

Úvod do R

Petr Marek

23. - 24. listopadu 2021

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Důležité odkazy

- R Project - <https://www.r-project.org/>
- RStudio - <https://www.rstudio.com/products/RStudio/>
- Quick R - <http://www.statmethods.net/>
- Stack Overflow - <http://stackoverflow.com/>
- R-bloggers - <https://www.r-bloggers.com/>
- contact@petrmarek.eu

RStudio

- **RStudio != R**
- **Dle <https://www.rstudio.com/products/RStudio>:**
 - **RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.**
 - **RStudio is available in open source and commercial editions and runs on the desktop (Windows, Mac, and Linux).**

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Kalkulačka

```
> 1 + 2
```

```
[1] 3
```

```
> 3 ^ 2
```

```
[1] 9
```

```
> 5 * 5
```

```
[1] 25
```

```
> sqrt(2)
```

```
[1] 1.41
```

```
> 2 ^ (1/2)
```

```
[1] 1.41
```

Kalkulačka

+	sčítání
-	odečítání
*	násobení
/	dělení
^	mocnění
%/%	celočíselné dělení
%%	modulo

Nápověda

- Komplexní nápověda pomocí `?` , popř. `help()`

```
> ?sqrt
```

```
> help(sqrt)
```

- Zadání názvu funkce vypíše její kód

```
> sqrt
```

```
function (x) .Primitive("sqrt")
```


Volání funkcí

```
> seq(1, 5)
```

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

```
> seq(from = 1, to = 5)
```

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

```
> seq(1, to = 5)
```

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

Základní matematické funkce

<code>abs</code>	absolutní hodnota
<code>sqrt</code>	druhá odmocnina
<code>log</code>	přirozený logaritmus
<code>log10</code>	dekadický logaritmus
<code>exp</code>	exponenciální funkce
<code>cos</code> , <code>sin</code> , <code>tan</code>	trigonometrické funkce

Základy

Úkol

- Zobrazte nápovědu k funkci `sum`
- Spočítejte $\sqrt{3}$
- Spočítejte 3^3
- Spočítejte $\sqrt[3]{3}$
- Spočítejte zbytek po celočíselném dělení 743 číslem 3

Základy

Řešení

```
> ?sum
```

```
> sqrt(3)
```

```
[1] 1.73
```

```
> 3 ^ 3
```

```
[1] 27
```

```
> 3 ^ (1 / 3)
```

```
[1] 1.44
```

```
> 743 %% 3
```

```
[1] 2
```

Proměnné

```
> x = 5
```

```
> x + 5
```

```
[1] 10
```

```
> x
```

```
[1] 5
```

```
> x = x + 5
```

```
> x
```

```
[1] 10
```

```
> (y = 3)
```

```
[1] 3
```

Proměnné

Alternativní zápis

```
> x <- "Tento zápis nebudeme používat"  
> "Tento také ne" -> x
```

Proměnné

Úkol

- Vytvořte proměnnou x s hodnotou $\sqrt{3}$
- Vytvořte proměnnou y s hodnotou $\frac{\sqrt{3^3}}{\sqrt[3]{11}}$
- Vytvořte proměnnou z s logickou hodnotou `TRUE` pokud je x větší než y a `FALSE` pokud je menší nebo rovna

Proměnné

Řešení

```
> x = sqrt(3)
```

```
> x
```

```
[1] 1.73
```

```
> y = sqrt(3 ^ 3) / (11 ^ (1 / 3))
```

```
> y
```

```
[1] 2.34
```

```
> z = x > y
```

```
> z
```

```
[1] FALSE
```


Logické operátory

```
> 1 < 2
```

```
[1] TRUE
```

```
> 1 > 2
```

```
[1] FALSE
```

```
> 1 == 2 # jsou stejné?
```

```
[1] FALSE
```

```
> 1 != 2 # jsou jiné?
```

```
[1] TRUE
```

Logické operátory

Negace

> !TRUE

```
[1] FALSE
```

> !FALSE

```
[1] TRUE
```

> !(1 == 2)

```
[1] TRUE
```

> !1

```
[1] FALSE
```

Logické operátory

Kombinace podmínek - a

> (1 < 2) & (3 < 4)

```
[1] TRUE
```

> (1 < 1) & (3 < 4)

```
[1] FALSE
```

> (1 < 1) & (3 < 4) & FALSE

```
[1] FALSE
```

Logické operátory

Kombinace podmínek - nebo

> (1 < 2) | (3 < 4)

```
[1] TRUE
```

> (1 < 1) | (3 < 4)

```
[1] TRUE
```

> (1 < 1) | (3 < 3)

```
[1] FALSE
```

Logické operátory

Kombinace podmínek - mix

> (5 > 4) & (3 < 2)

```
[1] FALSE
```

> (1 < 2) | ((5 > 4) & (3 < 2))

```
[1] TRUE
```

> !!!!TRUE

```
[1] TRUE
```

Logické operátory

Úkol

- **Jakou hodnotu nabývají následující výrazy?**

> TRUE & FALSE

> TRUE | FALSE

> (5 > 4) | ((3 < 3) | (1 >= 2))

> (5 > 4) & ((3 < 3) | (1 >= 2))

> (5 > 4) | ((3 < 3) & (1 >= 2))

> (5 > 4) & ((3 < 3) & (1 >= 2))

Logické operátory

Řešení

> TRUE & FALSE

```
[1] FALSE
```

> TRUE | FALSE

```
[1] TRUE
```

Logické operátory

Řešení

> (5 > 4) | ((3 < 3) | (1 >= 2))

```
[1] TRUE
```

> (5 > 4) & ((3 < 3) | (1 >= 2))

```
[1] FALSE
```

> (5 > 4) | ((3 < 3) & (1 >= 2))

```
[1] TRUE
```

> (5 > 4) & ((3 < 3) & (1 >= 2))

```
[1] FALSE
```


Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

Vektor

Matice

List

Data frame

Faktor

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Funkce str

- **Funkce** `str()`, viz. `?str` .

```
> str(1)
```

```
num 1
```

```
> str('a')
```

```
chr "a"
```

```
> str(TRUE)
```

```
logi TRUE
```

Elementární druhy dat 1/2

```
> 1 # num / numeric
```

```
[1] 1
```

```
> "ahoj" # chr / character
```

```
[1] "ahoj"
```

```
> 'ahoj' # chr / character
```

```
[1] "ahoj"
```

```
> NULL # NULL
```

```
NULL
```

Elementární druhy dat 2/2

> TRUE # logi / logical

```
[1] TRUE
```

> FALSE # logi / logical

```
[1] FALSE
```

> NA # logi / logical

```
[1] NA
```

> 3+2i # cplx / complex

```
[1] 3+2i
```

> NaN # num / numeric

```
[1] NaN
```

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

Vektor

Matice

List

Data frame

Faktor

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Vektor

Konstrukce 1/2

```
> 10
```

```
[1] 10
```

```
> c(1, 3, 5, 7)
```

```
[1] 1 3 5 7
```

```
> c(10, c(1, 3, 5, 7))
```

```
[1] 10 1 3 5 7
```

```
> rep(1, 5)
```

```
[1] 1 1 1 1 1
```

Vektor

Konstrukce 2/2

```
> seq(0, 5, by = 0.5)
```

```
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
> seq(0, 100, length.out = 0.5)
```

```
[1] 0
```

```
> 1:10
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
> 5:3
```

```
[1] 5 4 3
```

Vektor

Datové typy 1/2

```
> c(1, 2, 3)
```

```
[1] 1 2 3
```

```
> c('A', 'B', 'C')
```

```
[1] "A" "B" "C"
```

```
> c(TRUE, FALSE, TRUE)
```

```
[1] TRUE FALSE TRUE
```


Vektor

Datové typy 2/2

```
> c(1, 'B', TRUE)
```

```
[1] "1"      "B"      "TRUE"
```

```
> c(5, TRUE, FALSE)
```

```
[1] 5 1 0
```

```
> c('D', TRUE, FALSE)
```

```
[1] "D"      "TRUE"   "FALSE"
```

Vektor

Základní operace - mutace

```
> 1:5 > 2
```

```
[1] FALSE FALSE TRUE TRUE TRUE
```

```
> c(1, 2, 3) + 3
```

```
[1] 4 5 6
```

```
> c(1, 2, 3) + c(1, 2, 3)
```

```
[1] 2 4 6
```

```
> c(1, 2, 3) * 3
```

```
[1] 3 6 9
```

Vektor

Úkol

- Vypište uplynulé roky 21. století
- Vytvořte následující proměnné:
 - `x` obsahující deset čísel mezi 1 a 2
 - `y` rovnu druhé odmocnině ze dvou
 - `z` obsahující patnáct nul (funkce `rep`)
 - `total` jež bude obsahovat hodnoty `x`, `y` a `z`
- Zjistěte počet prvků proměnné `total` (funkce `length`), spočítejte průměrnou hodnotu (funkce `mean`) a součet (funkce `sum`)
- Vytvořte proměnnou `halves`, která obsahuje hodnoty odpovídající polovinám hodnot proměnné `total`

Vektor

Řešení

```
> 2000:2020
```

```
[1] 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009  
    2010 2011 2012 2013 2014  
16] 2015 2016 2017 2018 2019 2020
```

```
> x = seq(1, 2, length.out = 10)
```

```
> y = sqrt(2)
```

```
> z = rep(0, 15)
```

```
> total = c(x, y, z)
```

Vektor

Základní operace - přístup k prvkům 1/4

```
> x = 5:10
```

```
> x
```

```
[1] 5 6 7 8 9 10
```

```
> x[1]
```

```
[1] 5
```

```
> x[c(1, 3)]
```

```
[1] 5 7
```

```
> x[-1]
```

```
[1] 6 7 8 9 10
```

Vektor

Základní operace - přístup k prvkům 2/4

```
> x = 5:10
```

```
> x[c(FALSE, FALSE, FALSE, TRUE, TRUE, TRUE)]
```

```
[1] 8 9 10
```

```
> x > 7
```

```
[1] FALSE FALSE FALSE TRUE TRUE TRUE
```

```
> x[x > 7]
```

```
[1] 8 9 10
```

Vektor

Základní operace - přístup k prvkům 3/4

```
> x = 5:10  
> x.halves = x / 2  
> x.halves
```

```
[1] 2.5 3.0 3.5 4.0 4.5 5.0
```

```
> x.is.even = (x %% 2 == 0)  
> x.is.even
```

```
[1] FALSE TRUE FALSE TRUE FALSE TRUE
```

```
> even.from.x = x[x.is.even]  
> even.from.x
```

```
[1] 6 8 10
```

Vektor

Základní operace - přístup k prvkům 3/4

```
> x = 5:10
```

```
> x
```

```
[1] 5 6 7 8 9 10
```

```
> x = x[-1]
```

```
> x
```

```
[1] 6 7 8 9 10
```

```
> x[3] = 0
```

```
> x
```

```
[1] 6 7 0 9 10
```


Vektor

Popis

```
> x = 5:10
```

```
> length(x)
```

```
[1] 6
```

```
> mean(x)
```

```
[1] 7.5
```

```
> median(x)
```

```
[1] 7.5
```

```
> summary(x)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
5.00	6.25	7.50	7.50	8.75	10.00

Vektor

Úkol

- Vytvořte vektor `numbers` obsahující čísla od 1 do 100
- Vypište následující
 - hodnoty z `numbers` větší než 50
 - lichá čísla z `numbers`
 - průměrnou hodnotu lichých čísel z `numbers`
 - počet čísel větších než 34 a menších než 66 z `numbers`
 - součet čísel větších než 90 nebo menších než 10 z `numbers`
- Změňte dvě prostřední hodnoty z `numbers` na 0 a spočítejte průměr z `numbers`

Vektor

Řešení

```
> numbers = 1:100
```

```
> numbers[numbers > 50]
```

```
[1] 51 52 53 54 55 56 57 58 59 60 61 62  
    63 64 65 66 67 68 69  
20] 70 71 72 73 74 75 76 77 78 79 80 81  
    82 83 84 85 86 87 88  
39] 89 90 91 92 93 94 95 96 97 98 99 100
```

Vektor

Řešení

```
> numbers[numbers %% 2 == 1]
```

```
[1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33  
    35 37 39 41 43 45 47 49  
26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83  
    85 87 89 91 93 95 97 99
```

```
> mean(numbers[numbers %% 2 == 1])
```

```
[1] 50
```

Vektor

Řešení

```
> length(numbers[numbers > 34 & numbers < 66])
```

```
[1] 31
```

```
> sum(numbers[numbers > 90 | numbers < 10])
```

```
[1] 1000
```

Vektor

Řešení

```
> length(numbers)
```

```
[1] 100
```

```
> numbers[50] = 0
```

```
> numbers[51] = 0
```

```
> mean(numbers)
```

```
[1] 49.5
```

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

Vektor

Matice

List

Data frame

Faktor

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Matice

Konstrukce 1/2

```
> x = matrix(1:6, ncol = 3, nrow = 2)
```

```
> x
```

```
[ ,1] [ ,2] [ ,3]  
  1    3    5  
  2    4    6
```

```
> str(x)
```

```
int [1:2, 1:3] 1 2 3 4 5 6
```

```
> dim(x)
```

```
[1] 2 3
```

```
> c(nrow(x), ncol(x))
```

```
[1] 2 3
```


- **column bind**

```
> cbind(c(1, 2, 3), c(1, 2, 3))
```

```
[,1] [,2]  
  1   1  
  2   2  
  3   3
```

- **row bind**

```
> rbind(c(1, 2, 3), c(1, 2, 3))
```

```
[,1] [,2] [,3]  
  1   2   3  
  1   2   3
```

Matice

Přístup k prvkům 1/2

```
> x = rbind(c(1, 2, 3), c(1, 2, 3))
```

```
> x
```

```
 [ ,1] [ ,2] [ ,3]  
      1     2     3  
      1     2     3
```

```
> x[1,1]
```

```
[1] 1
```

```
> x[2,]
```

```
[1] 1 2 3
```

```
> x[,1]
```

```
[1] 1 1
```

Matice

Přístup k prvkům 2/2

```
> x = rbind(c(1, 2, 3), c(1, 2, 3))
```

```
> x
```

```
[,1] [,2] [,3]  
  1   2   3  
  1   2   3
```

```
> x[-1,]
```

```
[1] 1 2 3
```

```
> x[1, 1] = 0
```

```
> x
```

```
[,1] [,2] [,3]  
  0   2   3  
  1   2   3
```

Matice

Mutace

```
> x = rbind(c(1, 2, 3), c(1, 2, 3))
```

```
> x * 3
```

```
[,1] [,2] [,3]  
 3    6    9  
 3    6    9
```

```
> x %*% t(x)
```

```
[,1] [,2]  
14   14  
14   14
```

Malice

Funkce

<code>dim</code>	rozměry matice
<code>nrow</code>	počet řádků matice
<code>ncol</code>	počet sloupců matice
<code>%*%</code>	maticové násobení
<code>t</code>	transpozice matice
<code>det</code>	determinant matice
<code>eigen</code>	vlastní čísla a vlastní vektory matice
<code>diag</code>	extrakce diagonály matice

Matice

Apply 1/2

Funkce apply umožňuje pracovat podle řádků (1) nebo sloupců (2)

```
> x = rbind(c(1, 2, 3), c(1, 2, 3))
```

```
> x
```

```
[,1] [,2] [,3]  
  1    2    3  
  1    2    3
```

```
> apply(x, 1, sum) # sum pro každý řádek
```

```
[1] 6 6
```

```
> apply(x, 2, sum) # sum pro každý sloupec
```

```
[1] 2 4 6
```

Matice

Apply 2/2

```
> x = rbind(c(1, 2, 3), c(1, 2, 3))
```

```
> x
```

```
[,1] [,2] [,3]  
  1    2    3  
  1    2    3
```

```
> sums = apply(x, 2, sum)
```

```
> sums                                # Součty sloupců
```

```
[1] 2 4 6
```

```
> x[, sums > 3]                        # Sloupce se součtem větším než 3
```

```
[,1] [,2]  
  2    3  
  2    3
```

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

Vektor

Matice

List

Data frame

Faktor

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

List

Motivace

- **Vektory jsou atomické**
- **Prvky vektoru musí být stejného typu**
- **Listy umožňují složitější datovou strukturu**
- **Listy jsou často návratovou hodnotou funkcí**

List

Konstrukce

```
> our.list = list("name" = c("Paul", "John", "Joe"),  
+               "age" = c(15, 40, 30),  
+               "weight" = c(60, 90, 80),  
+               "single" = c(FALSE, TRUE, TRUE))  
> our.list
```

```
$name  
[1] "Paul" "John" "Joe"  
  
$age  
[1] 15 40 30  
  
$weight  
[1] 60 90 80  
  
$single  
[1] FALSE TRUE TRUE
```

List

Struktura

```
> str(our.list)
```

```
List of 4  
 $ name  : chr [1:3] "Paul" "John" "Joe"  
 $ age   : num [1:3] 15 40 30  
 $ weight: num [1:3] 60 90 80  
 $ single: logi [1:3] FALSE TRUE TRUE
```

```
> str(our.list['age'])
```

```
List of 1  
 $ age: num [1:3] 15 40 30
```

```
> str(our.list[['age']])
```

```
num [1:3] 15 40 30
```

List

Přístup k hodnotám 1/2

```
> our.list$name      # vektor
```

```
[1] "Paul" "John" "Joe"
```

```
> our.list['name']  # list
```

```
$name  
[1] "Paul" "John" "Joe"
```

```
> our.list[['name']] # vektor
```

```
[1] "Paul" "John" "Joe"
```

List

Přístup k hodnotám 2/2

```
> our.list[1]      # list
```

```
$name  
[1] "Paul" "John" "Joe"
```

```
> our.list[[1]]   # vektor
```

```
[1] "Paul" "John" "Joe"
```

```
> our.list[c(1, 2)] # list
```

```
$name  
[1] "Paul" "John" "Joe"  
  
$age  
[1] 15 40 30
```

List

Lapply

Funkce `lapply` **je obdobou maticového** `apply` **pro list**

```
> lapply(our.list, length)
```

```
$name
```

```
[1] 3
```

```
$age
```

```
[1] 3
```

```
$weight
```

```
[1] 3
```

```
$single
```

```
[1] 3
```

List

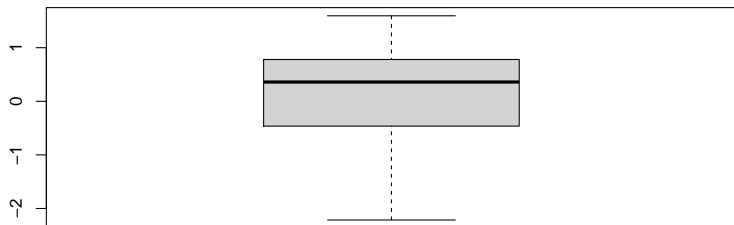
Jako návratová hodnota 1/2

```
> gaussian.random = rnorm(20)
```

```
> head(gaussian.random, 7)
```

```
[1] -0.626  0.184 -0.836  1.595  0.330 -0.820  0.487
```

```
> result = boxplot(gaussian.random)
```



List

Jako návratová hodnota 2/2

```
> str(result)
```

```
List of 6
 $ stats: num [1:5, 1] -2.215 -0.463 0.36 0.78 1.595
 $ n      : num 20
 $ conf  : num [1:2, 1] -0.0795 0.7989
 $ out   : num(0)
 $ group: num(0)
 $ names: chr "1"
```

```
> result$n
```

```
[1] 20
```


Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

Vektor

Matice

List

Data frame

Faktor

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Data frame

Motivace

- **Struktura odpovídající databázové tabulce**
- **Sloupečky různých typů - řetězce, čísla atd.**
- **Sloupce = proměnné**
- **Řádky = záznamy/pozorování**

Data frame

Konstrukce

```
> our.data.frame = data.frame(  
+   "name" = c("Paul", "John", "Joe"),  
+   "age" = c(15, 40, 30),  
+   "weight" = c(60, 90, 80),  
+   "single" = c(FALSE, TRUE, TRUE),  
+   stringsAsFactors = FALSE  
+ )  
> our.data.frame
```

name	age	weight	single
Paul	15	60	FALSE
John	40	90	TRUE
Joe	30	80	TRUE

Data frame

Struktura

```
> str(our.data.frame)
```

```
'data.frame':^^I3 obs. of 4 variables:  
 $ name   : chr  "Paul" "John" "Joe"  
 $ age    : num  15 40 30  
 $ weight: num  60 90 80  
 $ single: logi  FALSE TRUE TRUE
```

```
> dim(our.data.frame)
```

```
[1] 3 4
```

```
> c(nrow(our.data.frame), ncol(our.data.frame))
```

```
[1] 3 4
```

Data frame

Přístup k hodnotám

```
> our.data.frame$age
```

```
[1] 15 40 30
```

```
> our.data.frame[, 'age']
```

```
[1] 15 40 30
```

```
> our.data.frame[2, ]
```

```
name age weight single
John  40     90   TRUE
```

```
> our.data.frame[, 2]
```

```
[1] 15 40 30
```

```
> our.data.frame[2, 2]
```

```
[1] 40
```

Data frame

Mazání sloupce

```
> our.data.frame$age = NULL  
> our.data.frame
```

name	weight	single
Paul	60	FALSE
John	90	TRUE
Joe	80	TRUE

Data frame

Mazání řádku

```
> our.data.frame = our.data.frame[-1, ]  
> our.data.frame
```

name	weight	single
John	90	TRUE
Joe	80	TRUE

Data frame

Přidání sloupce

```
> our.data.frame$visits = c(19, 30)
>
> our.data.frame = cbind(our.data.frame, value = c(5, 18))
>
> our.data.frame
```

name	weight	single	visits	value
John	90	TRUE	19	5
Joe	80	TRUE	30	18

Data frame

Přidání řádku

```
> our.data.frame = rbind(our.data.frame, list('Ben', 30, FALSE, 0, 1))  
>  
> our.data.frame[nrow(our.data.frame) + 1, ] = list('Tom', 12, TRUE, 3, 1)  
>  
> our.data.frame
```

name	weight	single	visits	value
John	90	TRUE	19	5
Joe	80	TRUE	30	18
Ben	30	FALSE	0	1
Tom	12	TRUE	3	1

Data frame

Úkol

- Vytvořte data frame `df` o 100 řádcích s následujícími sloupci:
 - `id` obsahující čísla od 413 do 512
 - `weight` obsahující náhodný výběr z Normálního rozdělení $N(70, 9)$, viz. `?rnorm` (pozor na rozptyl vs směrodatná odchylka)
- Smažte prvních 5 řádků
- Přidejte řádek s hodnotami `id = 513` a `weight = 73`
- Vypište prvních pár řádků pomocí funkce `head`
- Spočítejte průměr sloupce `weight`
- Smažte sloupec `id`

Data frame

Řešení

```
> df = data.frame(id = 413:512, weight = rnorm(100, 70, 3))  
> df = df[-1:-5, ]  
> df[nrow(df) + 1, ] = c(513, 73)  
> head(df)
```

id	weight
18	69.8
19	69.5
20	65.6
21	68.6
22	71.3
23	74.1

```
> mean(df$weight)
```

```
[1] 70.3
```

```
> df$id = NULL
```

Data frame

Úkol

- Použijte data frame `mtcars`, viz. `?mtcars`
- Prohlídněte jej pomocí funkcí `str` a `head`
- Vytvořte data frame `mtcars.filtered`, který bude obsahovat jen automobily se čtyřmi válci (sloupeček `cyl`)
- Přidejte do `mtcars.filtered` sloupec `lp100km` s průměrnou spotřebou v L/100km dle vzorce

$$lp100km = \frac{282.48}{mpg}$$

Data frame

Řešení

```
> str(mtcars)
```

```
'data.frame': ^I32 obs. of  11 variables:  
 $ mpg  : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8  
          19.2 ...  
 $ cyl  : num   6 6 4 6 8 6 8 4 4 6 ...  
 $ disp: num  160 160 108 258 360 ...  
 $ hp   : num  110 110 93 110 175 105 245 62 95 123 ...  
 $ drat: num   3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69  
          3.92 3.92 ...  
 $ wt   : num   2.62 2.88 2.32 3.21 3.44 ...  
 $ qsec: num   16.5 17 18.6 19.4 17 ...  
 $ vs   : num   0 0 1 1 0 1 0 1 1 1 ...  
 $ am   : num   1 1 1 0 0 0 0 0 0 0 ...  
 $ gear: num   4 4 4 3 3 3 3 4 4 4 ...  
 $ carb: num   4 4 1 1 2 1 4 2 2 4 ...
```

Data frame

Řešení

```
> head(mtcars)
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1.0	6	160	110	3.90	2.62	16.5	0	1	4	4
1.0	6	160	110	3.90	2.88	17.0	0	1	4	4
2.8	4	108	93	3.85	2.32	18.6	1	1	4	1
1.4	6	258	110	3.08	3.21	19.4	1	0	3	1
8.7	8	360	175	3.15	3.44	17.0	0	0	3	2
8.1	6	225	105	2.76	3.46	20.2	1	0	3	1

```
> mtcars.filtered = mtcars[mtcars$cyl == 4, ]
```

```
> mtcars.filtered$lp100km = 282.48 / mtcars.filtered$mpg
```

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

Vektor

Matice

List

Data frame

Faktor

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

- R implementace kategoriálních proměnných
- v data frame automaticky, pokud nevypneme pomocí argumentu `stringsAsFactors`

Faktor

Konstrukce

```
> x = c('a', 'a', 'b', 'c', 'a', 'b')  
> x.factor = factor(x)  
> x.factor
```

```
[1] a a b c a b  
Levels: a b c
```

```
> str(x.factor)
```

```
Factor w/ 3 levels "a","b","c": 1 1 2 3 1 2
```

```
> summary(x.factor)
```

```
a b c  
3 2 1
```

Faktor

Data frame

```
> x = c('a', 'a', 'b', 'c', 'a', 'b')  
> y = seq_along(x)  
> y
```

```
[1] 1 2 3 4 5 6
```

```
> df = data.frame(x, y)  
> str(df)
```

```
'data.frame': ^I6 obs. of 2 variables:  
 $ x: chr  "a" "a" "b" "c" ...  
 $ y: int  1 2 3 4 5 6
```

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Working directory

- `getwd()`
- `setwd()`
- **na lomítkách v cestě nezáleží, pozor na escapování zpětného lomítka**

> "D:\\Dokumenty\\R3"

```
[1] "D:\\Dokumenty\\R3"
```

> "D:/Dokumenty/R3"

```
[1] "D:/Dokumenty/R3"
```

R - source

- Načtení R skriptu a jeho vykonání
- Projde cílový skript a vykoná jednotlivé řádky jako bychom je sami znova zadali
- V aktuálním kontextu - tj. zůstanou nám všechny proměnné atd.

```
> source('path/to/my/script.R')
```

Tabulková data

Funkce	header	sep	dec	write funkce
<code>read.table</code>	FALSE	" "	"."	<code>write.table</code>
<code>read.csv</code>	TRUE	", "	"."	<code>write.csv</code>
<code>read.csv2</code>	TRUE	"; "	", "	<code>write.csv2</code>
<code>read.delim</code>	TRUE	" [^] "	"."	<code>write.delim</code>
<code>read.delim2</code>	TRUE	" [^] "	", "	<code>write.delim2</code>

- `?read.table`
- `?write.table`
- **Parametr** `header`
- **Parametr** `skip`

Data [lengths]

```
> df = read.csv('../..//data/lengths.csv')  
> df
```

identifier	sex	age	length
7412301	M	18	11.7
7412302	F	33	16.7
7412303	F	41	13.4
7412304	F	40	12.7
7412305	M	23	16.2
7412306	M	58	17.2
7412307	F	31	11.2

Data [excel]

Balíček **openxlsx** [Walker, 2018]

```
> library('openxlsx')  
> excel.car.fuel = read.xlsx('../..//data/excel.xlsx')  
> str(excel.car.fuel)
```

```
'data.frame': ^I1142 obs. of 8 variables:  
 $ Model.Yr : num 2012 2012 2012 2012 2012 ...  
 $ Mfr.Name : chr "aston martin" "aston martin"  
 "aston martin" "aston martin" ...  
 $ Carline : chr "V12 Vantage" "V8 Vantage" "V8  
 Vantage" "V8 Vantage" ...  
 $ Engine : num 5.9 4.7 4.7 4.7 4.7 4.2 4.2 5.2 5.2  
 4.2 ...  
 $ Cylinders: num 12 8 8 8 8 8 8 10 10 8 ...  
 $ MPG : num 13.1 16 16 15.2 16 ...  
 $ Gears : num 6 6 7 6 7 6 6 6 6 6 ...  
 $ LPH : num 17.9 14.7 14.7 15.5 14.7 ...
```


Excel 2/2

```
> str(read.xlsx('../..//data/excel.xlsx', sheet = 'mtcars'))
```

```
'data.frame':^^I32 obs. of 12 variables:  
 $ X1 : chr "Mazda RX4" "Mazda RX4 Wag" "Datsun 710"  
 "Hornet 4 Drive" ...  
 $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8  
 19.2 ...  
 $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
 $ disp: num 160 160 108 258 360 ...  
 $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
 $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69  
 3.92 3.92 ...  
 $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
 $ qsec: num 16.5 17 18.6 19.4 17 ...  
 $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
 $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
 $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
 $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

Zápis dat

Úkol

- Uložte data frame `mtcars` do CSV souboru
 - Použijte argument `row.names = TRUE`

- Uložte sloupce `mpg` a `hp` z data frame `mtcars` do dalšího CSV souboru
 - Použijte argument `row.names = TRUE`

Zápis dat

Řešení

```
> write.csv(mtcars,  
+           '/tmp/r-slides/mtcars.csv',  
+           row.names = TRUE)  
> write.csv(mtcars[, c('mpg', 'hp')],  
+           '/tmp/r-slides/mtcars-mpg-hp.csv',  
+           row.names = TRUE)
```

Načtení dat

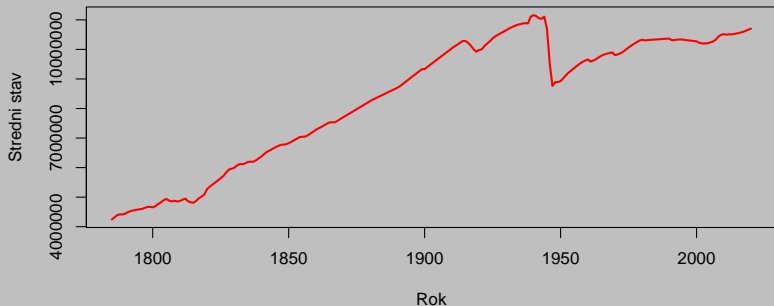
Úkol

- **Stáhněte si tabulku** Graf 1 Počet obyvatel v letech 1785–2020
[✂ http://czso.petrmarek.eu](http://czso.petrmarek.eu)
- Načtěte z CSV záznamy o počtu obyvatel a letech.
- Vykreslete pomocí funkce `plot` vývoj obyvatelstva do grafu
 - použijte křivku (argument `type = 'l'`)
 - červenou barvu (argument `col = 'red'`)
 - tlustou čáru (argument `lwd = 2`)
- Pro zájemce
 - Přidejte do načtených dat sloupec uvádějící rozdíl v počtu obyvatel oproti předchozímu roku
 - Uložte upravená data do nového CSV souboru

Načtení dat

Řešení

```
> data = read.csv('../..//data/13007021g01.csv')  
> plot(data, type = 'l', col = 'red', lwd = 2,  
+       xlab = 'Rok',  
+       ylab = 'Stredni stav')
```



Načtení dat

Řešení

```
> data$differences = NA
> for (i in 2:nrow(data)) {
+   data$differences[i] = data[i, 2] - data[i - 1, 2]
+ }
> write.csv(data, '/tmp/r-slides/13007021g01-update.csv')
```

Balíček **foreign** [R Core Team, 2014b]

```
> library('foreign')  
> read.spss('file-name.spss')
```

viz. `?read.spss`

Další zdroje dat

Stata

Balíček **foreign** [R Core Team, 2014b]

```
> library('foreign')  
> read.dta('file-name.dta')
```

viz. `?read.dta`

Balíček **odbc** [Hester and Wickham, 2021]

```
> library(odbc)
>
> con = DBI::dbConnect(odbc::odbc(),
+                       Driver   = "PostgreSQL Unicode",
+                       Server   = "localhost",
+                       Database = "r_test_database",
+                       UID      = "petr",
+                       PWD      = "petr",
+                       Port     = 5432)
>
> result = DBI::dbGetQuery(con,
+                            "SELECT id, file_name FROM folio_files LIMIT 3")
```

Další zdroje dat

SQL databáze 2/2

```
> head(result)
```

```
id                file_name
60                verejna-vyzva-upis-creditas-1911.pdf
87                cni_odkoupene-vydane-akcie_lock.xlsx
56 dodatek-1-statut-creditas-fond-sicav-1911.pdf
```

```
> str(result)
```

```
'data.frame': ^I3 obs. of  2 variables:
 $ id          :integer64 60 87 56
 $ file_name: chr
   "verejna-vyzva-upis-creditas-1911.pdf"
   "cni_odkoupene-vydane-akcie_lock.xlsx"
   "dodatek-1-statut-creditas-fond-sicav-1911.pdf"
```

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

Jednorozměrná data

Dvourozměrná data

Vícerozměrná data

Obecná nastavení grafů

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

R balíčky

- `install.packages()`
- `library()`
- `citation()`

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

Jednorozměrná data

Dvourozměrná data

Vícerozměrná data

Obecná nastavení grafů

6. Programování

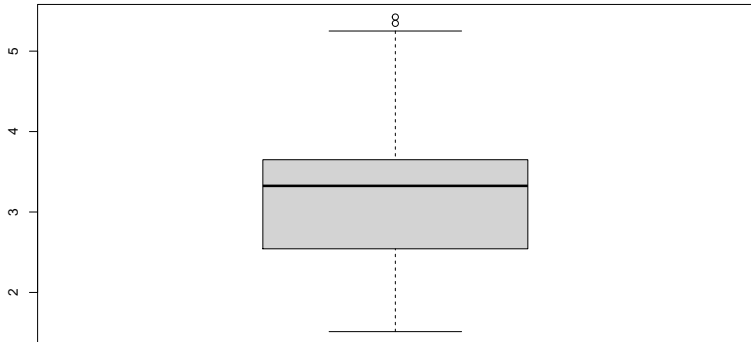
7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

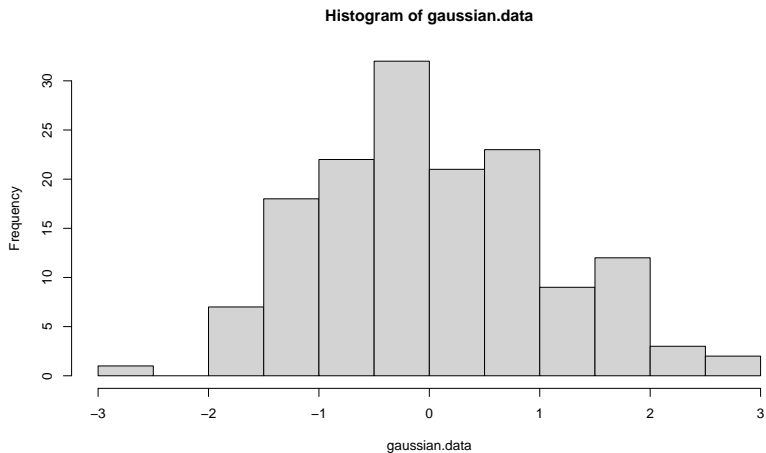
Krabičkový graf

```
> boxplot(mtcars$wt)
```



Histogram

```
> gaussian.data = rnorm(150)
> hist(gaussian.data)
```



Histogram

breaks 1/3

- Argument `breaks`
- Výchozím nastavením Sturgesovo pravidlo

$$k = 1 + \log_2 n$$

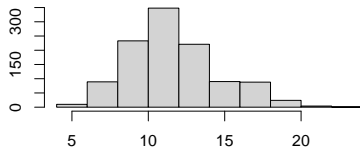
nevhodné pro velké výběry - má tendenci příliš vyhlazovat
použitím nedostatečného počtu intervalů [Scott, 1979].

Histogram

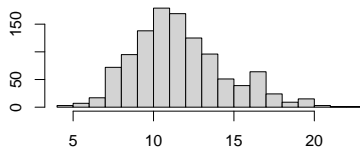
breaks 2/3

Data [car-fuel] n = 1142

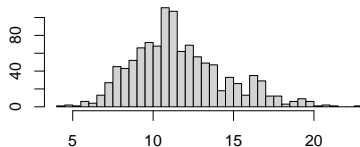
sturges



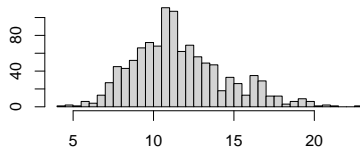
scott



fd



\sqrt{n}



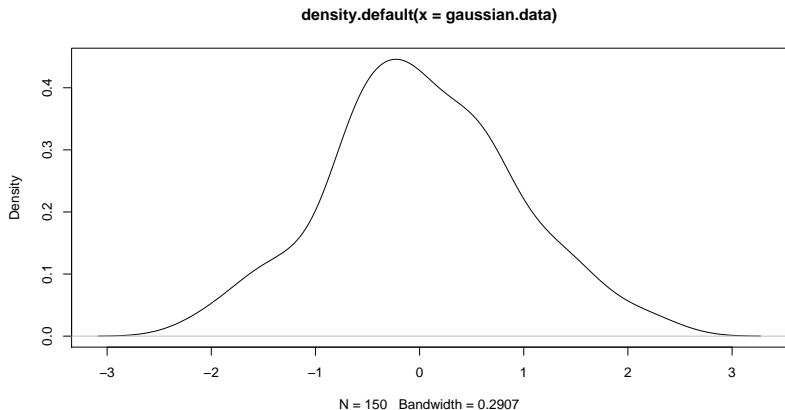
Histogram

breaks 3/3

```
> data = read.csv('../..//data/car-fuel.csv')
> par(mfrow = c(2,2), cex = 1.2, mar = c(3,3,3,1))
> rules = c('sturges', 'scott', 'fd')
> for (rule in rules) {
+   hist(data$LPH, breaks = rule, main = rule)
+ }
> hist(data$LPH,
+       breaks = sqrt(length(data$LPH)),
+       main = bquote(sqrt('n')))
```

Jádrový odhad hustoty

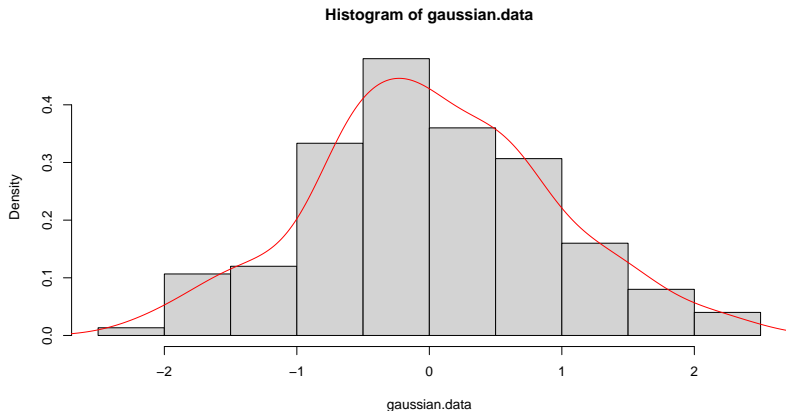
```
> set.seed(1)
> gaussian.data = rnorm(150)
> density.est = density(gaussian.data)
> plot(density.est)
```



Jádrový odhad hustoty

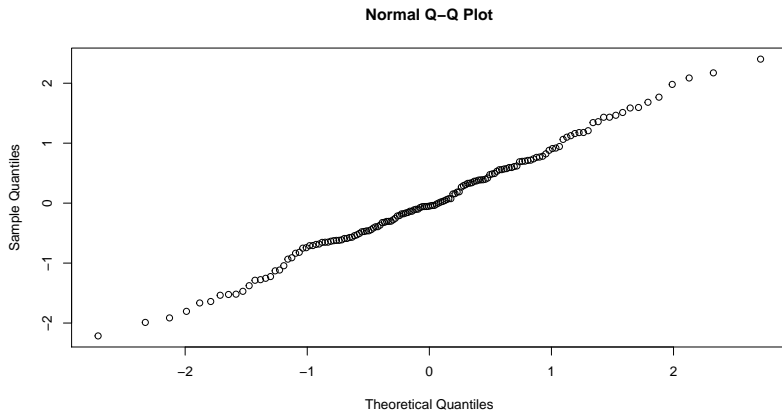
nad histogramem

```
> set.seed(1)
> gaussian.data = rnorm(150)
> hist(gaussian.data, freq = FALSE)
> lines(density(gaussian.data), col = 'red')
```



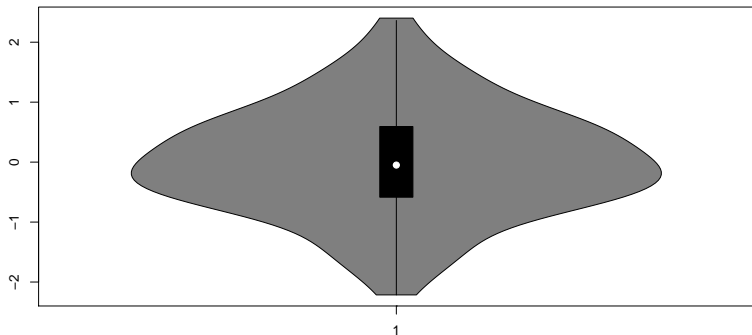
QQ graf

```
> set.seed(1)
> gaussian.data = rnorm(150)
> qqnorm(gaussian.data)
```



Violin plot

- > suppressPackageStartupMessages(library('vioplot'))
- > set.seed(1)
- > vioplot(rnorm(150))



Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

Jednorozměrná data

Dvourozměrná data

Vícerozměrná data

Obecná nastavení grafů

6. Programování

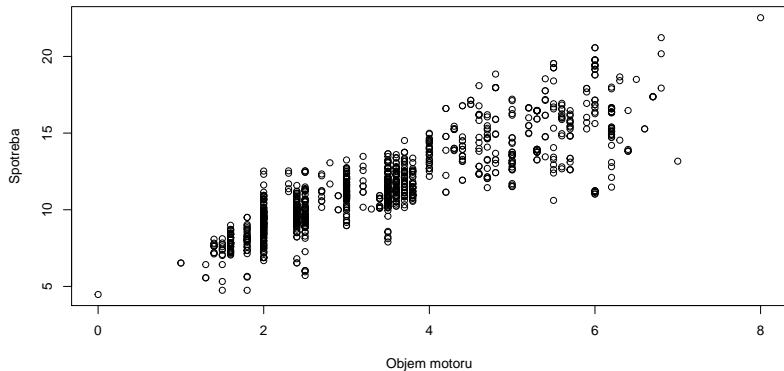
7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Data [car-fuel]

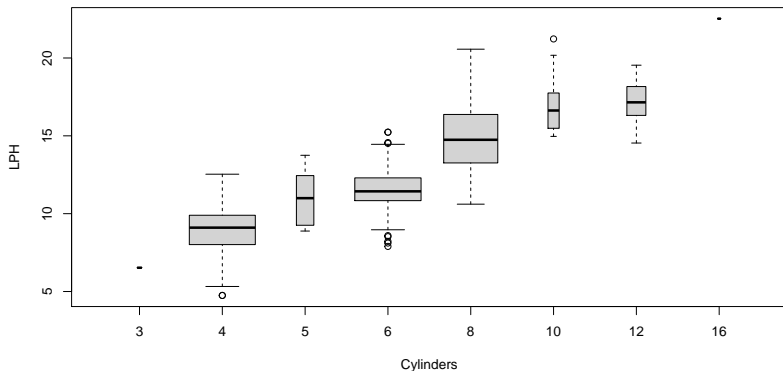
- > data = read.csv('.././data/car-fuel.csv')
- > plot(LPH ~ Engine, data = data, xlab = 'Objem motoru', ylab = 'Spotreba')



Vícenásobný krabičkový graf

Data [car-fuel]

```
> data = read.csv('../..//data/car-fuel.csv')  
> boxplot(LPH ~ Cylinders, data = data, varwidth = TRUE)
```



Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

Jednorozměrná data

Dvourozměrná data

Vícerozměrná data

Obecná nastavení grafů

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

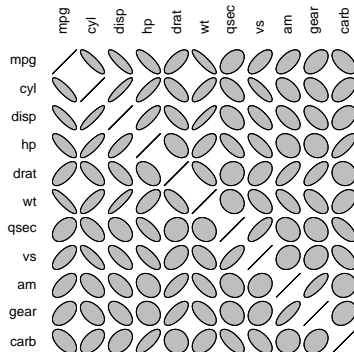
9. Použitá data

Korelační matice

balíček ellipse

Balíček **ellipse** [Murdoch and Chow, 2012]

```
> suppressPackageStartupMessages(library('ellipse'))  
> plotcorr(cor(mtcars))
```

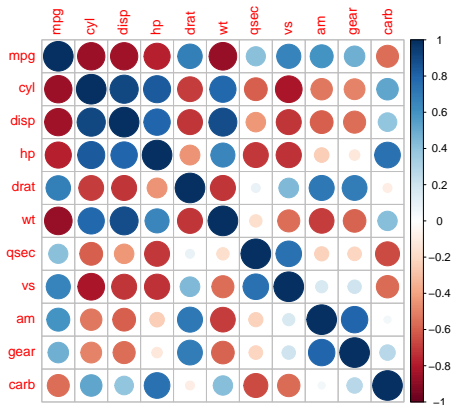


Korelační matice

balíček `corrplot`

Balíček `corrplot` [Wei and Simko, 2017]

```
> suppressPackageStartupMessages(library('corrplot'))  
> corrplot(cor(mtcars))
```

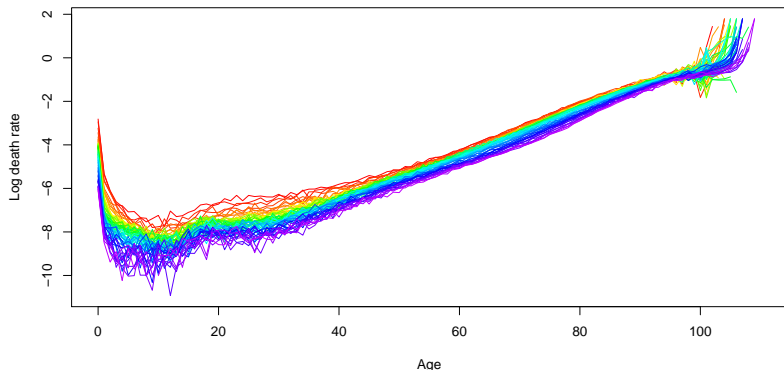


Rainbow plot

Data [mort]

- > suppressPackageStartupMessages(library('demography'))
- > load('../..../data/mort.RData')
- > plot(mort)

Czech Republic: female death rates (1950–2011)



Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

Jednorozměrná data

Dvourozměrná data

Vícerozměrná data

Obecná nastavení grafů

6. Programování

7. Regrese

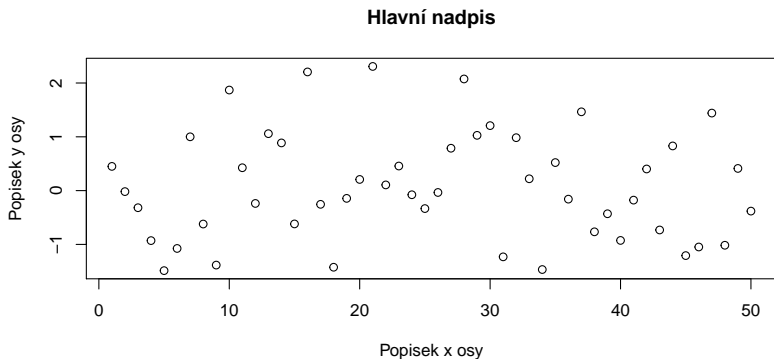
8. Výpočet úmrtnostních tabulek

9. Použitá data

main	hlavní nadpis
xlab	popisek x osy
ylab	popisek y osy

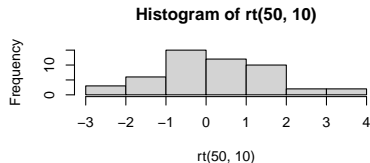
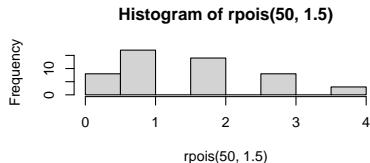
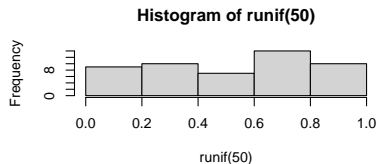
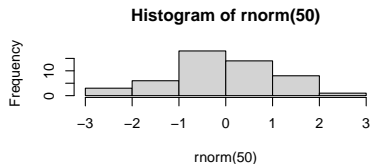
Popisky

```
> plot(rnorm(50),  
+      main = 'Hlavní nadpis',  
+      xlab = 'Popisek x osy',  
+      ylab = 'Popisek y osy')
```



Více grafů v jednom obrázku

```
> par(mfrow = c(2,2))  
> hist(rnorm(50))  
> hist(runif(50))  
> hist(rpois(50, 1.5))  
> hist(rt(50, 10))
```



Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

6. Programování

If/else

For cyklus

While cyklus

Funkce

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Obsah prezentace

1. Obecné
2. Základy
3. Datové struktury
4. Externí data
5. Grafická znázornění
- 6. Programování**
 - If/else**
 - For cyklus
 - While cyklus
 - Funkce
7. Regrese
8. Výpočet úmrtnostních tabulek
9. Použitá data

If/else

```
> if (CONDITION) {  
+   # run this code if CONDITION is TRUE (or truthy)  
+ } else {  
+   # run this code if CONDITION is FALSE (or falsey)  
+ }  
> if (1 < 3) {  
+   print('Yes')  
+ } else {  
+   print('No')  
+ }
```

```
[1] "Yes"
```

If bez else

```
> x = 3
>
> if (x < 3) {
+   print('x is lower than 3')
+ }
>
> if (x < 5) {
+   print('x is lower than 5')
+ }
```

```
[1] "x is lower than 5"
```

Vnořené if

```
> x = 10
>
> if (x > 5) {
+   print('x is high')

+   if (x > 7) {
+     print('x is super-high!')
+   }
+ }
```

```
[1] "x is high"
[1] "x is super-high!"
```

ifelse

```
> x = c('a', 'b', 'a', 'a', 'c')  
> ifelse(x == 'a', 'a', 'not a')
```

```
[1] "a"      "not a" "a"      "a"      "not a"
```

```
> y = 1:6  
> data.frame(num = y, compared.to.3 = ifelse(y > 3, '>', '<='))
```

```
num compared.to.3  
1                <=  
2                <=  
3                <=  
4                >  
5                >  
6                >
```

ifelse

Úkol

- Použijte data frame `mtcars`, viz. `?mtcars`.
- Doplněte sloupec `consumption` s hodnotami "high" pro `mpg < 16` a "low" pro ostatní.
- Doplněte sloupec `strong` s hodnotami `TRUE` a `FALSE` podle sloupce `hp` (uvažujme auto jako `strong` pokud má více než 130 koní)

ifelse

Řešení

```
> # Optionally copy mtcars to a new data frame
> # so that we don't override the default
> mtcars.new = mtcars
> mtcars.new$consumption = ifelse(mtcars.new$mpg < 16, 'high', 'low')
> mtcars.new$strong = mtcars$hp > 130
```

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

6. Programování

If/else

For cyklus

While cyklus

Funkce

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

For cyklus

```
> for (i in 1:3) {  
+   print(i)  
+ }
```

```
[1] 1  
[1] 2  
[1] 3
```

```
> x = c('a', 'b', 'c')  
>  
> for (i in x) {  
+   print(i)  
+ }
```

```
[1] "a"  
[1] "b"  
[1] "c"
```

For cyklus

Proměnné

```
> x = 10:15
> y = c()
>
> for (i in seq_along(x)) {
+   y[i] = x[i]
+ }
>
> y
```

```
[1] 10 11 12 13 14 15
```

For cyklus - sudé/liché

Úkol

- Pro každé z čísel od 1 do 20 vypište, zda je sudé, nebo liché

For cyklus

Příklad s if/else

```
> balance = 5
> transfers = c(100, -5, -20)
> for (transfer in transfers) {
+   if (transfer > 0) {
+     print(paste('received', transfer))
+   } else {
+     print(paste('paid', -transfer))
+   }
+   balance = balance + transfer
+   print(paste('current account balance:', balance))
+ }
```

```
[1] "received 100"
[1] "current account balance: 105"
[1] "paid 5"
[1] "current account balance: 100"
[1] "paid 20"
[1] "current account balance: 80"
```

For cyklus - vektor

Úkol

- Vytvořte vektor `numbers` obsahující čísla od 1 do 100
- Vytvořte proměnnou `numbers.sum` a pomocí `for` cyklu do ní uložte součet všech čísel z vektoru `numbers`

For cyklus - vektor

Řešení

```
> numbers = 1:100  
> numbers.sum = 0  
> for (i in numbers) {  
+   numbers.sum = numbers.sum + i  
+ }  
> numbers.sum
```

```
[1] 5050
```

For cyklus

Data frame

```
> df = data.frame(letter = c('A', 'B', 'C'),  
+                 is.vowel = c(TRUE, FALSE, FALSE))  
>  
> for (i in 1:nrow(df)) {  
+   print(df[i, ])  
+ }
```

```
letter is.vowel  
  A      TRUE  
letter is.vowel  
  B     FALSE  
letter is.vowel  
  C     FALSE
```

For cyklus - mtcars

Úkol

- Použijte data frame `mtcars`, viz. `?mtcars`
- Pro každé auto vypište, jestli je silnější nebo slabší než to předchozí (dle sloupce `hp`)

For cyklus - mtcars

Řešení

```
> for (i in 2:nrow(mtcars)) {  
+   current.value = mtcars$hp[i]  
+   previous.value = mtcars$hp[i - 1]  
+   if (current.value > previous.value) {  
+     print('silnejsi')  
+   } else {  
+     print('slabsi')  
+   }  
+ }
```

```
[1] "slabsi"  
[1] "slabsi"  
[1] "silnejsi"  
[1] "silnejsi"  
[1] "slabsi"  
[1] "silnejsi"  
[1] "slabsi"  
[1] "silnejsi"  
[1] "silnejsi"  
[1] "slabsi"  
[1] "silnejsi"
```

Obsah prezentace

1. Obecné
2. Základy
3. Datové struktury
4. Externí data
5. Grafická znázornění
- 6. Programování**
 - If/else
 - For cyklus
 - While cyklus**
 - Funkce
7. Regrese
8. Výpočet úmrtnostních tabulek
9. Použitá data

While cyklus

```
> x = 1  
> while (x < 5) {  
+   print(x)  
+   x = x + 1  
+ }
```

```
[1] 1  
[1] 2  
[1] 3  
[1] 4
```

```
> x
```

```
[1] 5
```

While cyklus

Úkol

- Pomocí `while` určete, kolik je potřeba přirozených čísel (vzestupně od 1, tj. 1, 2, 3, ...), aby jejich součet byl větší než 100.
- **Nápověda**
 - vnější proměnné `count` a `total`
 - `1:count`, funkce `sum`

While cyklus

Řešení

```
> count = 0
> total = 0
> while (total < 100) {
+   count = count + 1
+   total = sum(1:count)
+ }
> count
```

```
[1] 14
```


Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

6. Programování

If/else

For cyklus

While cyklus

Funkce

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Funkce

Motivace

- **flexibilita, znovupoužitelnost, organizace delších skriptů**
- **DRY kód**

```
> FunctionName = function() {  
+   return(1 + 2)  
+ }  
> FunctionName()
```

```
[1] 3
```

Funkce

Argumenty 1/2

```
> MySum = function(x, y) {  
+   return(x + y)  
+ }  
>  
> MySum(2, 3)
```

```
[1] 5
```

Funkce

Argumenty 2/2

```
> MySum = function(x = 3, y = 5) {  
+   return(x + y)  
+ }  
>  
> MySum()
```

```
[1] 8
```

```
> MySum(1)
```

```
[1] 6
```

```
> MySum(1, 2)
```

```
[1] 3
```

```
> MySum(y = 0)
```

```
[1] 3
```

Funkce

Variable scope 1/2

```
> MySum = function(x) {  
+   inner.variable = 3  
+   return(x + inner.variable)  
+ }  
> MySum(5)
```

```
[1] 8
```

```
> inner.variable
```

```
Error in eval(expr, envir, enclos): object 'inner.variable' not found
```

Funkce

Variable scope 2/2

```
> z = 1
> MySum = function(x) {
+   print(paste('z:', z))
+   z = z + 2
+   print(paste('z:', z))
+   return(x + z)
+ }
> MySum(5)
```

```
[1] "z: 1"
[1] "z: 3"
[1] 8
```

```
> z
```

```
[1] 1
```

Funkce

Příklad 1/2

```
> NumberIsEven = function(x) {  
+   if (x %% 2 == 0) {  
+     return(TRUE)  
+   } else {  
+     return(FALSE)  
+   }  
+ }  
  
> NumberIsOdd = function(x) {  
+   return(!NumberIsEven(x))  
+ }
```

Funkce

Příklad 2/2

> NumberIsEven(11)

```
[1] FALSE
```

> NumberIsOdd(11)

```
[1] TRUE
```

> NumberIsEven(28)

```
[1] TRUE
```

> NumberIsOdd(28)

```
[1] FALSE
```


Funkce - součet sloupce

Úkol

- Vytvořte funkci `SumColumn` přijímající dva argumenty
 - data frame
 - jméno či index sloupce

- Funkce by měla vrátit součet hodnot v daném sloupci

```
> SumColumn(mtcars, 'cyl')
```

```
[1] 198
```

Funkce - součet sloupce

Řešení

```
> SumColumn = function(df, name.or.index) {  
+   column = df[, name.or.index]  
+   return(sum(column))  
+ }  
> SumColumn(mtcars, 'cyl')
```

```
[1] 198
```

Funkce - čtvercová matice

Úkol

- Vytvořte funkci `SquareMatrix` přijímající číslo n jako argument.
- Funkce by měla vrátit čtvercovou matici o rozměrech $n \times n$ vyplněnou přirozenými čísly postupně od 1

```
> SquareMatrix(2)
```

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

Funkce - čtvercová matice

Řešení

```
> SquareMatrix = function(n) {  
+   numbers = 1:(n^2)  
+   square.matrix = matrix(numbers, nrow = n, ncol = n)  
+   return(square.matrix)  
+ }  
> SquareMatrix(2)
```

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

Funkce - variační koeficient

Úkol

- Vytvořte funkci `CoefficientOfVariation` přijímající vektor jako argument, ze kterého spočítá variační koeficient.

$$c_v = \frac{\sigma}{\mu}$$

- Pro odhad μ použijte průměr `mean`
- Pro odhad σ použijte směrodatnou odchylku `sd`, viz. `?sd`

Funkce - variační koeficient

Řešení

```
> CoefficientOfVariation = function(data) {  
+   data.mean = mean(data)  
+   data.sd = sd(data)  
+   return(data.sd / data.mean)  
+ }  
> CoefficientOfVariation(c(1, 2, 3))
```

```
[1] 0.5
```

Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Data [mort] a balíček **demography** [Hyndman et al., 2012] pro snazší práci s daty

```
> load('../..'/data/mort.RData')  
> suppressPackageStartupMessages(library('demography'))  
> mort
```

```
Mortality data for Czech Republic  
Series: female male total  
Years: 1950 - 2011  
Ages: 0 - 110
```

Regresní analýza úmrtnosti mužů v roce 2011.

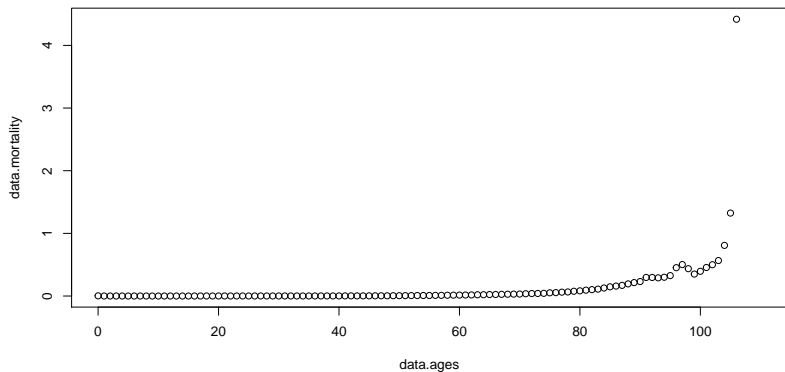
```
> last.column = ncol(mort$rate$male)  
> data.mortality = mort$rate$male[, last.column]  
> data.ages = mort$age
```


Data

Graf

Prozkoumejme data graficky

```
> plot(data.mortality ~ data.ages)
```

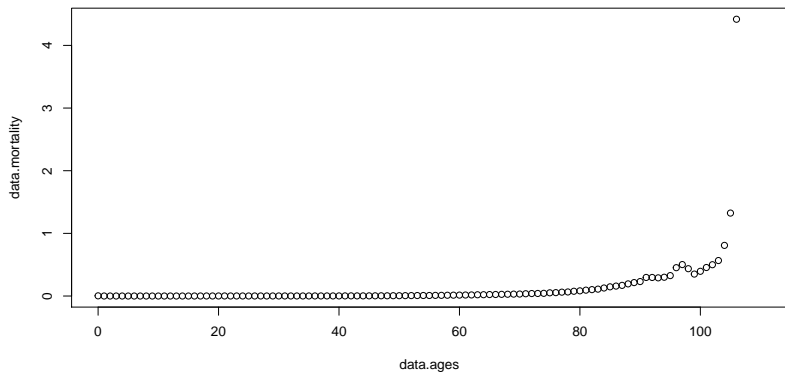


Data

Graf

Prozkoumejme data graficky

```
> plot(data.mortality ~ data.ages)
```



Naivní regresní přímka I

```
> fit.line = lm(data.mortality ~ data.ages)
> summary(fit.line)
```

```
Call:
lm(formula = data.mortality ~ data.ages)

Residuals:
    Min       1Q   Median       3Q      Max
-0.225 -0.164 -0.068  0.066  3.951

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.20047    0.08049   -2.49   0.014 *
data.ages     0.00629    0.00131    4.79 0.0000054 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
                0.1 ' ' 1
```

Naivní regresní přímka II

```
Residual standard error: 0.419 on 105 degrees of
  freedom
(4 observations deleted due to missingness)
Multiple R-squared:  0.18, Adjusted R-squared:  0.172
F-statistic: 23 on 1 and 105 DF, p-value: 0.00000543
```

Grafická diagnostika přes

```
> plot(fit.line)
```

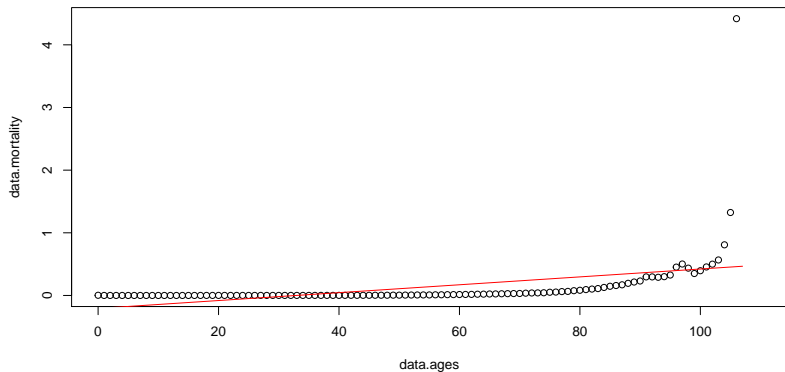
Funkce pracující s výstupem lm()

<code>coefficients</code>	vektor odhadů parametrů
<code>confint</code>	intervaly spolehlivosti pro odhady parametrů
<code>fitted</code>	hodnoty modelu
<code>residuals</code>	rezidua
<code>anova</code>	anova tabulka
<code>vcov</code>	kovarianční matice parametrů
<code>influence</code>	hodnoty pro další hodnocení kvality modelu

Naivní regresní přímka

Zobrazení odhadu

- > plot(data.mortality ~ data.ages)
- > lines(fitted(fit.line), col = 'red')



Parabola I

```
> fit.curve = lm(data.mortality ~ data.ages + I(data.ages^2))  
> summary(fit.curve)
```

```
Call:  
lm(formula = data.mortality ~ data.ages +  
    I(data.ages^2))  
  
Residuals:  
    Min      1Q  Median      3Q     Max  
-0.326 -0.155 -0.008  0.091  3.546  
  
Coefficients:  
                Estimate Std. Error t value Pr(>|t|)  
(Intercept)    0.2050832  0.1071423   1.91   0.05835  
.  
data.ages      -0.0168855  0.0046714  -3.61   0.00047  
***
```

Parabola II

```
I(data.ages^2)  0.0002186  0.0000426    5.13 0.0000014
***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
                0.1 ' ' 1

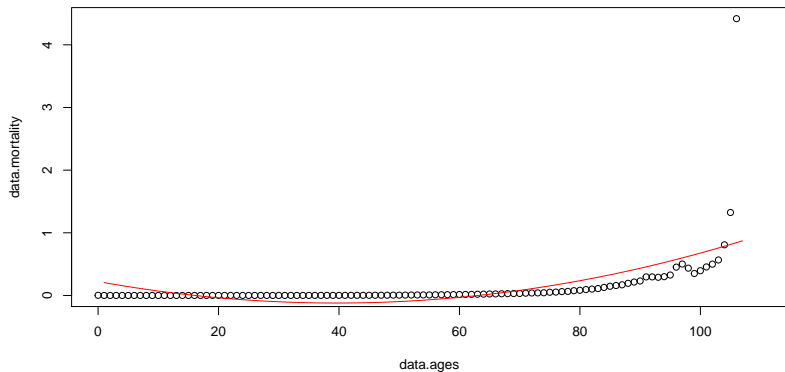
Residual standard error: 0.376 on 104 degrees of
  freedom
(4 observations deleted due to missingness)
Multiple R-squared:  0.345, Adjusted R-squared:
  0.332
F-statistic: 27.4 on 2 and 104 DF,  p-value:
  0.000000000277
```

```
> plot(fit.curve)
```


Parabola

Zobrazení odhadu

- > plot(data.mortality ~ data.ages)
- > lines(fitted(fit.curve), col = 'red')



Parabola bez úroňové konstanty I

- > fit.curve.no.intercept = lm(data.mortality ~ data.ages + I(data.ages^2) - 1)
- > summary(fit.curve.no.intercept)

Call:

```
lm(formula = data.mortality ~ data.ages +  
    I(data.ages^2) - 1)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.293	-0.148	0.034	0.110	3.612

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
data.ages	-0.0091831	0.0024025	-3.82	0.00022 ***
I(data.ages^2)	0.0001584	0.0000291	5.44	0.00000035 ***

Parabola bez úroňové konstanty II

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  
                0.1 ' ' 1
```

```
Residual standard error: 0.381 on 105 degrees of  
  freedom
```

```
(4 observations deleted due to missingness)
```

```
Multiple R-squared:  0.374, ^IAdjusted R-squared:  
  0.363
```

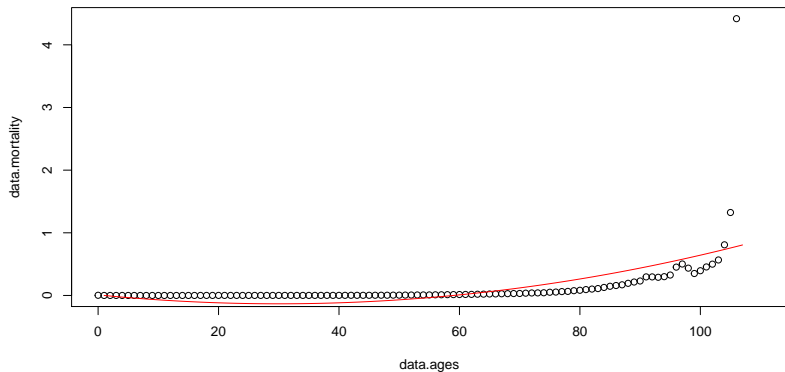
```
F-statistic: 31.4 on 2 and 105 DF,  p-value:  
  0.0000000000201
```

```
> plot(fit.curve)
```

Parabola bez úroňové konstanty

Zobrazení odhadu

- > plot(data.mortality ~ data.ages)
- > lines(fitted(fit.curve.no.intercept), col = 'red')



Obsah prezentace

1. Obecné

2. Základy

3. Datové struktury

4. Externí data

5. Grafická znázornění

6. Programování

7. Regrese

8. Výpočet úmrtnostních tabulek

9. Použitá data

Výpočet úmrtnostních tabulek

Úkol

- Provedte zjednodušený výpočet úmrtnostních tabulek pro muže / ČR / 2020
- [↗](#) Vstupní data (věk, Dx a Px) z ČSÚ tabulek - csv z czso.petrmarek.eu
- [↗](#) Metodika
 - Zjednodušení na “naivní” výpočet - použijeme jen reálné míry úmrtnosti
- Vstupem cesta k CSV s ČSÚ věk/Dx/Px, výstupem data.frame a vygenerované CSV s vlastním (naivním) výpočtem.
- Vytvořte R skript, poté zkuste zobecnit na funkci přijímající argumenty, např.

```
NaiveLifeTable = function(csv.path.input, csv.path.output = NULL) {}
```

Výpočet úmrtnostních tabulek

Řešení

 life-tables.R

Skript je vystaven také na <http://czso.petrmarek.eu/>

Obsah prezentace

1. Obecné
2. Základy
3. Datové struktury
4. Externí data
5. Grafická znázornění
6. Programování
7. Regrese
8. Výpočet úmrtnostních tabulek
- 9. Použitá data**

Použitá data

lengths.csv

```
> str(read.csv('../..data/lengths.csv'))
```

```
'data.frame':^^I7 obs. of 4 variables:  
 $ identifier: int 7412301 7412302 7412303 7412304  
 7412305 7412306 7412307  
 $ sex : chr "M" "F" "F" "F" ...  
 $ age : int 18 33 41 40 23 58 31  
 $ length : num 11.7 16.7 13.4 12.7 16.2 17.2 11.2
```

Použitá data



```
> str(read.csv('.././data/car-fuel.csv'))
```

```
'data.frame':^^I1142 obs. of 8 variables:  
 $ Model.Yr : int 2012 2012 2012 2012 2012 2012 2012 2012  
 2012 2012 2012 ...  
 $ Mfr.Name : chr "aston martin" "aston martin"  
 "aston martin" "aston martin" ...  
 $ Carline : chr "V12 Vantage" "V8 Vantage" "V8  
 Vantage" "V8 Vantage" ...  
 $ Engine : num 5.9 4.7 4.7 4.7 4.7 4.2 4.2 5.2 5.2  
 4.2 ...  
 $ Cylinders: int 12 8 8 8 8 8 8 10 10 8 ...  
 $ MPG : num 13.1 16 16 15.2 16 ...  
 $ Gears : int 6 6 7 6 7 6 6 6 6 6 ...  
 $ LPH : num 17.9 14.7 14.7 15.5 14.7 ...
```

Použitá data

 excel.xlsx

Kombinace dat Data [car-fuel] a data framu `mtcars` jako jednotlivých listů v Excel sešitu

Roční panelová data udávající míry úmrtnosti v Českých zemích mezi lety 1950 a 2011. Data pocházejí z Human Mortality Database a jsou (po registraci) k dispozici ke stažení. Pro jednoduchou práci s daty využívám balíčku **demography** [Hyndman et al., 2012], který velmi snadno zachází s daty strukturovaných dle Human Mortality Database.

```
> load('../..'/data/mort.RData')  
> suppressPackageStartupMessages(library('demography'))  
> class(mort)
```

```
[1] "demogdata"
```

```
> mort
```

```
Mortality data for Czech Republic  
Series: female male total  
Years: 1950 - 2011  
Ages: 0 - 110
```

Reference I

- J. Hester and H. Wickham. *odbc: Connect to ODBC Compatible Databases (using the DBI Interface)*, 2021. URL <https://CRAN.R-project.org/package=odbc>. R package version 1.3.2.
- R. J. Hyndman, contributions from Heather Booth, L. Tickle, J. Maindonald, S. Wood, and R. C. Team. *demography: Forecasting mortality, fertility, migration and population data*, 2012. URL <http://CRAN.R-project.org/package=demography>. R package version 1.14.
- D. Murdoch and E. D. Chow. *ellipse: Functions for drawing ellipses and ellipse-like confidence regions*, 2012. URL <http://CRAN.R-project.org/package=ellipse>. R package version 0.3-7.

Reference II

- R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014a. URL <http://www.R-project.org/>.
- R Core Team. *foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...*, 2014b. URL <http://CRAN.R-project.org/package=foreign>. R package version 0.8-61.
- D. W. Scott. On optimal and data-based histograms. *Biometrika*, 66 (3):pp. 605–610, 1979. ISSN 00063444. URL <http://www.jstor.org/stable/2335182>.
- A. Walker. *openxlsx: Read, Write and Edit XLSX Files*, 2018. URL <https://CRAN.R-project.org/package=openxlsx>. R package version 4.1.0.

Reference III

- T. Wei and V. Simko. *R package "corrplot": Visualization of a Correlation Matrix*, 2017. URL <https://github.com/taiyun/corrplot>. (Version 0.84).
- Y. Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2014. URL <http://yihui.name/knitr/>. R package version 1.8.

Feedback

https:

[//docs.google.com/forms/d/e/1FAIpQLSf5ERhzT6MXyCG-UBIW_TYM1AaGCCHMcG-HoUaC7EuhkuU5kg/viewform](https://docs.google.com/forms/d/e/1FAIpQLSf5ERhzT6MXyCG-UBIW_TYM1AaGCCHMcG-HoUaC7EuhkuU5kg/viewform)