# R – tahák

(Poslední změna: 14. dubna 2018)

## 1 Zahájení práce

```
setwd("popis_cesty_k_adresari")   # nastavení adresáře
rm(list=ls())                     # úklid
```

Dále se může hodit:

```
getwd()               # zjištění pracovního adresáře
ls()                  # výpis proměnných
```

## 2 Práce s daty

Načtení dat (data si pojmenujeme "Data"):

```
Data <- read.csv2("popis/cesty/soubor.csv", header=TRUE)  # načtení dat ve formátu csv
Data <- load("popis/cesty/soubor.RData", header=TRUE)     # načtení dat ve formátu RData
```

Uložení dat:

```
write.table(Data, file = "nazev.csv",row.names=FALSE, col.names=TRUE, sep=";", dec=",")
save(Data, file = "nazev.RData")
```

Prohlížení:

```
View(Data)            # zobrazení dat
print(Data)           # vypsání dat
```

Přístup k jednotlivým údajům:

```
Data$velicina          # přístup k proměnné "veličina" v datech "Data"
Data[5,2]              # přístup k hodnotě v 5. řádku a 2. sloupci souboru Data
Data[5,]               # celý 5. řádek
Data[,2]               # celý 2. sloupec
Data[5,"neco"]         # přístup k hodnotě v 5. řádku ve sloupci s názvem "něco"
```

Přímý přístup k jednotlivým proměnných v datech (nechceme-li používat $).

```
attach(Data)           # "připojení" dat
detach(Data)           # "odpojení" dat
```

Tvorba podmnožiny dat:

```
Data2 <- subset(Data, podminka)      # př. podmínky: veličina1 == veličina2,
                                                   veličina1 <= veličina2, atd.
```

Změna typu proměnné:

```
class(velicina)         # zjistění typu veličiny (numeric, factor, integer, atd.)
as.numeric(velicina)    # R bude veličinu vnímat jako číselnou (typ "numeric")
as.factor(velicina)     # R bude veličinu vnímat jako kategoriální (typ "factor")
```

Práce s faktory:

- `velicina` nabývá tří číselných hodnot a my chceme tyto úrovně pojmenovat a, b, c (úroveň s nejnižším číslem bude pojmenována "a", atd.). Novou proměnnou nazveme `Fvelicina`.

```
Fvelicina<-factor(velicina,labels=c("a","b","c"))
```

- změna uspořádání úrovní (místo a, b, c chceme b, c, a)

```
Fvelicina<-factor(Fvelicina,levels=c("b","c","a"))
```

# 3 Základní popisné statistiky

```
summary(velicina)        # průměr a některé "důležité" kvantily
summary(Data)            # pro všechny proměnné datového souboru
mean(velicina)           # výběrový průměr
sd(velicina)             # výběrová směrodatná odchylka
var(velicina)            # výběrový rozptyl
median(velicina)         # medián
min(velicina)            # minimum
max(velicina)            # maximum
quantile(velicina, prob=c(0, 0.25, 0.50, 0.75, 1.00))        # vybrané kvantily
length(velicina)         # délka vektoru (počet pozorování)
cov(velicina1,velicina2)         # výběrová kovariance
cor(velicina1,velicina2)         # korelační koeficient
mean(scale(velicina)^3)          # výběrová šikmost
mean(scale(velicina)^4)          # výběrová špičatost
sd(velicina)/sqrt(length(velicina))      # směrodatná chyba průměru
```

Dále se může hodit:

```
sum(velicina)            # součet všech hodnot
sqrt(cislo)              # odmocnina ("cislo" může být skalár nebo vektor)
round(cislo, pocet desetinnych mist)     # zaokrouhlení
```

Pro faktorovou veličinu:

```
table(velicina)          # absolutní četnosti jednotlivých kategorií
prop.table(table(velicina))     # relativní četnosti jednotlivých kategorií
```

# 4 Grafy

```
boxplot(velicina)        # krabicový graf
hist(velicina)           # histogram
plot(velicinaX,velicinaY)        # graf závislosti veličinyY na veličiněX
plot(velicinaY ~ velicinaX)      # graf závislosti veličinyY na veličiněX
points(bodX,bodY)        # přidání bodu o souřadnicích [bodX,bodY] do stávajícího grafu
abline(a, b)             # kreslí přímku y = a + b x.
qqnorm(velicina)         # normální QQ graf
qqline(velicina)         # proloží přímku QQ grafem
barplot(table(velicina))                # sloupcový graf pro absolutní četnosti
barplot(prop.table(table(velicina)))    # sloupcový graf pro relativní četnosti
pie(table(velicina))     # koláčový graf pro relativní četnosti
```

**Plot of means** (graf průměrů s "anténami"). `Nvelicina` je numerická, `Fvelicina` je faktorová.

```
prumery <- tapply(Nvelicina, Fvelicina, mean)
N <- nlevels(Fvelicina)
pocty <- table(Fvelicina)
# Co chceme jako "antény"?
# chceme-li vykreslit směrodatné odchylky:
sds<- tapply(Nvelicina, Fvelicina, sd)
# chceme-li vykreslit směrodatné chyby průměrů:
# sds<- tapply(Nvelicina, Fvelicina, sd)/sqrt(pocty)
# chceme-li vykreslit intervaly spolehlivosti (o spolehlivosti 1-alfa):
# alfa <- 0.05
# sds<- tapply(Nvelicina, Fvelicina, sd)/sqrt(pocty)*qt(1-alfa/2,pocty-1)
horni <- prumery + sds
dolni <- prumery - sds
ylim <- range(c(dolni, horni))
plot(1:N, prumery, xlim=c(0.75,N+0.25), ylim=ylim, xaxt="n", pch=16, cex=1.5,
   col="darkgreen", xlab="popis osy x", ylab="popis osy y")
axis(1, at=1:N, labels=levels(Fvelicina))
lines(1:N, prumery, col="darkgreen")
for (i in 1:N){
  lines(c(i, i), c(dolni[i], horni[i]), lty=2, col="red")
  lines(i+c(-0.05, 0.05), rep(dolni[i], 2), lty=2, col="red")
  lines(i+c(-0.05, 0.05), rep(horni[i], 2), lty=2, col="red")
}
```

Přidání **legendy** (příklad se dvěma kategoriemi):

```
legend(umisteni, legend=c("popisek1", "popisek2"), col=("barva1","barva2"),
        pch=("symbol1","symbol2"))
```

Umístění legendy zadáme pomocí bodové souřadnice (např. legend(4,5, legend= ...) umístí legendu do bodu [4,5]), nebo slovně: "topright", "topleft", "bottomright", "bottomleft", "top", "bottom", ...

Všechny výše uvedené grafy lze zkrášlovat pomocí argumentů:

```
xlab="popis_osy_x"      # X label, pojmenování osy x
ylab="popis_osy_y"      # Y label, pojmenování osy y
col="barva"             # color, "barva" nahrad'te anglickým názvem barvy
col=heat.colors(N)      # použije paletu heat.colors s N barvami (další palety viz níže)
xlim="c(od,do)"         # X limits, nastavení rozsahu osy x
ylim="c(od,do)"         # Y limits, nasteavní rozsahu osy y
pch=cislo               # point character (číslo od 0 do 25)
main="Nadpis grafu"     # nadpis grafu
```

Pro sloupcový a koláčový graf se mohou hodit **barevné palety** (`N` značí počet kategorií dané veličiny):

```
terrain.colors(N)
topo.colors(N)
heat.colors(N)
```

Histogram s grafem hustoty normálního rozdělení:

```
hist(velicina,prob=TRUE)
curve(dnorm(x,mean(velicina),sd(velicina)),col="red",add=TRUE)
```

Matice scatterplotů s histogramy na diagonále (příklad pro 3 veličiny, ale může jich být více)

```
library(car)
scatterplotMatrix(~ velicina1 + velicina2 + velicina3, reg.line=lm, smooth=TRUE,
                  diagonal = "histogram")
```

Více grafů v jednom obrázku:

```
par(mfrow=c(m, n))     # grafy budou uspořádány do m řad a n sloupců
```

# 5 Hustoty a distribuční funkce

**Hustoty**: předpona "d" (z anglického *density*) + název rozdělení (syntaxe platí i u diskrétních rozdělení, kde se obvykle označení "hustota" nepoužívá). Argumentem je bod a dále parametry rozdělení.

```
dbinom(x,n,p)       # pravděpodobnost hodnoty x v binomickém rozdělení Bi(n,p)
```

Pozor! U spojitých rozdělení nemá hustota v bodě žádný význam. Příkaz se hodí akorát pro vykreslení (pomocí `curve`, viz histogram s hustotou výše).

```
dunif(x,a,b)                # rovnoměrné rozdělení na intervalu [a,b]
dnorm(x,mu,sigma)           # normální rozdělení N(mu,sigma^2)
dexp(x,lambda)              # exponenciální rozdělení Exp(lambda)
```

**Distribuční funkce**: předpona "p" (z anglického *probability*) + název rozdělení
Hodnota distribuční funkce v bodě `x`:

```
pbinom(x,n,p)               # binomické rozdělení Bi(n,p)
punif(x,a,b)                # rovnoměrné rozdělení na intervalu [a,b]
pnorm(x,mu,sigma)           # normální rozdělení N(mu,sigma^2)
pexp(x,lambda)              # exponenciální rozdělení Exp(lambda)
```

# 6 Posouzení normality

Graficky:

```
qqnorm(velicina)         # normální QQ graf
qqline(velicina)         # proloží přímku QQ grafem
```

nebo (histogram s grafem hustoty normálního rozdělení):

```
hist(velicina,prob=TRUE)
curve(dnorm(x,mean(velicina),sd(velicina)),col="red",add=TRUE)
```

Testem:

```
shapiro.test(velicina)
```

# 7 Interval spolehlivosti

Intervaly spolehlivosti pro střední hodnotu veličiny s normálním rozdělením při neznámém rozptylu. Spolehlivost je číslo mezi 0 a 1, nejčastěji 0.95.

```
t.test(velicina,conf.level=spolehlivost)$conf.int       # oboustranný interval
t.test(velicina,conf.level=spolehlivost,alternative="less")$conf.int     # dolní interval
t.test(velicina,conf.level=spolehlivost,alternative="greater")$conf.int  # horní interval
```

# 8 Testování hypotéz

### Jednovýběrový t-test

```
t.test(velicina,mu=mu0)      # test H0: mu=mu0 proti H1: mu různé od mu0
t.test(velicina,mu=mu0,alternative="less")      # H0: mu=mu0, H1: mu < mu0
t.test(velicina,mu=mu0,alternative="greater")   # H0: mu=mu0, H1: mu > mu0
```

Jinou než 5% hladinu lze nastavit použitím argumentu

```
conf.level=spolehlivost
```

ve funkcích výše, kde `spolehlivost` je 1−`hladina`.

### Jednovýběrový Wilcoxonův test

```
wilcox.test(velicina,mu=mu0)
wilcox.test(velicina,mu=mu0,alternative="greater")
wilcox.test(velicina,mu=mu0,alternative="less")
```

### Znaménkový test

```
U <- sum(velicina > mu0)         # počet kladných znamének
n2 <- sum(velicina != mu)        # úprava počtu pozorování
prop.test(U, n2, alternative="greater")    # znaménkový test
```

### Dvouvýběrový t-test / Welchův test

Mám-li výběry ve dvou různých datových souborech (1. výběr: `data1`, 2. výběr: `data2`). `mu1` a `mu2` jsou střední hodnoty 1. a 2. výběru.

```
t.test(data1$Nvelicina,data2$Nvelicina)    # test H0: mu1=mu2 proti H1: mu1 různé od mu2
t.test(data1$Nvelicina,data2$Nvelicina,alternative="less")   # H0: mu1=mu2, H1: mu1 < mu2
t.test(data1$Nvelicina,data2$Nvelicina,alternative="greater") # H0: mu1=mu2, H1: mu1 > mu2
```

Mám-li výběry určené pomocí kategoriální proměnné `Fvelicina` v datech:

```
t.test(data$Nvelicina~data$Fvelicina)      # test H0: mu1=mu2 proti H1: mu1 různé od mu2
t.test(data$Nvelicina~data$Fvelicina,alternative="less")    # H0: mu1=mu2, H1: mu1 < mu2
t.test(data$Nvelicina~data$Fvelicina,alternative="greater") # H0: mu1=mu2, H1: mu1 > mu2
```

`mu1` a `mu2` jsou střední hodnoty 1. a 2. výběru. Pořadí výběrů je dáno pořadím úrovní kategoriální proměnné `Fvelicina` (lze zjistit pomocí `levels(Fvelicina)`).

Pokud jste nezamítli shodu rozptylů obou výběrů, hodí se přidat do funkce argument `var.equal=TRUE`.

Pro test `H0: mu1=mu2+D` použijte ve funkcích výše argument `mu=D`.

#### F-test shody rozptylů

```
var.test(data1$Nvelicina,data2$Nvelicina)    # pro výběry ve dvou datových souborech
var.test(data$Nvelicina~data$Fvelicina)      # pro výběry určené veličinou Fvelicina
```

#### Dvouvýběrový Wilcoxonův test / Mannův-Whitneyův test

```
wilcox.test(data1$Nvelicina,data2$Nvelicina)  # pro výběry ve dvou datových souborech
wilcox.test(data$Nvelicina~data$Fvelicina)    # pro výběry určené veličinou Fvelicina
```

Jinou než 5% hladinu lze opět nastavit pomocí argumentu `conf.level=spolehlivost`, kde `spolehliovst = 1− hladina`.

#### Analýza rozptylu / jednoduché třídění

```
mod <- aov(Nvelicina~Fvelicina)       # vytvoření modelu při shodných rozptylech
summary(mod)                          # tabulka analýzy rozptylu
TukeyHSD(mod)                         # mnohonásobné porovnání
plot(TukeyHSD(mod))                   # grafické znázornění mnohonásobného porovnání
oneway.test(Nvelicina~Fvelicina)      # analýza rozptylu při nestejných rozptylech
```

Ověřování předpokladů:

```
library(car)                          # potřebná knihovna
leveneTest(Nvelicina~Fvelicina)       # test shody rozptylů
bartlett.test(Nvelicina~Fvelicina)    # test shody rozptylů
shapiro.test(rstandard(mod))          # test normality (provedený na rezidua)
```

#### Kruskalův-Wallisův text

Neparametrická obdoba analýzy rozptylu.

```
kruskal.test(Nvelicina~Fvelicina)
```

## 9 Korelace

```
cor(velicina1,velicina2)            # Pearsonův korelační koeficient
cor.test(velicina1,velicina2)       # test nezávislosti dvou veličin pomocí Pearsonova
                                    # korelačního koeficientu
cor(velicina1,velicina2,method="spearman")       # Spearmanův korelační koeficient
cor.test(velicina1,velicina2,method="spearman")  # test nezávislosti dvou veličin pomocí
                                                 # Spearmanova korelačního koeficientu
```

Vypsání korelační matice: (příklad pro 3 veličiny, ale může jich být více)

```
cor(cbind(velicina1,velicina2,velicina3))
```

## 10 Lineární regrese

```
model<-lm(odezva~regresor1+regresor2)       # vytvoření modelu, regresorů může být více
model<-lm(odezva~regresor1+regresor2+regresor1:regresor2)    # model s interakcí
summary(model)                    # odhady parametrů a další údaje
confint(model, level=0.95)        # intervaly spolehlivosti pro regresní koeficienty

# oveření předpokladů
plot(model)                       # graficky
shapiro.test(rstandard(model))    # ověření normality chyb
library(lmtest)                   # knihovna obsahující bptest
bptest(model)     # ověření homoskedasticity (Breusch-Paganův test)
```

## 11 Analýza kategoriálních dat

Shoda s multinomickým rozdělením:

```
chisq.test(nn,p=pp)     # chi-kvadrát test dobré shody napozorovaných četností nn
                        # s teoretickými pravděpodobnostmi pp
chisq.test(nn,p=pp)$expected      # očekávané teoretické četnosti
```

Možnosti vytvoření kontingenční tabulky (tabulku si nazveme TAB):

```
TAB<-matrix(c(čísla z tabulky),nrow=počet řádků)  # uložení kontingenční tabulky do matice
TAB<-table(factor1,factor2)     # kontingenční tabulka daná dvěma kategoriálními
                                # veličinami z dat
```

Testy nezávilosti dvou kategoriálních veličin:

```
chisq.test(TAB,correct=FALSE)   # chi-kvadrát test nezávislosti bez Yatesovy korekce
chisq.test(TAB)                 # chi-kvadrát test nezávislosti s Yatesovou korekcí
chisq.test(TAB)$expected        # očekávané četnosti
fisher.test(TAB)      # Fisherův faktoriálový test nezávislosti
```

Test symetrie kontingenční tabulky. (Tabulka musí být čtvercová!)

```
mcnemar.test(TAB)     # McNemarův (Bowkerův) test symetrie
```

Test shody pravděpodobností ve dvou binomických rozdělení. Tj. $Y_1 \sim Bi(n_1, p_1)$, $Y_2 \sim Bi(n_2, p_2)$, $H_0 : p_1 = p_2$. Napozorované hodnoty $Y_1$ a $Y_2$ si označme `y1` a `y2`.

```
prop.test(c(y1,y2),c(n1,n2))                  # Raoův skórový test s Yatesovou korekcí
prop.test(c(y1,y2),c(n1,n2),correct=FALSE)    # Raoův skórový test bez Yatesovy korekce
```

Yatesova korekce se používá při malém počtu pozorování.