# ANALÝZA DAT V R

**Podmíněné příkazy a příkazy cyklů**

**Tvorba funkcí**

---

## 1 Podmíněné příkazy

| **RELAČNÍ OPERÁTORY** | |
|---|---|
| == | rovná se |
| != | nerovná se |
| < | menší |
| > | větší |
| <= | menší nebo rovno |
| >= | větší nebo rovno |
| **LOGICKÉ OPERÁTORY** | |
| & | a |
| \| | nebo |

### 1.1 `if`

- syntaxe: `if(podminka) {prikaz}`
- je-li splněna podmínka `podminka`, pak se provede příkaz `prikaz`, v opačném případě se nic neprovede
- *Je (v reálných číslech) definována druhá odmocnina čísla $x$?*

```
> x <- 2
> if (x >= 0) {"Druha odmocnina je definovana."}
[1] "Druha odmocnina je definovana."
> x <- -5
> if (x >= 0) {"Druha odmocnina je definovana."}
```

- nelze použít na vektor (a ani žádné následující podmíněné příkazy)

### 1.2 `if else`

- syntaxe: `if(podminka) {prikaz1} else {prikaz2}`

– je-li splněna podmínka `podminka`, pak se provede příkaz `prikaz1`, v opačném případě se provede příkaz `prikaz2`

– *Je (v reálných číslech) definována druhá odmocnina čísla x?*

```
> x <- 2
> if (x >= 0) {"Druha odmocnina je definovana."} else {"Druha odmocnina neni definovana."}
[1] "Druha odmocnina je definovana."
> x <- -5
> if (x >= 0) {"Druha odmocnina je definovana."} else {"Druha odmocnina neni definovana."}
[1] "Druha odmocnina neni definovana."
```

### 1.3 ifelse

– alternativa k if else příkazu

– syntaxe: `ifelse(podminka, prikaz1, prikaz2)`

– je-li splněna podmínka `podminka`, pak se provede příkaz `prikaz1`, v opačném případě se provede příkaz `prikaz2`

– *Je (v reálných číslech) definována druhá odmocnina čísla x?*

```
> x <- 2
> ifelse(x >= 0, "Druha odmocnina je definovana.", "Druha odmocnina neni definovana.")
[1] "Druha odmocnina je definovana."
> x <- -5
> ifelse(x >= 0, "Druha odmocnina je definovana.", "Druha odmocnina neni definovana.")
[1] "Druha odmocnina neni definovana."
```

### 1.4 if else if else ... else

– pokud rozlišujeme více než dva stavy

– syntaxe:

```
if (podminka1) {prikaz1} else if
(podminka2) {prikaz2} else if
(podminka3) {prikaz3} else if
⋮
else {prikazk}
```

– je-li splněna podmínka `podminka1`, pak se provede příkaz `prikaz1`, je-li splněna podmínka `podminka2`, pak se provede příkaz `prikaz2`, atd., není-li splněna ani jedna z uvedených podmínek, provede se příkaz `prikazk`

– *Je číslo x kladné, záporné, nebo nula?*

```
> x <- -0.2
> if (x > 0) {"Kladne cislo."} else if
+ (x < 0) {"Zaporne cislo."} else
+ {"Nula."}
[1] "Zaporne cislo."
```

## 2 Příkazy cyklů

– pro opakované provedení příkazů

### 2.1 `for`

– používáme, pokud víme počet opakování

– syntaxe: `for (promenna in rozsah) {prikazy}`

`promenna` ... indexační proměnná

`rozsah` ... hodnoty proměnné `promenna` (zpravidla vektor)

`prikazy` ... posloupnost příkazů, které mají být vykonány; ve složených závorkách; odděleny středníkem nebo každý příkaz na jeden řádek (doporučeno)

– vytištění hodnoty pomocí příkazu `print()`

– *Určete druhé mocniny přirozených čísel od 1 do 10.*

```
> for (i in 1:10) {print(i*i)}
```

### 2.2 `while`

– používáme, pokud nevíme počet opakování - závisí na jisté podmínce

– syntaxe: `while (podminka) {prikazy}`

– dokud je splněna podmínka `podminka`, provádějí se příkazy `prikazy`

– *Vypište takové druhé mocniny přirozených čísel, které jsou menší než 650.*

```
> i <- 1
> while (i**2 < 650) {print(i**2); i <- i+1}
```

### 2.3 repeat

– alternativa k `while`

– pro ukončení se používá příkaz `break`, kde v závorce je uvedena ukončovací podmínka

– syntaxe:

```
repeat {prikazy
if (podminka) {break}}
```

– dokud není splněna podmínka `podminka`, provádějí se příkazy `prikazy`

– *Vypište takové druhé mocniny přirozených čísel, které jsou menší než 650.*

```
> i <- 1
> repeat {print(i**2)
+ i <- i+1
+ if (i**2>=650) {break}
```

Poznámka: For cyklus lze obejít použitím funkcí `apply()`, `lapply()`, `sapply()` a `tapply()`. `apply()` se používá pro vektory, matice a pole a tato funkce má tři argumenty: první je název struktury, druhý je určení, jak se aplikuje (u matic může být po řádcích (1) a po sloupcích (2), u polí další dimenze), třetí je název aplikované funkce. Pro seznam se používají funkce `lapply()` a `sapply()` (první argument je název seznamu, druhý je aplikovaná funkce). Pro datovou tabulku se používá `tapply()`.

## 3 Tvorba funkcí

– v R možnost vytvářet vlastní funkce

– syntaxe funkce je vždy ve tvaru

`nazev <- function(arg1, arg2, ...){telo}`

`nazev` ... název funkce (vyhýbat se názvům již existujících funkcí)

`arg1, arg2` ... posloupnost vstupních hodnot, odděleny čárkou

`telo` ... posloupnost příkazů, které mají být vykonány; ve složených závorkách; odděleny středníkem nebo každý příkaz na jeden řádek (doporučeno)

- funkce lze psát rovnou do konzole nebo ji lze vytvořit v editoru, tj. *File* → *New File* → *R Script*
- výstupní hodnota je vrácena pomocí příkazu `return()`
- *Vytvořte funkci pro výpočet objemu kvádru, jsou-li zadány délky stran* $a, b, c$.

```
> kvadr.objem <- function(a,b,c){
+ V <- a*b*c
+ return(V)}
```

- pokud je výstupních hodnot více, je potřeba sdružit tyto hodnoty do seznamu, tj. `return(list=c(...))`
- *Vytvořte funkci pro výpočet objemu a povrchu kvádru, jsou-li zadány délky stran* $a, b, c$.

```
> kvadr <- function(a,b,c){
+ V <- a*b*c
+ S <- 2*(a*b+b*c+a*c)
+ return(list=c(objem=V,povrch=S))}
```

- zapsáním názvu funkce do konzole je vypsána definice funkce

```
> kvadr
```

- zapsáním názvu funkce do konzole včetně konkrétních vstupních hodnot v závorce je vypsán výsledek/výsledky funkce

```
> kvadr(4,8,3)
```

- pokud je vytvořena funkce v editoru, je potřeba ji uložit jako soubor s příponou `.R` a následně spustit pomocí příkazu `source()`, kde v závorce je uveden název funkce v uvozovkách (POZOR! Nejprve je potřeba nastavit správný adresář.)
- příkaz `warning()`vytiskne upozorňující hlášku

```
> kvadr <- function(a,b,c){
+ V <- a*b*c
+ S <- 2*(a*b+b*c+a*c)
+ warning("Delky stran museji byt kladna cisla!")
+ return(list=c(objem=V,povrch=S))}
```

- příkaz `stop()` zastaví provádění příkazů a vytiskne chybovou hlášku, když je splněna podmínka

```
> kvadr <- function(a,b,c){
```

```
+ if (a<=0 | b<=0 | c<=0) stop("Delky stran museji byt kladna cisla!") else
+ V <- a*b*c
+ S <- 2*(a*b+b*c+a*c)
+ return(list=c(objem=V,povrch=S))}
```

**Příklad 1**

Napište funkci, která pro libovolný vstupní vektor vrátí průměr, směrodatnou odchylku a rozpětí. Nepoužívejte žádné implementované statistické funkce typu `mean()` apod.!

**Příklad 2**

Napište funkci, která pro libovolný vstupní vektor vrátí variační koeficient. Uvědomte si, pro které hodnoty má smysl počítat variační koeficient a ošetřete to během definování funkce!

**Příklad 3**

Napište funkci, která pro libovolný vstupní vektor vrátí počet čísel dělitelných 3. Ošetřete, aby na vstupu byla jen přirozená čísla!

**Příklad 4**

Napište funkci, která pro libovolnou vstupní hodnotu vrátí její dělitele. Ošetřete, aby na vstupu bylo jen přirozené číslo!