Procvičování 11 - řešení

Naimportujte do **R** dataframy *spe* a *env* z excelového souboru *dat09.xls* ve studijních materiálech (cv09).

1. Vytvořte funkci `minmax()`, která vrátí nejnižší a nevyšší hodnotu ve vektoru. Otestujte funkci na libovolném vektoru.

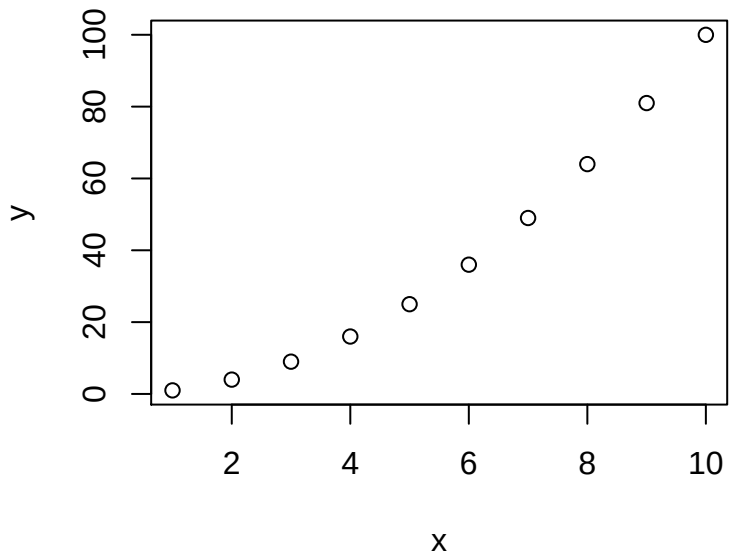
```
minmax<- function(x){
  c(min(x), max(x))
}
minmax(c(2,-3,1,5,0))

## [1] -3  5

# co kdyz budou v x nejake NA hodnoty? pak by se hodilo ponechat prostor pro pouziti
# argumentu na.rm= ve funkcich min() a max(), pripadne tento argument pouzit:

minmax<- function(x, ...){
  c(min(x, ...), max(x, ...))
}

minmax(c(2,-3,1,5,0))

## [1] -3  5

minmax(c(2,-3,1,5,NA))

## [1] NA NA

minmax(c(2,-3,1,5,NA), na.rm= T)

## [1] -3  5

# pokud explicitne na.rm= pouzijeme v nasi funkci, muzeme mu priradit defaultni hodnotu
# treba TRUE:
minmax<- function(x, na.rm= T){
# (funkce minmax bude funkci x a na.rm, argument na.rm jsem schvalne pojmenoval na.rm,
# protoze tak se jmenuje i v jinych funkcich, mohl by se jmenovat ale jakkoliv jinak).
  c(min(x, na.rm= na.rm), max(x, na.rm= na.rm))
}
# nenechte se zmast tim na.rm= na.rm. prvni na.rm= je argument funkce min(),
# druhe na.rm (to za rovnitkem) je argument nasi minmax() funkce, ktery nese hodnotu TRUE,
# pokud ji pri pouziti funkce minmax() nezmenime na FALSE.
# rikame tak R, aby ve funkci min() pouzil argument na.rm= tak, jak jsme ho pouzili ve funkci
# minmax()

minmax(c(2,-3,1,5,NA))

## [1] -3  5

minmax(c(2,-3,1,5,NA), na.rm= F)

## [1] NA NA
```

2. Vytvořte funkci `mocnina()`, která vrátí umocněné hodnoty na druhou.

```
mocnina<- function(x){
  x^2
}
mocnina(c(1,3,5))

## [1]  1  9 25
```

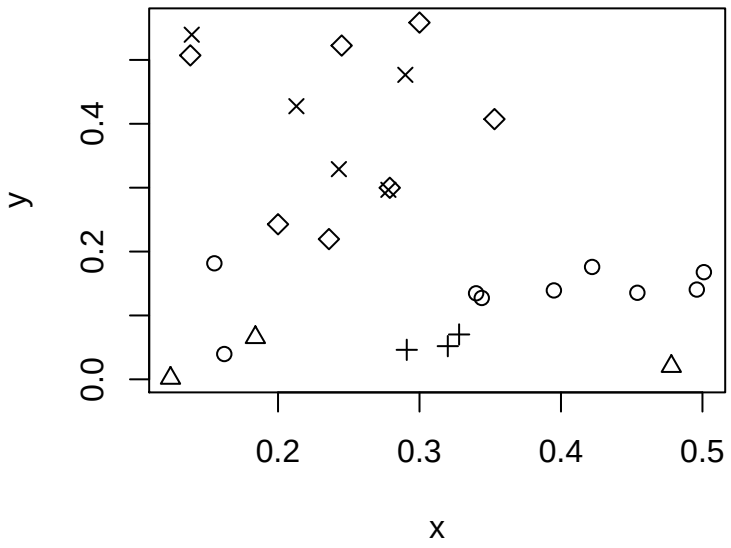
3. Vytvořte funkci `myplot()`, která v bodovém grafu zobrazí proti sobě 2 zadané vektory.

```
par(mar= c(4,4,1,1))
myplot<- function(x, y){
  plot(y ~ x)
}
myplot(1:10, (1:10)^2)
```



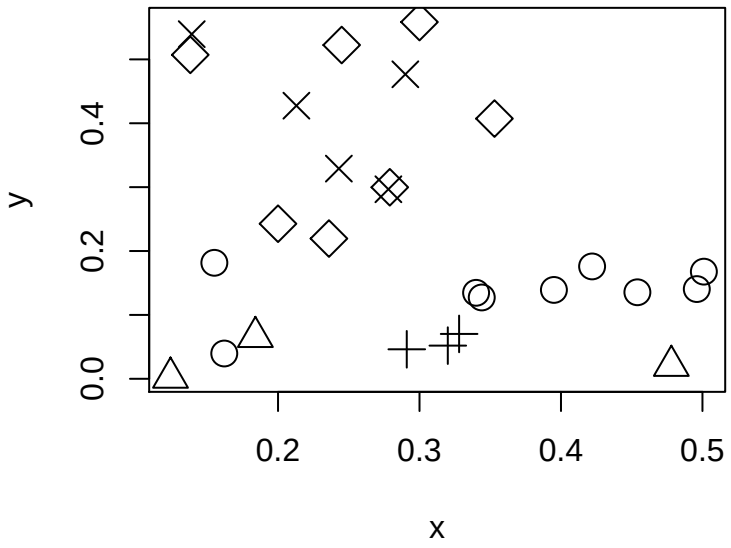
4. Do funkce `myplot()` přidejte argument `group=`, která zajistí odlišení bodů v grafu pomocí různých symbolů. Otestujte na proměnných *depth*, *fr* a *gr* dataframu *env*.

```
par(mar= c(4,4,1,1))
myplot<- function(x, y, group){
  plot(y ~ x, pch= as.numeric(group))
}
myplot(env$depth, env$fr, env$gr)

## Warning: Name partially matched in data frame
```



5. Do funkce `myplot()` přidejte argument `...`, který ponechá možnost přidání dalších argumentů funkce `plot()`, například nastavení větší nebo menší velikosti bodů v grafu. Otestujte na proměnných *depth*, *fr* a *gr* dataframu *env*, zkuste několik variant s různě velkými symboli.

```
par(mar= c(4,4,1,1))
myplot<- function(x, y, group, ...){
  plot(y ~ x, pch= as.numeric(group), ...)
}
myplot(env$depth, env$fr, env$gr, cex= 1.8)

## Warning: Name partially matched in data frame
```



6. Vytvořte funkci `showme()`, která zobrazí prvních 5 řádků a sloupců zadaného tabulkového objektu. Otestujte na dataframu *spe*.

```
showme<- function(tab){
  return(tab[1:5, 1:5])
}
showme(spe)

##     ablabesp apsetrif brilmode brilflav cladotsp
## s01        0        1        0        0        0
## s02        1        1        0        1        3
## s03        0        2        0        0        1
## s04        0        0        0        0        5
## s05        1        0        2        1       18
```

7. Upravte funkci `showme()` tak aby zobrazila prvních 5 a posledních 5 řádků a sloupců zadaného tabulkového objektu.

```
showme<- function(dtf){
  nRow<- nrow(dtf)
  nCol<- ncol(dtf)
  dtf[c(1:5, (nRow-4):nRow), c(1:5, (nCol-4):nCol)]
}
showme(spe)

##     ablabesp apsetrif brilmode brilflav cladotsp tanytasp thellasp
## s01        0        1        0        0        0        4        5
## s02        1        1        0        1        3        1        9
## s03        0        2        0        0        1        8        3
## s04        0        0        0        0        5       12        4
## s05        1        0        2        1       18       31       16
## s23        0        0        0        0       22        5        3
## s24        0        0        0        0        2        4        1
## s25        0        0        0        1        0       11        0
## s26        3        6        0        1        9       47        5
## s27        0        3        9        1        2        4        2
##     thiegrge tvetbaca tvetdive
## s01        5        4        4
## s02       15       11        7
## s03        8        8        5
## s04       17        2        1
## s05       59        2        0
## s23        6        0        0
## s24        6      177      116
## s25        1        1        1
## s26      127        1        1
## s27       32        0        2
```

8. Přidejte do funkce `showme()` argument `n=`, pomocí něhož budé možné specifikovat, kolik (`n`) prvních a posledních řádků zadaného tabulkového objektu bude zobrazeno. Jako default nastavte hodnotu `n` na 5. Počet sloupců nechejte na 5 + 5.

```
showme<- function(dtf, n){
  nRow<- nrow(dtf)
  nCol<- ncol(dtf)
  dtf[c(1:n, (nRow-(n-1)):nRow), c(1:n, (nCol-(n-1)):nCol)]
}
showme(spe, 3)

##     ablabesp apsetrif brilmode thiegrge tvetbaca tvetdive
## s01        0        1        0        5        4        4
## s02        1        1        0       15       11        7
## s03        0        2        0        8        8        5
## s25        0        0        0        1        1        1
## s26        3        6        0      127        1        1
## s27        0        3        9       32        0        2
```

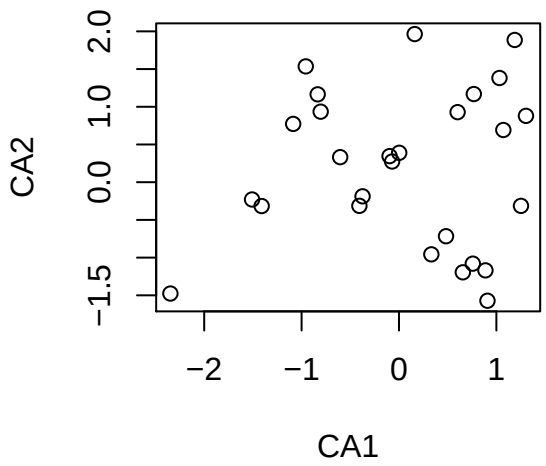
9. Vytvořte funkci `colstat()`, která vrátí počet nenulových hodnot, minimální, mediánovou a maximální hodnotu těchto nenulových hodnot pro všechny sloupce zadaného tabulkového objektu (*mat*). Zjištěné hodnoty nechť jsou uspořádány do matice, kde řádky odpovídají sloupcům *mat* a ve sloupcích jsou jednotlivé statistiky (nenul, min, median, max). Řádky výsledné matice nechť jsou seřazeny sestupně podle mediánu. Otestujte na dataframu *spe*.

```
# tady  muzeme pouzit reseni z prikladu 10, jen ho zobecnime a malicku upravime

colstat<- function(dtf){
  stat<- matrix(NA, nrow= ncol(dtf), ncol= 4)
  dimnames(stat)<- list(names(dtf), c("non-zeros", "min", "median", "max"))

  for(i in names(dtf)){
    vals_i<- dtf[, i]
    # vals_i obsahuje vytazene hodnoty sloupce i
    nuly_i<- vals_i == 0
    # nuly_i oznacuji nulove hodnoty sloupce i
    stat[i, "non-zeros"]<- sum(!nuly_i)
    stat[i, "min"]<- min(vals_i[!nuly_i])
    stat[i, "median"]<- median(vals_i[!nuly_i])
    stat[i, "max"]<- max(vals_i[!nuly_i])
  }
  return(stat)
}

colstat(spe[, 1:10])

##          non-zeros min median max
## ablabesp         4   1    1.0   3
## apsetrif         8   1    2.0   9
## brilmode         3   1    2.0   9
## brilflav        11   1    1.0   4
## cladotsp        17   1    3.0  73
## corysp.         26   1   14.0  43
## cricannu         5   1    1.0   2
## cricbici         7   1    1.0   2
```

```
## cricbigr        16   1    2.5  13
## crictrgr        12   1    2.0   8
```

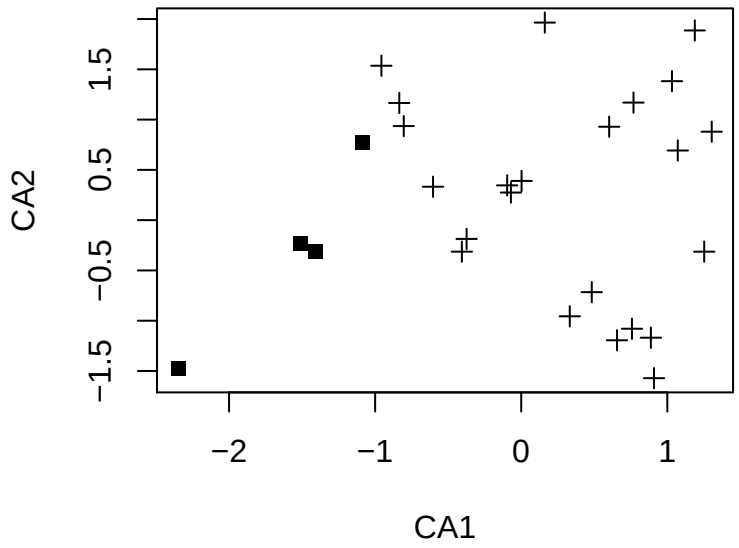
10. Teď připojíme knihovnu balíku *vegan* a uděláme ordinaci pakomářích společenstev pomocí korespondenční analýzy (CA). Vyextrahujeme skóre lokalit na prvních dvou osách ordinace a s těmi budeme dál pracovat.

```
par(mar= c(4,4,1,1))
library(vegan)

## Loading required package: permute
## Loading required package: lattice
## This is vegan 2.0-10

ca1<- cca(log1p(spe))
sc1<- scores(ca1, display= 'sites')
plot(sc1)
```



11. Vytvořte funkci `show.species()`, která v zadané ordinaci odliší symboly lokality, na kterých se zadaný druh vyskytoval od ostatních. Výsledek by měl vypadat asi takhle:

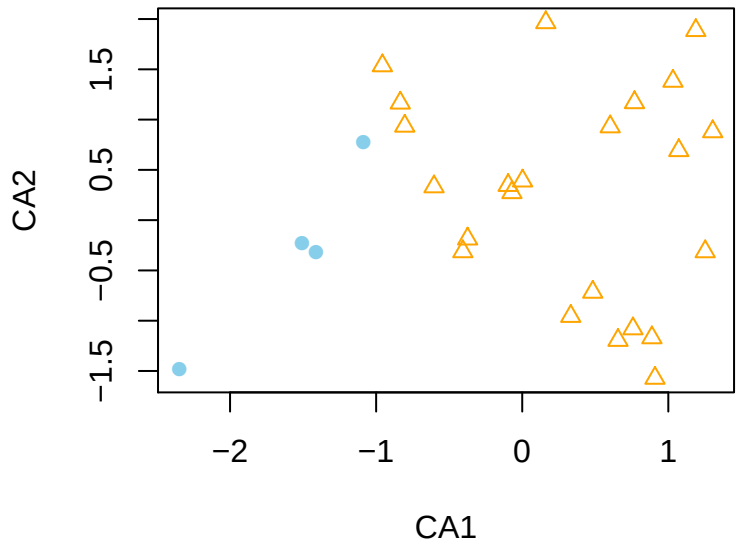
```
par(mar= c(4,4,1,1))
show.species(sc1, "prodoliv")
```




```
# jedno z moznych reseni (slo by to i bez funkce points()):
show.species<- function(ordinace, druh){
  plot(ordinace, type= 'n')
  points(ordinace, pch= c(3,15)[factor(spe[,druh] > 0)])
}

# argument type= n u funkce plot() zajisti, ze neni nic zobrazeno, jen je vytvoreno graficke okno,
# s nastavenym rozsahem os.
# vsimnete si take, ze pro zobrazeni lokalit v dvourozmernem prostoru jejich skore
# (vlastne se jedna o 2 promenne, prvni a druhou osu) staci plot(skore), samozrejme
# bychom mohli pouzit i plot(skore[,1], skore[,2]), nebo plot(skore[,2]~skore[,1]) apod.
```

12. Přidejte do funkce `show.species()` argument, pomocí něhož bude možné specifikovat symboly a barvy bodů zobrazovaných v grafu (samozřejmě aby jedna barva a jeden symbol patřil lokalitám s výskytem druhu a druhá barva a symbol těm ostatním). Příklad:

```
par(mar= c(4,4,1,1))
show.species(sc1, "prodoliv", col=c('orange','skyblue'), pch= c(2,16))
```




```r
show.species<- function(ordinace, druh, col, pch){
  plot(ordinace, type= 'n')
  points(ordinace, pch= pch[factor(spe[,druh] > 0)], col= col[factor(spe[,druh] > 0)])
}
```

13. Přidejte do funkce `show.species()` argument `...`, pomocí něhož bude možné specifikovat další grafické argumenty, například velikost symbolů.

```r
show.species<- function(ordinace, druh, col, pch, ...){
  plot(ordinace, type= 'n', ...)
  points(ordinace, pch= pch[factor(spe[,druh] > 0)], col= col[factor(spe[,druh] > 0)], ...)
}

# ... musi byt ve funkci function() a u jedne z funkci plot() a points(),
# mohou byt i u obou.

# pozorni jedinci by v tuto chvili uz dokazali napsat funkci, ktera by z datasetu
# vytahla zadany pocet nejbeznejsich druhu a zobrazila je obdobnym zpusobem v ordinaci,
# treba jeste s velikosti symbolu odpovidajici (pravdepodobne transformovane, treba sqrt)
# abundanci druhu na lokalitach :)
```