parameters with functions envfit() and ordisufr()(5 points).
4. Draw conclusions about the most important factors (5 points).

\*Data sources\*\* Trees on Barro Colorado (data from Condit et al. (2002), ‘BCI’ data, ‘vegan’ package). Backtick: 4

library(tidyr)

## Warning: пакет 'tidyr' был собран под R версии 4.3.3

library(dplyr)

##   
## Присоединяю пакет: 'dplyr'

## Следующие объекты скрыты от 'package:stats':  
##   
## filter, lag

## Следующие объекты скрыты от 'package:base':  
##   
## intersect, setdiff, setequal, union

library(broom)  
  
library(tidyverse)

## Warning: пакет 'tidyverse' был собран под R версии 4.3.3

## Warning: пакет 'ggplot2' был собран под R версии 4.3.3

## Warning: пакет 'readr' был собран под R версии 4.3.3

## Warning: пакет 'forcats' был собран под R версии 4.3.3

## Warning: пакет 'lubridate' был собран под R версии 4.3.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ forcats 1.0.0 ✔ readr 2.1.5  
## ✔ ggplot2 3.5.0 ✔ stringr 1.5.1  
## ✔ lubridate 1.9.3 ✔ tibble 3.2.1  
## ✔ purrr 1.0.2

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ggmap)

## Warning: пакет 'ggmap' был собран под R версии 4.3.3

## ℹ Google's Terms of Service: <https://mapsplatform.google.com>  
## Stadia Maps' Terms of Service: <https://stadiamaps.com/terms-of-service/>  
## OpenStreetMap's Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles/>  
## ℹ Please cite ggmap if you use it! Use `citation("ggmap")` for details.

library(ggvegan)

## Загрузка требуемого пакета: vegan

## Warning: пакет 'vegan' был собран под R версии 4.3.3

## Загрузка требуемого пакета: permute  
## Загрузка требуемого пакета: lattice  
## This is vegan 2.6-4

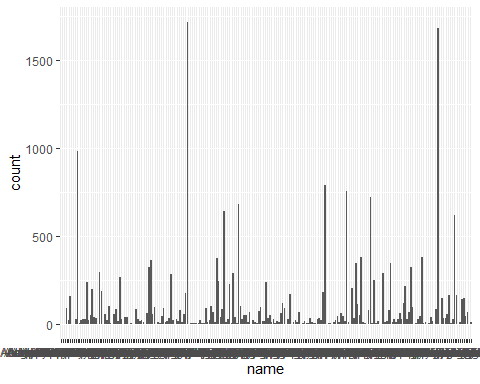
library(vegan)

Download the data of “Trees on Barro Colorado”

data(BCI.env)  
data(BCI)

EDA

name <- -1  
count <- -1  
  
for (i in colnames(BCI))  
{  
 sum\_col <- sum(BCI[i])  
 name <- c(name, i)  
 count <- c(count, sum\_col)  
 #sums <- sum(BCI)  
}  
  
name <- name[2:length(name)]  
count <- count[2:length(count)]  
df\_count <- data.frame(name,count)  
  
ggplot(data=df\_count, aes(x=name, y=count)) +  
 geom\_col()



sort(table(as.character(df\_count$count)))

##   
## 100 101 111 118 121 143 147 149 156 163 164 167 1681 17 1717 177   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 18 184 188 19 20 201 203 218 229 244 248 264 285 288 289 294   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 30 31 32 322 325 345 346 36 364 376 379 381 40 41 47 49   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 50 51 54 58 6 61 617 64 644 681 70 724 755 76 78 788   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 80 81 82 87 88 93 983 99 236 29 38 39 43 45 52 63   
## 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2   
## 67 68 85 9 92 14 15 16 22 27 28 55 8 98 21 23   
## 2 2 2 2 2 3 3 3 3 3 3 3 3 3 4 4   
## 25 26 13 33 4 7 10 12 5 3 2 1   
## 4 4 5 5 5 6 7 7 8 9 13 19

df\_remove <- df\_count$name[df\_count$count <= 3]  
  
df\_new <- subset(BCI, select = ! (names(BCI) %in% df\_remove))   
  
df\_new$other <- rowSums(BCI[,df\_remove])

colnames(BCI.env)

## [1] "UTM.EW" "UTM.NS" "Precipitation" "Elevation"   
## [5] "Age.cat" "Geology" "Habitat" "Stream"   
## [9] "EnvHet"

env\_BCI <- BCI.env[,c("Age.cat","Habitat","Stream")]  
print(is.factor(env\_BCI$Habitat))

## [1] TRUE

print(levels(env\_BCI$Habitat))

## [1] "OldHigh" "OldLow" "OldSlope" "Swamp" "Young"

print(levels(env\_BCI$Age.cat))

## [1] "c2" "c3"

print(levels(env\_BCI$Stream))

## [1] "No" "Yes"

Build an ordination of objects using NMDS methods

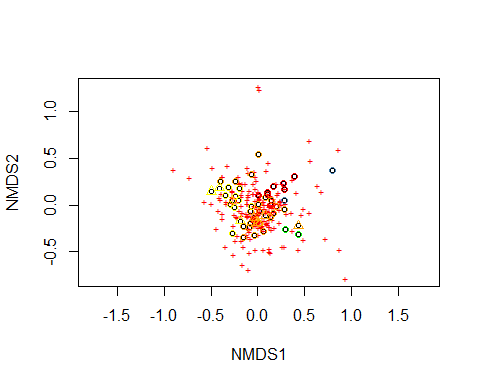
ord\_BCI <- metaMDS(df\_new, dist = "bray")

## Square root transformation  
## Wisconsin double standardization  
## Run 0 stress 0.2498986   
## Run 1 stress 0.2509668   
## Run 2 stress 0.2517131   
## Run 3 stress 0.2514389   
## Run 4 stress 0.2499884   
## ... Procrustes: rmse 0.01810955 max resid 0.07083669   
## Run 5 stress 0.2941859   
## Run 6 stress 0.2688946   
## Run 7 stress 0.2517131   
## Run 8 stress 0.2518347   
## Run 9 stress 0.2537426   
## Run 10 stress 0.2683135   
## Run 11 stress 0.2495902   
## ... New best solution  
## ... Procrustes: rmse 0.01445933 max resid 0.05433657   
## Run 12 stress 0.2786377   
## Run 13 stress 0.2644418   
## Run 14 stress 0.2680237   
## Run 15 stress 0.2524586   
## Run 16 stress 0.2538829   
## Run 17 stress 0.2511005   
## Run 18 stress 0.2825958   
## Run 19 stress 0.2521248   
## Run 20 stress 0.2495902   
## ... Procrustes: rmse 2.275433e-05 max resid 0.0001026438   
## ... Similar to previous best  
## \*\*\* Best solution repeated 1 times

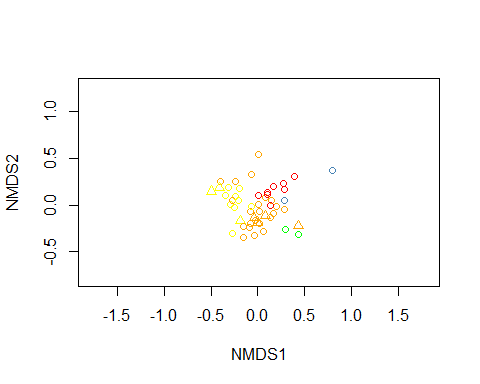
ord\_BCI\_auto <- metaMDS(df\_new, dist = "bray", autotransform = F)

## Run 0 stress 0.1745903   
## Run 1 stress 0.1851058   
## Run 2 stress 0.1997372   
## Run 3 stress 0.191893   
## Run 4 stress 0.2106615   
## Run 5 stress 0.1789257   
## Run 6 stress 0.2000756   
## Run 7 stress 0.1745903   
## ... Procrustes: rmse 5.702491e-06 max resid 2.779882e-05   
## ... Similar to previous best  
## Run 8 stress 0.1851059   
## Run 9 stress 0.1919389   
## Run 10 stress 0.1745903   
## ... New best solution  
## ... Procrustes: rmse 1.227572e-05 max resid 7.479701e-05   
## ... Similar to previous best  
## Run 11 stress 0.1745903   
## ... Procrustes: rmse 1.537893e-05 max resid 9.567874e-05   
## ... Similar to previous best  
## Run 12 stress 0.2116272   
## Run 13 stress 0.1889226   
## Run 14 stress 0.1851058   
## Run 15 stress 0.183977   
## Run 16 stress 0.1793857   
## Run 17 stress 0.1911512   
## Run 18 stress 0.1889227   
## Run 19 stress 0.1793859   
## Run 20 stress 0.1989564   
## \*\*\* Best solution repeated 2 times

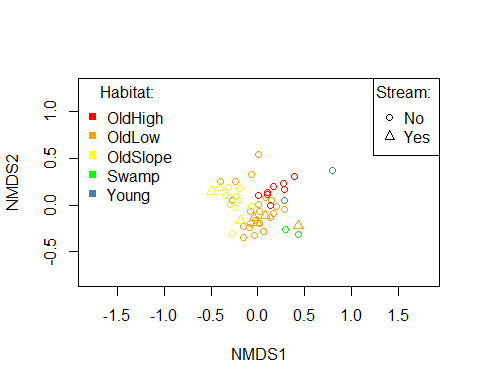
pal\_col <- c("red","orange", "yellow", "green", "steelblue")  
pal\_sh <- c(1, 2)  
  
  
ordiplot(ord\_BCI\_auto, type = "point")  
points(ord\_BCI\_auto, col = pal\_col[env\_BCI$Habitat], pch = pal\_sh[env\_BCI$Stream])



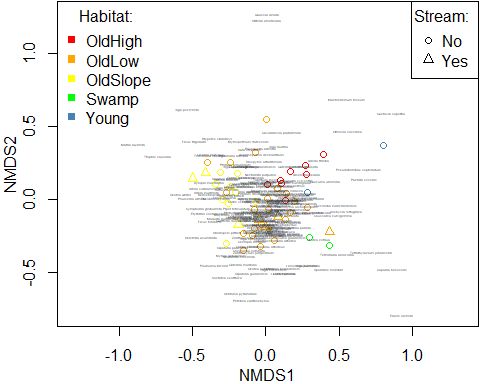
ordiplot(ord\_BCI\_auto, type = "n")  
points(ord\_BCI\_auto, col = pal\_col[env\_BCI$Habitat], pch = pal\_sh[env\_BCI$Stream])

 We have to use ord\_BCI\_auto because without autotransformation the stress is more than 0.2

ordiplot(ord\_BCI\_auto, type = "n")  
points(ord\_BCI\_auto, col = pal\_col[env\_BCI$Habitat], pch = pal\_sh[env\_BCI$Stream])  
# Legend (example of relative and absolute positioning)  
legend("topleft", bty = "n",  
 title = "Habitat: ",  
 legend = levels(env\_BCI$Habitat), col = pal\_col, pch = 15)  
legend("topright", xjust = 1, yjust = 1,  
 title = "Stream: ",  
 legend = levels(env\_BCI$Stream), col = "black", pch = pal\_sh)



op <- par(mar = c(3, 3, 0.1, 0.1), mgp = c(2, 1, 0))  
ordiplot(ord\_BCI\_auto, type = "n")  
points(ord\_BCI\_auto, col = pal\_col[env\_BCI$Habitat], pch = pal\_sh[env\_BCI$Stream])  
legend("topleft", bty = "n",  
 title = "Habitat: ",  
 legend = levels(env\_BCI$Habitat), col = pal\_col, pch = 15)  
legend("topright", xjust = 1, yjust = 1,  
 title = "Stream: ",  
 legend = levels(env\_BCI$Stream), col = "black", pch = pal\_sh)  
text(ord\_BCI\_auto, display = "species", cex = 0.2, col = "grey20")



par(op)

## Envfit

ef <- envfit(ord\_BCI\_auto, env\_BCI)  
  
scores(ef, "vectors")  
  
ef$vectors

## NULL

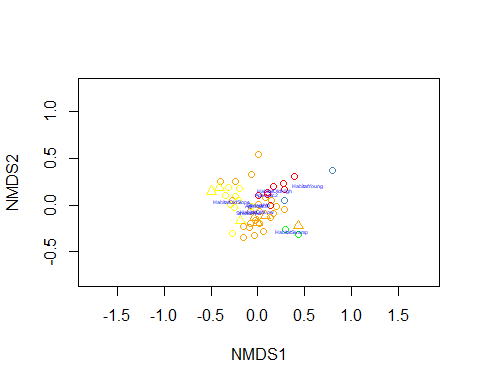
ef$factors

## Centroids:  
## NMDS1 NMDS2  
## Age.catc2 0.1070 0.1087  
## Age.catc3 -0.0022 -0.0022  
## HabitatOldHigh 0.1833 0.1552  
## HabitatOldLow 0.0000 -0.0582  
## HabitatOldSlope -0.2731 0.0349  
## HabitatSwamp 0.3646 -0.2850  
## HabitatYoung 0.5413 0.2110  
## StreamNo 0.0143 0.0116  
## StreamYes -0.0880 -0.0710  
##   
## Goodness of fit:  
## r2 Pr(>r)   
## Age.cat 0.0046 0.800   
## Habitat 0.4926 0.001 \*\*\*  
## Stream 0.0200 0.363   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Permutation: free  
## Number of permutations: 999

colnames(env\_BCI)

## [1] "Age.cat" "Habitat" "Stream"

ordiplot(ord\_BCI\_auto, type = "n")  
points(ord\_BCI\_auto, col = pal\_col[env\_BCI$Habitat], pch = pal\_sh[env\_BCI$Stream])  
plot(ef, cex = 0.3)

 I didn’t get any vectors and I decided to change my variables from character to numeric

levels(env\_BCI$Habitat)

## [1] "OldHigh" "OldLow" "OldSlope" "Swamp" "Young"

df <- env\_BCI  
  
df %>% mutate\_if(is.factor, as.character) -> df  
class(df$Habitat)

## [1] "character"

df['Habitat'][df['Habitat'] == 'Young'] <- 0  
df['Habitat'][df['Habitat'] == 'Swamp'] <- 1  
df['Habitat'][df['Habitat'] == 'OldSlope'] <- 2  
df['Habitat'][df['Habitat'] == 'OldLow'] <- 3  
df['Habitat'][df['Habitat'] == 'OldHigh'] <- 4  
  
df['Stream'][df['Stream'] == 'No'] <- 0  
df['Stream'][df['Stream'] == 'Yes'] <- 1  
  
df['Age.cat'][df['Age.cat'] == 'c2'] <- 0  
df['Age.cat'][df['Age.cat'] == 'c3'] <- 1  
  
df$Habitat\_f <- as.factor(df$Habitat)  
levels(df$Habitat\_f)

## [1] "0" "1" "2" "3" "4"

df$Stream\_f <- as.factor(df$Stream)  
levels(df$Stream\_f)

## [1] "0" "1"

df$Age.cat\_f <- as.factor(df$Age.cat)  
levels(df$Age.cat\_f)

## [1] "0" "1"

ef <- envfit(ord\_BCI\_auto, df[,c("Habitat\_f", "Stream\_f")])  
  
scores(ef, "vectors")  
  
ef$vectors

## NULL

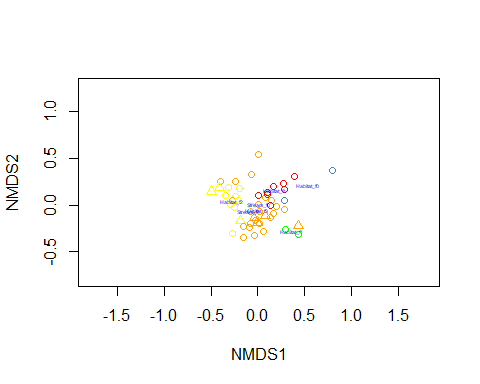
ef$factors

## Centroids:  
## NMDS1 NMDS2  
## Habitat\_f0 0.5413 0.2110  
## Habitat\_f1 0.3646 -0.2850  
## Habitat\_f2 -0.2731 0.0349  
## Habitat\_f3 0.0000 -0.0582  
## Habitat\_f4 0.1833 0.1552  
## Stream\_f0 0.0143 0.0116  
## Stream\_f1 -0.0880 -0.0710  
##   
## Goodness of fit:  
## r2 Pr(>r)   
## Habitat\_f 0.4926 0.001 \*\*\*  
## Stream\_f 0.0200 0.368   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Permutation: free  
## Number of permutations: 999

colnames(env\_BCI)

## [1] "Age.cat" "Habitat" "Stream"

ordiplot(ord\_BCI\_auto, type = "n")  
points(ord\_BCI\_auto, col = pal\_col[env\_BCI$Habitat], pch = pal\_sh[env\_BCI$Stream])  
plot(ef, cex = 0.3)

 And here I didn’t get vectors. I understood a problem, i need to use continuous variables:

ef <- envfit(ord\_BCI\_auto, BCI.env[,c("EnvHet","UTM.NS","UTM.EW")])  
  
scores(ef, "vectors")

## NMDS1 NMDS2  
## EnvHet -0.01922091 0.11581279  
## UTM.NS 0.14261225 -0.07170342  
## UTM.EW -0.14704688 0.85993489

levels(BCI.env$EnvHet)

## NULL

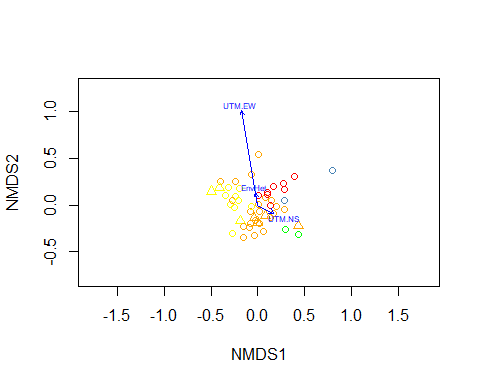
ef$vectors

## NMDS1 NMDS2 r2 Pr(>r)   
## EnvHet -0.16373 0.98651 0.0138 0.752   
## UTM.NS 0.89343 -0.44920 0.0255 0.540   
## UTM.EW -0.16855 0.98569 0.7611 0.001 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Permutation: free  
## Number of permutations: 999

ef$factors

## NULL

ordiplot(ord\_BCI\_auto, type = "n")  
points(ord\_BCI\_auto, col = pal\_col[env\_BCI$Habitat], pch = pal\_sh[env\_BCI$Stream])  
plot(ef, cex = 0.5)

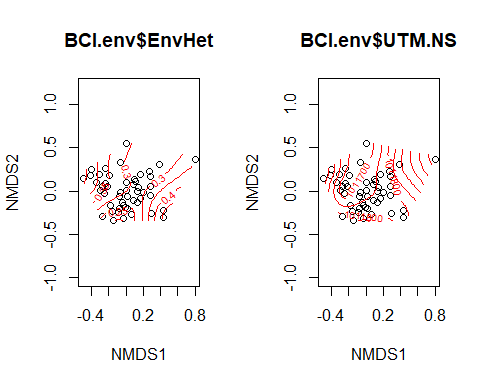


Ordisurf:

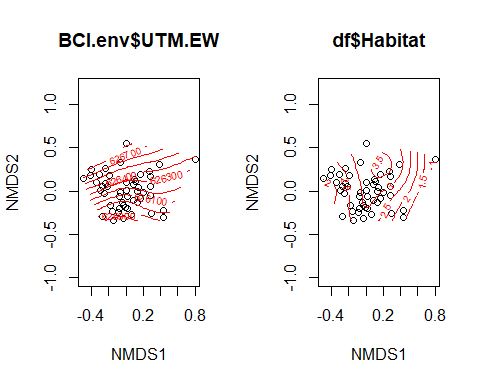
class(df$Habitat)

## [1] "character"

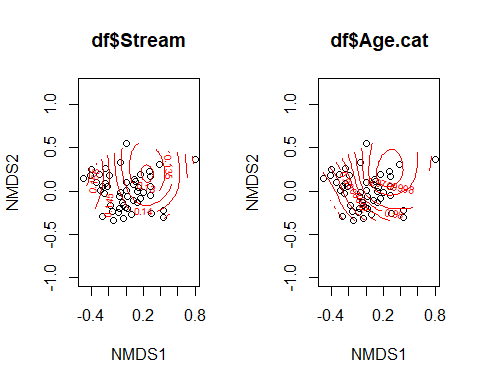
df$Habitat <- as.numeric(df$Habitat)  
df$Age.cat <- as.numeric(df$Age.cat)  
df$Stream <- as.numeric(df$Stream)  
  
par(mfrow = c(1, 2))  
os\_EnvHet <- ordisurf(ord\_BCI\_auto, BCI.env$EnvHet, method = "REML")  
os\_UTM.NS <- ordisurf(ord\_BCI\_auto, BCI.env$UTM.NS, method = "REML")



os\_UTM.EW <- ordisurf(ord\_BCI\_auto, BCI.env$UTM.EW, method = "REML")  
os\_EnvHet <- ordisurf(ord\_BCI\_auto, df$Habitat, method = "REML")



os\_UTM.NS <- ordisurf(ord\_BCI\_auto, df$Stream, method = "REML")  
os\_UTM.EW <- ordisurf(ord\_BCI\_auto, df$Age.cat, method = "REML")



par(mfrow = c(1, 1))

# The result of these vectors:

1. EnvHet and UTM.EW have a strong positive correlation
2. Two factors (EnvHet, UTM.EW) have a moderate negative correlation with UTM.NS

The most strong environmental factor is UTM.EW

# The result of surface plots

We can see the complex non-linear impact of different factors on the amount of trees except UTM.EW, at this plot we can see approximate linear relationship