Procvičování 5 s řešením

Natáhněte si pracovní prostředí z `http://www.sci.muni.cz/~syrovat/pakomari.RData`.

```
load(url("http://www.sci.muni.cz/~syrovat/pakomari.RData"))
ls()

## [1] "env" "spe"
```

1. Převeďte matici *spe* na dataframe a všechny její NA hodnoty nahraďte nulami (0).

```
spe <- as.data.frame(spe)
spe[is.na(spe)] <- 0
spe[1:10, 1:5]

##     ablabesp Apsetrif Brilbif Brilflav cladotsp
## s1         0        1       0        0        0
## s2         1        1       0        1        3
## s3         0        2       0        0        1
## s4         0        0       0        0        5
## s5         1        0       2        1       18
## s6         0        2       0        0       25
## s7         0        0       0        0        0
## s8         0        0       0        1        0
## s9         0        0       0        0        1
## s10        0        0       0        0        0
```

2. Přejmenujte proměnné *gr_ env* a *velocity* dataframu *env* na *hab* a *vel*, resp.

```
# samozrejme, moznosti je vic. muzu se nejprve podivat na jmena a zjistit
# poradi tech, ktera chci zmenit (predtim si vytvorim kopii originalu, abych
# mohl demonstrovat i jednu z dalsich moznosti):
env.orig <- env
names(env)

## [1] "gr_env"   "depth"    "velocity"

# vidim, ze se jedna o jmena na 1. a 3. pozici, zmenim tedy ty:
names(env)[c(1, 3)] <- c("hab", "vel")
head(env)

##        hab depth   vel
## s1      Ep 0.395 0.274
## s2      Ep 0.422 0.358
## s3      Ep 0.496 0.310
## s4 Ep_FPOM 0.291 0.078
## s5 Ep_FPOM 0.320 0.092
## s6 Ep_FPOM 0.328 0.126
```

```
# dalsi jednoduchou moznosti je nahrazovat postupne vybirana konkretni
# jmena:
env <- env.orig
names(env)

## [1] "gr_env"   "depth"    "velocity"

names(env)[names(env) == "gr_env"] <- "hab"
names(env)[names(env) == "velocity"] <- "vel"
head(env)

##        hab depth   vel
## s1      Ep 0.395 0.274
## s2      Ep 0.422 0.358
## s3      Ep 0.496 0.310
## s4 Ep_FPOM 0.291 0.078
## s5 Ep_FPOM 0.320 0.092
## s6 Ep_FPOM 0.328 0.126
```

3. Proměnnou *vel* dataframu *env* přepište absolutní hodnotou této proměnné (záporná rychlost proudu je nesmyslná).

```
env$vel <- abs(env$vel)
```

4. V dataframu *env* vytvořte proměnnou *fr* obsahující hodnoty Froudeho čísla ($Fr = U/(gD)^{1/2}$, kde U = absolutní hodnota rychlosti proudu, D = hloubka, g = 9.81.)

```
env$fr <- env$vel/sqrt(env$depth * 9.81)
head(env)

##        hab depth   vel      fr
## s1      Ep 0.395 0.274 0.13919
## s2      Ep 0.422 0.358 0.17595
## s3      Ep 0.496 0.310 0.14054
## s4 Ep_FPOM 0.291 0.078 0.04617
## s5 Ep_FPOM 0.320 0.092 0.05193
## s6 Ep_FPOM 0.328 0.126 0.07024
```

5. Vytvořte dataframe *micr* (Podle druhu *Microtendipes chloris*) s proměnnými *depth*, *fr* a *abund*, obsahující hloubku a hodnoty Froudeho čísla dataframu *env* a abundance druhu *Microtendipes chloris* (zkratka *micrchgr*) dataframu *spe*. Řádky dataframu *micr* pojmenujte stejně jako řádky dataframu *env*.

```
micr <- data.frame(depth = env$depth, fr = env$fr, abund = spe$micrchgr)
head(micr)
```

```
##   depth      fr abund
## 1 0.395 0.13919    25
## 2 0.422 0.17595    31
## 3 0.496 0.14054    31
## 4 0.291 0.04617    14
## 5 0.320 0.05193    30
## 6 0.328 0.07024    31
```

6. V dataframu *micr* vytvořte proměnnou *abundL* s logaritmovanými abundancemi druhu *Microtendipes chloris*. Nula nejde logaritmovat, proto logaritmujte (hodnoty abundance + 1) a použijte přirozený logaritmus.

```
micr$abundL <- log1p(micr$abund)
head(micr)

##   depth      fr abund abundL
## 1 0.395 0.13919    25  3.258
## 2 0.422 0.17595    31  3.466
## 3 0.496 0.14054    31  3.466
## 4 0.291 0.04617    14  2.708
## 5 0.320 0.05193    30  3.434
## 6 0.328 0.07024    31  3.466
```

7. V dataframu *micr* vytvořte proměnnou *prezence*, obsahující jedničky (1) a dvojky (2) a odlišující kladné nenulové hodnoty abundance druhu *M. chloris* od nulových. Jedničky budou odpovídat nulovým, a dvojky kladným nenulovým hodnotám.

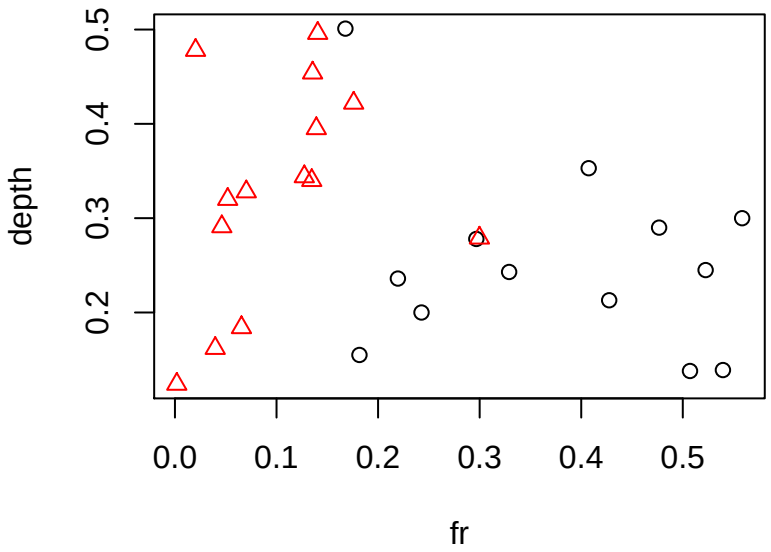
```
# zde neni spatne si prvne vytvoreni vektoru zkouset na necisto, a az kdyz
# vidim, ze to funguje spravne, ulozim si ho pod prislusnym jmenem do
# dataframu.  ...par marnych pokusu a konecne ten povedeny:
(micr$abund > 0) + 1

##  [1] 2 2 2 2 2 2 1 1 1 1 1 1 2 1 2 1 2 1 2 1 1 2 2 1 1 2 2

micr$prezence <- (micr$abund > 0) + 1
```

8. Vytvořte bodový graf (`plot()`) hloubky (*depth*, osa y) proti Froudeho číslu (osa x) a barevně a tvarem symbolu odlište lokality, kde *M. chloris* byl nalezen od těch, kde nebyl.

```
palette("default")  # abych mel defaultni paletu jako mate vy
par(mar = c(4, 4, 1, 0.1))  # aby obrazek nemel velke okraje
plot(depth ~ fr, data = micr, pch = prezence, col = prezence)
```

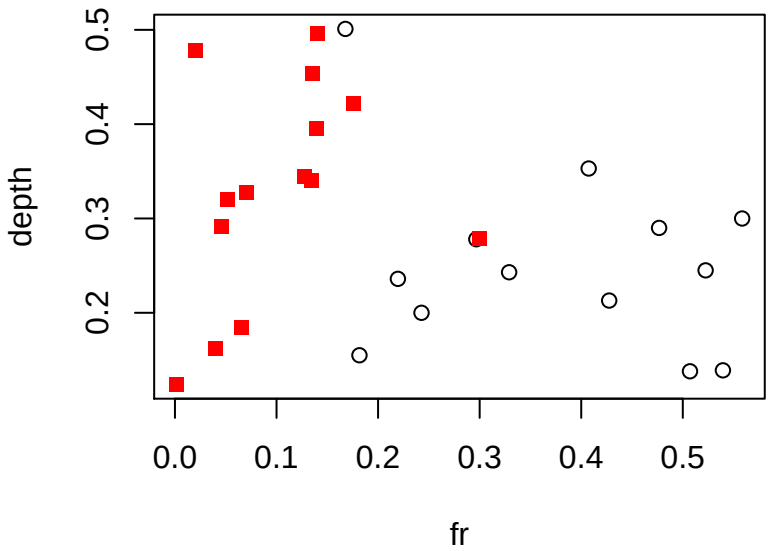


9. Z vektoru *prezence* vytvořte vektor jedniček (1) a patnáctek (15) *puntik* (místo dvojek budou patnáctky).

```
puntik <- c(1, 15)[micr$prezence]
```

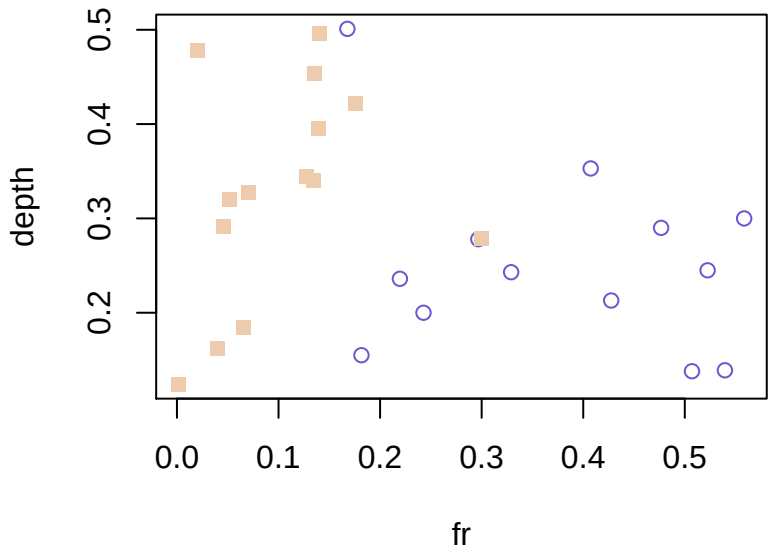
10. Vytořte stejný graf jako v bodě 8, jen pro odlišení tvaru puntíků použijte *puntik*.

```
par(mar = c(4, 4, 1, 0.1))
plot(depth ~ fr, data = micr, pch = puntik, col = prezence)
```



11. Barvy jdou specifikovat i slovně. Seznam pojmenovaných barev získáme příkazem `colors()`, nebo `colours()`. Vyberte si libovolné dvě a vytvořte vektor *barvy*, v němž se budou tyto dvě barvy opakovat podle stejného pravidla jako v bodech 5 a 7. Cílem je odlišit opět lokality s výskytem *M. chloris* od těch bez výskytu a použít k tomu nějaké hezké barvy.

```
barvy <- c("slateblue3", "peachpuff2")[micr$prezence]
par(mar = c(4, 4, 1, 0.1))
plot(depth ~ fr, data = micr, pch = puntik, col = barvy)
```



12. Zdá se, že *M. chloris* nemá rád, když voda moc teče, že si spíš libuje v tůních. Spočtěte průměrné Froudeho číslo lokalit, kde se druh vyskytoval a těch, kde se nevyskytoval (tedy 2 hodnoty).

```
mean(env$fr[micr$abund > 0])

## [1] 0.1035
```

13. Zjistěte, na kolika lokalitách se druh vyskytoval a kolik z těchto lokalit je tůňových (Froudeho číslo $<= 0.23$).

```
# kolik lokalit celkem
sum(micr$abund > 0)

## [1] 14

# kolik z nich je tunovych
sum(micr$abund > 0 & micr$fr <= 0.23)

## [1] 13
```

14. Vytvořte dataframe *micrPool*, který bude obsahovat všechny proměnné dataframu *micr*, ale jen ta měření, kde Froudeho číslo bylo nižší nebo rovno než 0.23.

```
micrPool <- micr[micr$fr <= 0.23, ]
dim(micrPool)

## [1] 16  5
```

15. Z dataframu *micrPool* odstraňte proměnnou *prezence*.

```
# opet vic moznosti. bud do promenne 'vlozime' NULL:
micr.orig <- micr
micr$prezence <- NULL
head(micr)

##   depth      fr abund abundL
## 1 0.395 0.13919    25  3.258
## 2 0.422 0.17595    31  3.466
## 3 0.496 0.14054    31  3.466
## 4 0.291 0.04617    14  2.708
## 5 0.320 0.05193    30  3.434
## 6 0.328 0.07024    31  3.466

# nebo prepiseme dataframe sebou samym bez one promenne (pripadne
# promennych):
micr <- micr.orig
names(micr)

## [1] "depth"    "fr"       "abund"    "abundL"   "prezence"

micr <- micr[, -5]
names(micr)

## [1] "depth"  "fr"     "abund"  "abundL"

micr <- micr.orig
names(micr)

## [1] "depth"    "fr"       "abund"    "abundL"   "prezence"

micr <- micr[, names(micr) != "prezence"]
names(micr)

## [1] "depth"  "fr"     "abund"  "abundL"
```

16. Vytvořte vektor, v němž se bude opakovat pětkrát za sebou sekvence celých čísel od 1 do 5.

```
rep(1:5, 5)

##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

17. Uspřádejte tento vektor do matice o 5 řádcích a 5 sloupcích: (a) po sloupcích, (b) po řádcích.

```
# (a)
matrix(rep(1:5, 5), nrow = 5)

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    1    1    1
```

```
## [2,]    2    2    2    2    2
## [3,]    3    3    3    3    3
## [4,]    4    4    4    4    4
## [5,]    5    5    5    5    5

# (b)
matrix(rep(1:5, 5), nrow = 5, byrow = T)

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    1    2    3    4    5
## [3,]    1    2    3    4    5
## [4,]    1    2    3    4    5
## [5,]    1    2    3    4    5
```

18. Vytvořte matici *ones* jedniček o 9 řádcích a 6 sloupcích.

```
ones <- matrix(1, nrow = 9, ncol = 6)
ones

##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    1    1    1    1    1
##  [2,]    1    1    1    1    1    1
##  [3,]    1    1    1    1    1    1
##  [4,]    1    1    1    1    1    1
##  [5,]    1    1    1    1    1    1
##  [6,]    1    1    1    1    1    1
##  [7,]    1    1    1    1    1    1
##  [8,]    1    1    1    1    1    1
##  [9,]    1    1    1    1    1    1
```

19. Matici *ones* vynásobte tak, aby výsledkem násobení byla matice *onesR*, která má v prvním řádku samé jedničky, ve druhém samé dvojky,... v devátém samé devítky.

```
onesR <- ones * 1:9
onesR

##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    1    1    1    1    1
##  [2,]    2    2    2    2    2    2
##  [3,]    3    3    3    3    3    3
##  [4,]    4    4    4    4    4    4
##  [5,]    5    5    5    5    5    5
##  [6,]    6    6    6    6    6    6
##  [7,]    7    7    7    7    7    7
##  [8,]    8    8    8    8    8    8
##  [9,]    9    9    9    9    9    9
```

20. Matici *ones* vynásobte tak, aby výsledkem násobení byla matice *onesC*, která má v prvním sloupci samé jedničky, ve druhém samé dvojky,... v šestém samé šestky.

```
# abych se vyhnul vytvareni matice, kterou bych nasobil, muzu si matici ones
# transponovat, a po nasobeni vratit zpet:
onesC <- t(t(ones) * 1:6)
onesC
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    2    3    4    5    6
##  [2,]    1    2    3    4    5    6
##  [3,]    1    2    3    4    5    6
##  [4,]    1    2    3    4    5    6
##  [5,]    1    2    3    4    5    6
##  [6,]    1    2    3    4    5    6
##  [7,]    1    2    3    4    5    6
##  [8,]    1    2    3    4    5    6
##  [9,]    1    2    3    4    5    6
```

```
# nebo si vytvorim tu matici:
ones * matrix(rep(1:6, each = 9), nrow = 9)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    2    3    4    5    6
##  [2,]    1    2    3    4    5    6
##  [3,]    1    2    3    4    5    6
##  [4,]    1    2    3    4    5    6
##  [5,]    1    2    3    4    5    6
##  [6,]    1    2    3    4    5    6
##  [7,]    1    2    3    4    5    6
##  [8,]    1    2    3    4    5    6
##  [9,]    1    2    3    4    5    6
```

```
# v kazdem pripade, na rozdil od nasobeni radku, je nasobeni jednotlivych
# sloupcu matice je krkolomne.
```

21. Použijte postupně matice *onesR* a *onesC* jako subscript pro výběr jmen druhů z dataframu *spe*.

```
names(spe)[onesR]
```

```
##  [1] "ablabesp" "Apsetrif" "Brilbif"  "Brilflav" "cladotsp" "corysp."
##  [7] "Cricannu" "Cricbici" "cricbigr" "ablabesp" "Apsetrif" "Brilbif"
## [13] "Brilflav" "cladotsp" "corysp."  "Cricannu" "Cricbici" "cricbigr"
## [19] "ablabesp" "Apsetrif" "Brilbif"  "Brilflav" "cladotsp" "corysp."
## [25] "Cricannu" "Cricbici" "cricbigr" "ablabesp" "Apsetrif" "Brilbif"
## [31] "Brilflav" "cladotsp" "corysp."  "Cricannu" "Cricbici" "cricbigr"
## [37] "ablabesp" "Apsetrif" "Brilbif"  "Brilflav" "cladotsp" "corysp."
## [43] "Cricannu" "Cricbici" "cricbigr" "ablabesp" "Apsetrif" "Brilbif"
## [49] "Brilflav" "cladotsp" "corysp."  "Cricannu" "Cricbici" "cricbigr"
```

```
names(spe)[onesC]
```

```
##  [1] "ablabesp" "ablabesp" "ablabesp" "ablabesp" "ablabesp" "ablabesp"
##  [7] "ablabesp" "ablabesp" "ablabesp" "Apsetrif" "Apsetrif" "Apsetrif"
```

```
## [13] "Apsetrif" "Apsetrif" "Apsetrif" "Apsetrif" "Apsetrif" "Apsetrif"
## [19] "Brilbif"  "Brilbif"  "Brilbif"  "Brilbif"  "Brilbif"  "Brilbif"
## [25] "Brilbif"  "Brilbif"  "Brilbif"  "Brilflav" "Brilflav" "Brilflav"
## [31] "Brilflav" "Brilflav" "Brilflav" "Brilflav" "Brilflav" "Brilflav"
## [37] "cladotsp" "cladotsp" "cladotsp" "cladotsp" "cladotsp" "cladotsp"
## [43] "cladotsp" "cladotsp" "cladotsp" "corysp."  "corysp."  "corysp."
## [49] "corysp."  "corysp."  "corysp."  "corysp."  "corysp."  "corysp."
```

22. Aplikujte funkce `row()` a `col()` na matici ones a zamyslete se nad výsledkem. K čemu se tyto funkce hodí?

```
row(ones)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    1    1    1    1    1
##  [2,]    2    2    2    2    2    2
##  [3,]    3    3    3    3    3    3
##  [4,]    4    4    4    4    4    4
##  [5,]    5    5    5    5    5    5
##  [6,]    6    6    6    6    6    6
##  [7,]    7    7    7    7    7    7
##  [8,]    8    8    8    8    8    8
##  [9,]    9    9    9    9    9    9
```

```
col(ones)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    2    3    4    5    6
##  [2,]    1    2    3    4    5    6
##  [3,]    1    2    3    4    5    6
##  [4,]    1    2    3    4    5    6
##  [5,]    1    2    3    4    5    6
##  [6,]    1    2    3    4    5    6
##  [7,]    1    2    3    4    5    6
##  [8,]    1    2    3    4    5    6
##  [9,]    1    2    3    4    5    6
```

```
# funkce row() a col() nam pro kazdou bunku maticoveho objektu
# (matice/dataframe) vrati jeho pozici na radku a ve sloupci, respective.
# funkci row() mozna prilis nevyuzijemem, ale col() je skvela pro nasobeni
# sloupcu - roztahne nam nas vektor va rozmery nasobene matice po radcich:
ones * (1:6)[col(ones)]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    2    3    4    5    6
##  [2,]    1    2    3    4    5    6
##  [3,]    1    2    3    4    5    6
##  [4,]    1    2    3    4    5    6
##  [5,]    1    2    3    4    5    6
##  [6,]    1    2    3    4    5    6
##  [7,]    1    2    3    4    5    6
##  [8,]    1    2    3    4    5    6
##  [9,]    1    2    3    4    5    6
```

```
# muzu samozrejme nasobit i nejakymi zajimavejsimi hodnotami, napr. 0.5, 2,
# 10, 0, 1 a -12:
ones * c(0.5, 2, 10, 0, 1, -12)[col(ones)]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]  0.5    2   10    0    1  -12
##  [2,]  0.5    2   10    0    1  -12
##  [3,]  0.5    2   10    0    1  -12
##  [4,]  0.5    2   10    0    1  -12
##  [5,]  0.5    2   10    0    1  -12
##  [6,]  0.5    2   10    0    1  -12
##  [7,]  0.5    2   10    0    1  -12
##  [8,]  0.5    2   10    0    1  -12
##  [9,]  0.5    2   10    0    1  -12
```

23. Jednou z možných transformací druhových dat předcházejících vícerozměrným analýzám je Wisconsin double standardization, která spočívá v tom, že abundance jednotlivých druhů nejprve standardizujeme druhovým maximem (hodnoty v každém sloupečku vydělíme maximální hodnotou daného sloupečku) a posléze lokalitní sumou (hodnoty každého řádku vydělíme sumou hodnot daného řádku). Do dataframu *spe.wis* standardizujte druhové abundance *spe* pomocí Wisconsin double standardization. (Nápověda: vektor druhových maxim získáme pomocí `sapply(spe, max)`).

```
spe.max <- spe/sapply(spe, max)[col(spe)]
spe.wis <- spe.max/rowSums(spe.max)
spe.wis[1:10, 1:6]
```

```
##     ablabesp Apsetrif Brilbif Brilflav cladotsp corysp.
## s1   0.00000  0.02748 0.00000  0.00000 0.000000 0.08628
## s2   0.04872  0.01624 0.00000  0.01462 0.006007 0.06459
## s3   0.00000  0.10873 0.00000  0.00000 0.006703 0.11379
## s4   0.00000  0.00000 0.00000  0.00000 0.013838 0.04698
## s5   0.03543  0.00000 0.02362  0.01063 0.026212 0.05439
## s6   0.00000  0.02514 0.00000  0.00000 0.038741 0.08419
## s7   0.00000  0.00000 0.00000  0.00000 0.000000 0.02199
## s8   0.00000  0.00000 0.00000  0.02892 0.000000 0.01345
## s9   0.00000  0.00000 0.00000  0.00000 0.001814 0.04927
## s10  0.00000  0.00000 0.00000  0.00000 0.000000 0.02639
```