Datové typy v R

Kdykoli programujete v jakémkoli jazyce, tak potřebujete některá data někam ukládat. Počítač si vámi zadaná data uloží do paměti. Různá data ale potřebují různé množství paměti. A aby počítač věděl, kolik paměti bude potřebovat, existují tzv. *datové typy*.

Vysvětlíme si to na příkladech.

`"Pět"` je *řetězec znaků*, text. Všimněte si, že jej schválně píšeme do uvozovek, aby bylo jasné, že jde o text. Takže `"5"` je taky text!

Naproti tomu je 5 číslo. To do uvozovek nedáváme. Vlastně jde o *celé číslo*. Když napíšeme `5.2`, jde o *reálné číslo* (a všimněte si desetinné *tečky*).

V jazyce R existují tyto datové typy:

- **logical** – logická hodnota (TRUE nebo FALSE)
- **numeric** – reálné číslo (`5.23675`), má podtypy:
  - **integer** – celé číslo (5L, u celých čísel se na konci píše L, aby bylo rozlišeno od reálných čísel)
  - **double** – reálné číslo (`5.23675`)
- **complex** – komplexní číslo (5+2i... no, ehm, pamatujete na matematiku? Více ve Wikipedii.)
- **character** – textový řetězec (`"pět"` nebo `"5"`, vzpomeňte na uvozovky)
- **raw** – binární data (`61 68 6f 6a`, což je binárně zapsané „ahoj“: každý znak má svůj dvoumístný kód)

Zkusíme si sepsat ještě přehlednou tabulku a už se vrhneme na jednotlivé typy podrobněji. Předem můžeme prozradit, že tyto prvky jsou samy o sobě ve své podstatě základní *vektory*, což bude vysvětleno v další kapitole o datových strukturách.

mód	typ	popis
logical	logical	logická hodnota (TRUE nebo FALSE)
numeric	integer	celé číslo (např. 5L)
	double	reálné číslo (např. `5.2435`)
complex	complex	komplexní číslo (např. 5+2i)
character	character	textový řetězec (např. `"pět švestek"`
raw	raw	binární data (např. `61 68 6f 6a`)

Logická hodnota (logical)

Logická hodnota určuje pravdu (TRUE) nebo *poněkud diplomaticky* nepravdu (FALSE). Využívá se například k vyhodnocování příkazů, porovnání (je 5 větší než 2?) a podobně. V jazyce R existuje pravda i nepravda i ve zkrácené verzi (T a F).

Zda je hodnota typu *logical*, zjistíme pomocí příkazu `is.logical()`, který nám napíše TRUE nebo FALSE (vidíte?). Nebo třídu zjistíme pomocí `class()`, což nám napíše výsledek textem.

```
# je TRUE logická hodnota?
is.logical(TRUE)
#> [1] TRUE

# do jaké třídy patří FALSE?
class(FALSE)
#> [1] "logical"

# do jaké třídy patří "TRUE"?
class("TRUE")
#> [1] "character"

# poznali jste chyták? jsou tam uvozovky, takže je to textový řetězec

# do jaké třídy patří T?
class(T)
#> [1] "logical"

# do jaké třídy patří F?
class(F)
#> [1] "logical"
```

Numerická hodnota (numeric)

Numerickou hodnotu představuje libovolné reálné číslo. Např.. `5, -13.8, 4.5578e15`.

Zda je hodnota typu *numeric*, zjistíme pomocí příkazu `is.numeric()`, který nám napíše TRUE nebo FALSE (logickou hodnotu uvedenou výše). Nebo třídu zjistíme pomocí známého příkazu `class()`, který nám napíše výsledek textem. Mrkneme na to prakticky.

```r
# je -13.8 numerická hodnota?
is.numeric(-13.8)
#> [1] TRUE

# jakou třídu představuje -13.8?
class(-13.8)
#> [1] "numeric"

# jakou třídu představuje nekonečno (Inf)?
class(Inf)
#> [1] "numeric"

# jakou třídu představuje "5"?
class("5")
#> [1] "character"

# hahaha, chyták, jsou tam uvozovky, je to tedy text :-)
```

Celé číslo (integer)

Integer je libovolné celé číslo uložené s danou přesností. Za číslem je v jazyce R napsáno velké L, aby se celé číslo odlišilo od reálného čísla. Např. `5L, 13L, -5L`.

⚠ V R mají celá čísla pouze 16-b přesnost. Pro práci s velkými celými čísly nutné balíčky *gmp* či *int64* (zvýší bitovou přesnost uložených celých čísel).

Zda je hodnota typu *integer*, zjistíme opět pomocí příkazů `is.integer()` a `class()`.

```r
# je číslo -13L celé číslo?
is.integer(-13L)
#> [1] TRUE

# do jaké třídy patří -13L?
class(-13L)
#> [1] "integer"

# je číslo -13 celé číslo?
is.integer(-13)
#> [1] FALSE

# jak to? co to teda je?
class(-13)
#> [1] "numeric"

# aha, je to reálné číslo, protože mu na konci chybí velké L
```

Pokud chceme změnit celé číslo na reálné, použijeme příkaz `as.numeric()`.

```r
# vezmu si proměnnou x a přiřadím jí hodnotu 5L
x <- 5L

# co je x?
class(x)
#> [1] "integer"

# převedu celé číslo na reálné číslo a přiřadím ho proměnné y
y <- as.numeric(x)

# co je y?
class(y)
#> [1] "numeric"

# fakt je to pořád 5?
y
#> [1] 5

# fakt.
```

Komplexní číslo (complex)

Komplexní číslo je libovolné x, pro které platí, že x = a + bi, kde a, b jsou reálná čísla a $i^2 = -1$. Pamatujete? Kdyžtak pročtěte Wikipedii.

Jako typ *complex* se označují komplexní čísla, tedy např. `1 + 2i`.

Zda je hodnota typu *complex*, zjistíme opět pomocí příkazů `is.complex()` a `class()`.

```r
# je 1+2i komplexní číslo?
is.complex(1 + 2i)
#> [1] TRUE

# do jaké třídy patří 0+1i?
class(0 + 1i)
#> [1] "complex"

# do jaké třídy patří výsledek odmocniny z (-1+0i)?
class(sqrt(-1 + 0i))
#> [1] "complex"

# a do jaké třídy patří odmocnina z -1?
class(sqrt(-1))
Warning message:
```

```
In sqrt(-1) : NaNs produced

# haha, -1 nelze odmocňovat, není to komplexní číslo
```

Textový řetězec (character)

Textový řetězec je libovolná sekvence znaků uzavřená mezi jednoduchými či dvojitými uvozovkami, např. `"ahoj"`, `"HJKU78789ddkj y"`, `'pět'`, `"5"`, `'TRUE'`, `""`.

Zda je hodnota typu *character*, zjistíme opět pomocí příkazů `is.character()` a `class()`.

```
# je "ahoj" text?
is.character("ahoj")
#> [1] TRUE

# jaký typ je "123"?
class("123")
#> [1] "character"

# jasně, jsou tam uvozovky! je to text, i když obsahuje číslo :-)

# tohle bude číslo, protože to je bez uvozovek
class(123)
#> [1] "numeric"

# je nekonečno číslo?
is.numeric(Inf)
#> [1] TRUE

# je "Inf" číslo?
is.numeric("Inf")
#> [1] FALSE

# ne, je to v uvozovkách,  je to tedy text
class("Inf")
#> [1] "character"

# ha, a teď si dáme špek
class(class(5))
#> [1] "character"

# jo, nejprve se vyhodnotí vnitřek:
# class(5) je "numeric"
# ... a pak vnějšek:
# class("numeric") je "character"
# OK!
```

Na textový řetězec lze převést libovolnou jinou hodnotu pomocí příkazu `as.character()`.

```
# přiřadím proměnné x hodnotu 5
x <- 5

# co je x?
x
#> [1] 5

class(x)
#> [1] "numeric"

# převedu na řetězec a uložím do y
y <- as.character(x)

# co je y?
y
#> [1] "5"

class(y)
#> [1] "character"
```

NA, NULL, NaN

- **NA** je hodnota typu Not Available, obvykle chybějící hodnota.
- **NULL** je null object, používá se pro bezhodnotovou inicializaci objektu... no prostě když už víme, jak se proměnná jmenuje, ale ještě jí nechceme přiřadit žádnou opravdovou hodnotu, dáme NULL.
- **NaN** je hodnota typu Not a Number, obvykle nevyjádřitelný výsledek matematické operace.

```
# zkusíme odmocninu z -1
sqrt(-1)
#> [1] NaN

# kdyby vás napadlo dělit nulou, tak bacha: výsledkem je nekonečno!
1/0
#> [1] Inf

is.nan(1/0)
#> [1] FALSE

# NaN je číslo
class(NaN)
#> [1] "numeric"

# NA je logická hodnota, ale není to FALSE
class(NA)
#> [1] "logical"
```

```r
FALSE == NA
#> [1] NA

# NaN patří do NA
is.na(NaN)
#> [1] TRUE

# ale NA nepatří do NaN
is.nan(NA)
#> [1] FALSE

# už vám vybuchuje hlava? :-P
```

S těmito zvláštními typy se budeme setkávat třeba v chybových hlášeních, když proměnná nabude hodnoty, se kterou jsme nepočítali.