# Statistika a programování v R

**Aleš Kozubík**
**Žilinská univerzita v Žilině**

**Project: Innovative Open Source Courses for Computer Science**

**31. 5. 2021**

Co-funded by the Erasmus+ Programme of the European Union

# Innovative Open Source Courses for Computer Science


This teaching material was written as one of the outputs of the project
"Innovative Open Source Courses for Computer Science",
funded by the Erasmus+ grant no. 2019-1-PL01-KA203-065564.
The project is coordinated by West Pomeranian University of Technology in Szczecin (Poland)
and is implemented in partnership with Mendel University in Brno (Czech Republic)
and University of Žilina (Slovak Republic).
The project implementation timeline is September 2019 to December 2022.

# Innovative Open Source Courses for Computer Science

Project was implemented under the Erasmus+.
Project name: "Innovative Open Source courses for Computer Science curriculum"
Project no.: 2019-1-PL01-KA203-065564
Key Action: KA2 – Cooperation for innovation and the exchange of good practices
Action Type: KA203 – Strategic Partnerships for higher education

Consortium: Zachodniopomorski uniwersytet technologiczny w Szczecinie
Mendelova univerzita v Brně
Žilinská univerzita v Žiline

Erasmus+ Disclaimer: This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

# Statistika a programování v R

## I. Úvod do prostředí R

# Charakteristika jazyka R

- je to jazyk a svobodné softwarové prostředí specializované na statistické výpočty a vizualizaci dat,
- je dostupný pro všechny běžné operační systémy: UNIX-ové platformy, Win nebo MacOS,
- je alternativou ku komerčnímu nástroji S resp. S-plus (jazyk a prostředí), jenž vyvinula společnost AT&T.

# Proč R

- je bezplatný, většina platforem statistického softwaru stojí tisíce dolarů,
- k programu je dostupné velké množství rozšiřujících balíčků,
- R dokáže snadno importovat údaje z různých zdrojů,
- R má implementovaných mnoho pokročilých statistických nástrojů,
- R poskytuje interaktivní platformu na analýzu údajů,
- prostředí R nabízí vizualizaci dat v podobě vysoce kvalitních a estetických grafů,
- je nezávislé na platformě, kompatibilní s většinou nejrozšířenějších operačních systémů,
- je kompatibilní s programovacími jazyky jako C,C++, Python, Java.

# Instalace R

Je volně dostupný ze zdroje Comprehensive R Archive Network (skratka CRAN).

Na internetu je umístěn na adrese https://cran.r-project.org.

K dispozici jsou předkompilované binární soubory pro běžné platformy Linux, Mac OS a Windows.

Ke stažení instalačního balíčku si můžeme zvolit vhodné zrcadlo.

# Instalace balíčků R

Ku jádru R existuje bohatý soubor balíčků rozšiřujících jeho funkce.

Balíčky zvyšují výkon R.

Na instalaci balíčků používáme funkci `install.packages()`

# První spuštění R

jestliže jsme nainstalovali R, můžeme ověřit jeho funkčnost.

Prostředí R spustíme jednoduše z příkazového řádku zadáním:

```
username@host:~$ R
```

Zobrazí se krátká úvodní poznámka, následovaná znakem

```
>
```

Tento symbol je znakem příkazového řádku prostředí R.

# Opuštění prostředí R

Prostředí R je nyní připraveno k práci.

Pro ukončení práce v prostředí R jednoduše zadáme

```
> q()
```

R reaguje otázkou:

```
Save workspace image? [y/n/c]:
```

Zvolíme-li `y`, záznam celé historie provedených příkazů se uloží do souboru `.Rhistory`, jenž se zapisuje do pracovního adresáře.

# Pracovní plocha a navigace

Všechny příkazy zadáváme interaktivně na příkazovém řádku.

V historii příkazů se pohybujeme pomocí kurzorových kláves, šipek směrem nahoru a dolů.

Tak se lze vrátit ke starším příkazům bez nutnosti jejich opětovného zápisu. Jen si vybereme požadovaný příkaz a opětovně ho odešleme klávesou Enter.

Uložíme-li si při opuštění prostředí historii, lze se vrátit i k příkazům z předchozí relace.

# Komunikace s OS

Předvolený pracovní adresář je adresář, v němž byl program R spuštěn. V tomto aktuálním pracovním adresáři R čte a ukládá soubory a výsledky. Aktuální pracovní adresář ověříme pomocí funkce `getwd()`.

Aktuální pracovní adresář lze měnit použitím funkce `setwd()`.

Ke spuštění příkazů operačního systému užijeme funkce `system()`.

Nový adresář vytvoříme příkazem

```
> system("mkdir new")
```

# Získání nápovědy

Funkce pro získání nápovědy má obecně jednoduchý tvar `help()` nebo zkráceně pomocí operátoru `?`.

Chceme-li získat informace o rozšiřujících balíčcích, použijeme

```
> help(package="jméno balíčku")
```

Některé balíčky obsahují rovněž ukázky kódu, které spustíme pomocí funkce `demo()`, například

```
> demo(package="stats")
```

# R jako kalkulátor

Konzola příkazového řádku umožňuje interaktívní výpočet výsledků operací a funkcí

```
> 5+3
[1] 8
```

Nevidíme-li úvodní znak příkazového řádku, může to být způsobeno tím, že jsme nezadali úplný příkaz

```
> 5-
+
```

Musíme doplnit zbytek příkazu a poté stlačit Enter nebo zrušit příkaz stlačením klávesy Esc.

# Aritmetické operace v R

| | |
|---|---|
| + | Sčítání. |
| - | Odčítání. |
| * | Násobení. |
| / | Dělení. |
| ^ | Umocňování. |
| %% | Modul (Zbytek po dělení celých čísel). |
| %/% | Celočíselný podíl. |

# Relační operátory v R

| | |
|---|---|
| < | Menší |
| > | Větší |
| <= | Menší nebo rovno |
| >= | Větší nebo rovno |
| == | Rovná se |
| != | Nerovná se |

# Často používané matematické funkce

| | | | |
|---|---|---|---|
| `exp()` | Exponenciála $e^x$ | `sqrt()` | Druhá odmocnina |
| `log()` | Logaritmus (implicitně přirozený) | `abs()` | Absolútní hodnota |
| `log10()` | Logaritmus se základem 10 | | |
| `sin()` | Sinus | `asin()` | Arkussinus |
| `cos()` | Kosinus | `acos()` | Arkuskosinus |
| `tan()` | Tangens | `atan()` | Arkustangens |
| `round()` | Zaokrouhlení (implicitně celé číslo) | | |

# Objekty

R je objektově orientovaný jazyk

V R je všechno objektem a představuje nějaké údaje, které byly uloženy v paměti

Objekty mohou mít libovolné názvy, je však nutno dodržovat tato pravidla:

- název se skládá pouze z malých nebo velkých písmen, číslic, podtržítek a teček,
- název začíná velkým nebo malým písmenem,
- R rozlišuje velikost písmen (to znamená, že A a a jsou dva různé objekty),
- názvem nesmí být žádné z rezervovaných slov R (jejich seznam lze zobrazit zadáním `help(reserved)`),

# Vytváření objektů

Nový objekt vytvoříme snadno pomocí operátoru přiřazení.

Operátor přiřazení má dvě možné podoby: <- nebo =.

Doporučuje se používat <-, protože = může někdy způsobovat chyby:

```
> log(x=25,base=5)
[1] 2
> x
Error: object 'x' not found
> log(x<-25,base=5)
[1] 2
> x
[1] 25
```

# Seznam a odstraňování objektů

Seznam všech existujících objektů získáme jako výstup funkce `ls()`.

Objekty, jež nebudeme v budoucnu používat lze odstranit z paměti pomocí funkce `rm()`.

# Statistika a programování v R

## II. Datové struktury v R

# Datové typy a struktury

Existuje 5 základních datových typů

- numeric,
- integer,
- complex,
- logical,
- character.

Lze je agregovat do datových struktur:

- vector,
- matrix,
- list,
- frame.

# Data tyou `nunmeric`

Data typu `numeric` představují reálná desetinná čísla .

Toto je implicitní typ každého nového objektu.

Vznikne, pokud do libovolné nové proměnné přiřadíme reálné číslo.

Typ libovolného objektu si ověříme pomocí funkce `class()`.

# Data `numeric` – příklad

Podívejme se na příklad.

```
1 > x<-12.35
2 > class(x)
3 [1] numeric
```

**Poznámka**

Číslo je reprezentováno jako vektor délky 1. Znak `[1]` na začátku řádku označuje první pozici v tomto vektoru.

# Data typu integer

Pro vytvoření objektu typu `integer` použijeme funkci `as.integer()`

Příklad

```
> a <- as.integer(12)
> a
[1] 12
> class(a)
[1] "integer"
> is.integer(a)
[1] TRUE
```

# Typ dat complex

Prostředí R umožňuje pracovat i s komplexními čísly.

Komplexní hodnota je v R definována imaginární jednotkou `i`

Příklad

```
> z<-1+2i
> class(z)
[1] "complex"
```

# Typ dat `logical`

Může nabývat dvou logických hodnot `TRUE` nebo `FALSE`

Často vzniká porovnáním proměnných.

```
1  > x<-10;y<-20
2  > z<-x<y
3  > z
4  [1] TRUE
5  > class(z)
6  [1] "logical"
```

# Typ dat `logical`

Jsou pro ně definovány všechny standardní logické operace

| | |
|---|---|
| `&` | Logické `AND` |
| `|` | Logické `OR` |
| `!` | Negace |

# Typ dat character

Slouží k ukládání řetězců znaků, řetězce se zadávají pomocí uvozovek.

```
> x<-"facina"
> class(x)
[1] "character"
#Ale take
> x<-as.character(3.1415926)
> x
[1] "3.1415926"
> class(x)
[1] "character"
```

# Typ dat character

Řetězce znaků lze spojit pomocí `paste()`

```
1  > name<-"Donald"
2  > surname<-"Knuth"
3  > paste(name,surname)
4  [1] "Donald␣Knuth"
5  # Chceme-li
6  > paste(name,surname,sep=",")
7  [1] "Donald,Knuth"
```

# Vektory

Vektor je nejjednodušší datová struktura.

Lze ji charakterizovat jako posloupnost prvků stejného datového typu.

Jednotlivé hodnoty obsažené ve vektoru se označují jako komponenty.

Počet složek vektoru se označuje jako jeho délka.

# Vektory

Vektor `v` je vytvořen pomocí funkce `c()`.

Jeho délka se zjistí pomocí funkce `length()`

```
1 > v<-c(1,3,5,7,9)
2 > length(v)
3 [1] 5
```

# Vektory – aritmetika

Vektorová aritmetika je implementována po jednotlivých komponentách.

Aritmetické operace jsou implementovány po složkách.

- `+` přičtení čísla ke všem komponentám nebo sčítání vektorů po komponentách
- `-` odečte číslo od všech složek nebo odečte vektory složku po složce,
- `*` násobení všech složek číslem nebo násobení vektorů po složkách,
- `/` dělení všech složek číslem nebo dělení vektorů po složkách.

# Vektory – aritmetika

```
> v<-c(1,3,5,7,9)
> u<-c(10,20,30,40,50)
> u+v
[1] 11 23 35 47 59
> u-v
[1] 9 17 25 33 41
> 5*v
[1] 5 15 25 35 45
> u*v
[1] 10 60 150 280 450
> u/5
[1] 2 4 6 8 10
> u/v
[1] 10.000000 6.666667 6.000000 5.714286 5.555556
```

# Matrice

Matice je dvourozměrná tabulka dat stejného typu uspořádaná do obdélníkového schématu.

Vytvoříme ji pomocí funkce `matrix()` s následujícími argumenty

`vector` obsahuje prvky matice,

`nrow` je celočíselná hodnota, která udává počet řádků v matici,

`ncol` je celočíselná hodnota, která určuje počet sloupců v matici,

`byrow` je logická hodnota, která určuje, zda má být matice vyplněna po řádcích (`byrows=TRUE`) nebo po sloupcích (`byrows=FALSE`), její výchozí hodnota je `FALSE`,

`dimnames` je seznam vektorů typu `character`, které obsahují nepovinná označení řádků a sloupců.

# Matrix – příklad zadání

vfill

```
> A<-matrix(3:8,nrow=3,ncol=2,byrow=TRUE)
> A
     [,1] [,2]
[1,]    3    4
[2,]    5    6
[3,]    7    8
> B<-matrix(3:8,nrow=3,ncol=2,byrow=FALSE)
> B
     [,1] [,2]
[1,]    3    6
[2,]    4    7
[3,]    5    8
```

# Matice – výběr podmatice

Řádky a sloupce definujeme pomocí funkce `c()`.

```
> C<-matrix(1:12,nrow=3)
> C
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> C[c(1,3),c(2,4)]
     [,1] [,2]
[1,]    4   10
[2,]    6   12
```

# Matice – přiřazení názvů

Řádkům a sloupcům přiřazujeme názvy pomocí funkcí `dimnames()` a `list()`.

```
> dimnames(A)<-list(c("row1","row2","row3"),
+ c("col1","col2"))
> A
     col1 col2
row1    3    4
row2    5    6
row3    7    8

> A["row2","col1"]
[1] 5
```

# Matice – transpozice

Matrici můžeme transponovat pomocí funkce `t()`

```
1  > B<-matrix(3:8,nrow=3,ncol=2,byrow=FALSE)
2  > t(B)
3       [,1] [,2] [,3]
4  [1,] 3 4 5
5  [2,] 6 7 8
```

Ostatní funkce jsou definovány v balíčku `matlib`.

# Matice – operace

Jsou definovány po jednotlivých složkách

Je to důležité při násobení matic. Běžná operace $*$ znamená násobení prvků na stejných pozicích.

Standardní násobení matic z lineární algebry je definováno jako operace %*%.

# Array

Pole `array` jsou zobecněním maticové datové struktury.

Ve skutečnosti se jedná o více než dvourozměrné matice.

Pole můžeme vytvořit pomocí funkce `array()`.

Syntaxe této funkce je následující

```
name<-array(vector, dimensions,dimnames)
```

# Pole – vytváření

Ukažme si vytvoření pole o rozměrech $3 \times 4 \times 3$.

Pro lepší orientaci v poli nejprve vytvořme názvy jednotlivých dimenzí.

```
1 > dim1<-c("A1", "A2", "A3")
2 > dim2<-c("B1", "B2", "B3", "B4")
3 > dim3<-c("C1", "C2", "C3")
```

# Struktura data frame

Data frame je nejběžnější struktura pro ukládání dat.

Umožňuje ukládat sloupcové vektory různých datových typů.

Data frramy se vytvářejí pomocí funkce `data.frame()`, jejíž obecná syntaxe je následující

```
1 > name<-data.frame(col1,col2,col3, ...)
```

# Data frame – vytváření

Vytvoříme krátký datový rámec obsahující údaje o střelbě basketbalistů.

```
> playerID<-c(1,2,3,4)
> position<-c("forward","guard","forward","center")
> attempted<-c(12,6,10,15)
> made<-c(7,4,6,12)
> players<-data.frame(playerID,position,attempted,made)
> players
  playerID position attempted made
1        1  forward        12    7
2        2    guard         6    4
3        3  forward        10    6
4        4   center        15   12
```

# Data frame – slučování

Často potřebujeme sloučit data ze dvou nebo více datových sad.

Používáme funkci `merge()`.

Argumenty jsou názvy dvou data framů, ježé se mají sloučit.

Třetí argument `by='''column_name'` definuje určující proměnnou pro sloučení dat.

# Data frame – slučování dat

Pro demonstraci sloučení nejprve vytvoříme nový data frame `rebounds`.

```
1 > offensive<-c(5,2,3,10)
2 > defensive<-c(6,3,8,12)
3 > rebounds<-data.frame(playerID,defensive,offensive)
4 > row.names(rebounds)<-c("Player1","Player2","Player3",
5 + "Player4")
```

# Seznam - List

Seznamy představují nejsložitější datovou strukturu.

Představují uspořádané kolekce objektů.

K vytvoření seznamu použijeme funkci `list()`. Jeho syntaxe je jednoduchá:

```
\list(objekt1,objekt2,...)
```

Jeho argumenty jsou názvy existujících objektů

# Seznamy

Volitelně pojmenujte objekty ve vytvořeném seznamu:

```
\list(name1=objekt1,name2=objekt2,...)
```

# Vstup z klávesnice

Nejjednodušší metoda (ale také časově nejnáročnější pro rozsáhlé vzorky)

Pracujeme se ve dvou krocích

- Vytvořte prázdný datový rámec s názvy a typy proměnných, které chceme uložit.
- Otevření jednoduchého editoru dat pomocí funkce `edit()`, jejímž argumentem je název datového rámce, který chceme upravovat.

# Vstup z klávesnice

Vytvoříme prázdný data frame s názvem `mydata` se čtyřmi proměnnými: `name`, která má typ `character`, a třemi číselnými proměnnými `age`, `height` a `weight`.

```
1 > mydata<-data.frame(name=character(0),age=numeric(0),
2 + height=numeric(0),weight=numeric(0))
3 > mydata<-edit(mydata)
```

**Poznámka**

Všimněte si, že přiřazení jako `numeric(0)` a `character(0)` vytvoří proměnnou daného typu, ale bez dat.

# Vstupní data ze souboru `.csv`

Nepovinné argumenty funkce `read.csv()`.

- `header` Logická hodnota, určuje, zda vstupní soubor obsahuje názvy proměnných jako první řádek, výchozí hodnota `TRUE`.
- `sep` určuje znak oddělující položky, výchozí hodnota je čárka,
- `dec` určuje znak použitý v souboru pro desetinná místa, výchozí hodnota je `.`, zmiňme také funkci `read.csv2()`, která používá čárku pro desetinná místa a středník jako oddělovač.
- `skip=n` určuje počet řádků, které se mají přeskočit před zahájením čtení dat. Tato možnost je užitečná pro datové tabulky s prázdnými řádky nebo textovými popisy na začátku souborů.
- `stringsAsFactors`, což je logická hodnota, která určuje, zda se řetězce převedou na faktory, pokud chcete převodu zabránit, nastavte ji na `FALSE`.
- `row.names` vektor s názvy řádků.

# Zápis dat do souboru .csv

R umí vytvořit soubor csv z existujícího data framu.

Používáme funkci `write.csv()` nebo `write.csv2()`, která používá čárku jako desetinnou tečku a středník jako oddělovač.

Běžná syntaxe

```
write.csv(object,file="file_name",...options)
```

`object` je povinný argument obsahující název data framu, který chceme uložit, a `file_name` je název (nebo úplná cesta) souboru.

# Zápis dat do souboru `.csv`

Vybrané možnosti funkce `write.csv()`

- `append`, což je logická hodnota, která udává, zda se výstup připojí na konec souboru. Výchozí hodnota je `FALSE` a všechny existující soubory s tímto názvem budou přepsány.
- `sep` definuje znak oddělovače položek. Hodnoty v každém řádku `object` jsou odděleny tímto znakem.
- `dec` řetězec, který se použije pro desetinné čárky v číselných nebo složených sloupcích, musí to být jeden znak. Výchozí hodnotou je desetinná tečka.
- `row.names` Logická hodnota určující, zda se mají zapsat názvy řádků `object`.

# Vstupní data ze souborů Excel

Existuje několik balíčků, které umožňují importovat data přímo ze souborů aplikace Excel. Uveďme si některé z nich:

- `xlsx`,
- `XLconnect`
- `readxl`

Excel 2007 a novější verze používají formát `xlsx`, proto zde uvedeme balí4ek `xlsx`.

# Vstupní data ze souborů Excel

Balíček nainstalujeme obvyklým příkazem:

```
install.packages("xlsx")
```

Pokud jej chceme použít v aktuálním pracovním prostoru, načteme jej standardním způsobem:

```
library("xlsx")
```

# Vstupní data ze souborů Excel

Tento balíček poskytuje dvě funkce pro načtení obsahu pracovního sešitu Excelu do R data.frame: `read.xlsx()` a `read.xlsx2()`.

Rozdíl mezi těmito dvěma funkcemi je následující:

- `read.xlsx()` zachovává datový typ, typ proměnné odpovídá každému sloupci v pracovním listu, ale je pomalý pro velké datové soubory (pracovní list s více než 100 000 buňkami). `read.xlsx2()` je rychlejší pro velké soubory.

# Vstupní data ze souborů Excel

Obě funkce mají podobnou syntaxi:

```
read.xlsx(file, sheetIndex, header=TRUE, colClasses=NA)
read.xlsx2(file, sheetIndex, header=TRUE, colClasses="character")
```

Jejich argumenty mají následující význam:

- `file` je název souboru, který obsahuje tabulku. Pokud se soubor nenachází v pracovním adresáři, je třeba zadat úplnou cestu.
- `sheetIndex` je číslo udávající index listu, který se má načíst. Lze jej nahradit argumentem `jméno listu`, zadaným jako řetězec znaků s názvem listu.
- `header`logická hodnota. Pokud `header=TRUE`, použije se první řádek jako konvence pro pojmenování proměnných.
- `colClasses` znakový vektor reprezentující třídu každého sloupce.
- `startRow`, `endRow` čísla určující index počátečního a posledního načítaného řádku.

# Zápis dat do souborů Excel

Balíček `xlsx` poskytuje dvě funkce zápisu `write.xlsx()` a `write.xlsx2()`.

Obecná syntaxe

```
write.xlsx(x, file, sheetName="Sheet1", col.names=TRUE,
row.names=TRUE, append=FALSE)

write.xlsx2(x, file, sheetName="Sheet1",col.names=TRUE,
row.names=TRUE, append=FALSE)
```

# Zápis dat do souborů Excel

Jejich argumenty mají následující význam:

- `x` data frame, který se zapíše do sešitu.
- `souborjméno` (nebo cesta) výstupního souboru.
- `sheetName` řetězec znaků s názvem listu.
- `col.names` Logická hodnota, určuje, zda se mají do souboru zapsat názvy sloupců `x`.
- `row.names` Logická hodnota, určuje, zda se do souboru zapíší názvy řádků `x`.
- `append` logická hodnota, určuje, zda má být `x` připojen k existujícímu souboru, pokud `FALSE`, přepíše existující soubor stejnou cestou.

# Čtení dat ze souborů JSON

JSON (JavaScript Object Notation) je jednoduchý formát pro výměnu dat.

Abychom mohli načítat soubory JSON do R, musíme nejprve nainstalovat nebo načíst balíček `rjson`.

Můžeme použít funkci `fromJSON()`

Použití závisí na umístění souboru `.json`.

```
data<-fromJSON(file="jméno souboru.json")
data<-fromJSON(file="URL na soubor json")
```

V obou případech je objekt `data` uložen jako seznam. Pro další analýzu můžeme data převést pomocí funkce `as.data.frame()`.

# Zápis dat do souborů JSONI

Musí být provedeno ve dvou krocích.

V prvním kroku musíme připravit objekt JSON a ve druhém kroku jej zapíšeme do souboru.

Pro vytvoření objektu JSON použijeme funkci `toJSON()`:

```
dataJSON<toJSON(data)
```

Pak použijeme funkci `write()`.

```
write(dataJSON, "filename.json")
```

# Statistika a programování v R

## III. Rozdělení pravděpodobností v R

# Implementované funkce

R umožňuje pracovat s velkým počtem pravděpodobnostních rozdělení.

Pro každé z rozdělení, se kterými R pracuje, jsou implementovány čtyři funkce.

Jejich názvy se skládají z kořenového názvu a předpony:

- `p` pro distribuční funkci,
- `d` pro hustotu nebo pravděpodobnostní funkci,
- `q` pro kvantilovou funkci, inverzní funkci rozdělení,
- `r` náhodná veličina se zadaným rozdělením (generátor náhodných hodnot).

# Diskrétní rozdělení

Pravděpodobnosti jsou určeny seznamem pravděpodobností diskrétních výsledků, známým jako pravděpodobnostní funkce.

Označíme-li množinu všech možných hodnot diskrétní náhodné veličiny $X$ jako $H$ , můžeme zavést pravděpodobnostní funkci $p(x)$ podle vzorce

$$p(x) = \mathbb{P}(X = x), \; x \in H. \tag{1}$$

# Diskrétní rozdělení

Uvedeme některá z nich:

- Bernoulliho rozdělení,
- binomické rozdělení,
- geometrické rozdělení,
- hypergeometrické rozdělení,
- negativní binomické rozdělení,
- Poissonovo rozdělení.

# Bernoulliho rozdělení

Nejjednodušší rozdělení pravděpodobnosti

Jedná se o rozdělení, které má pouze dvě možné hodnoty.

Lze ji interpretovat jako indikátorovou proměnnou, která určuje, zda k náhodné událost nastala, či nikoli.

Nechť $A$ je náhodná událost, $\mathbb{P}(A) = p$ a náhodná veličina $X = 1$, pokud nastane $A$, a $X = 0$ v opačném případě. Pak má pravděpodobnostní funkce tvar

$$p(x) = p^x(1-p)^{1-x} \text{ pro } x = 0 \text{ nebo } 1$$

Je implementována v balíčku `Rlab` jako `bern` s parametrem `prob`.

# Bernoulliho rozdělení

Máme k dispozici čtyři funkce:

- `rbern(n,prob)`, kde `n` je počet pozorování a `prob` je pravděpodobnost náhodné události $A$ (úspěch v experimentu). Generuje vektor `0` a `1` vybraný z Bernoulliho rozdělení s danou pravděpodobností.
- `pbern(q, prob, lower.tail = TRUE, log.p = FALSE)`
- `dbern(x, prob, log = FALSE)`
- `qbern(p, prob, lower.tail = TRUE, log.p = FALSE)`

# Binomické rozdělení

Binomické rozdělení je diskrétní rozdělení, které popisuje počet úspěchů v sérii pokusů se dvěma možnými výsledky.

Formálně nechť $p$ označuje pravděpodobnost úspěchu v jednom pokusu, $n$ počet nezávislých pokusů a $x$ počet úspěchů v posloupnosti $n$ nezávislých pokusů. Náhodná veličina $X$ se řídí binomickým rozdělením, jestliže její pravděpodobnostní funkce má tvar:

$$\mathbb{P}(X = x) = \binom{n}{x} p^x q^{n-x}, \quad x = 0, 1, 2, \ldots, n, \tag{2}$$

kde $q + p = 1$.

# Binomické rozdělení

Pro práci s binomickým rozdělením jsou implementovány 4 funkce:

- `rbinom(n,prob)`, kde `n` je počet pozorování, `p` je pravděpodobnost úspěchu. Tato funkce generuje $n$ náhodných veličin s danou pravděpodobností. `pbinom(x, n, k)`, kde `n` je celkový počet pokusů, `p` je pravděpodobnost úspěchu, `x` je hodnota, pro niž má být pravděpodobnost určena. `dbinom(x, n, p)`, kde `n` je celkový počet pokusů, `p` je pravděpodobnost úspěchu, `x` je hodnota, pro kterou má být pravděpodobnost určena. `qbinom(prob, n, p)`, kde `prob` je pravděpodobnost, `n` je celkový počet pokusů a `p` je pravděpodobnost úspěchu v jednom pokusu. Tato funkce slouží k určení $n$-tého kvantilu, tj. určí $k$ takové, že $P(X \leq k)$ .

# Hypergeometrické rozdělení

Hypergeometrické rozdělení je diskrétní rozdělení pravděpodobnosti, které popisuje pravděpodobnost $x$ úspěchů pro $n$ výběrů bez náhrady z konečné populace o velikosti $N$, která obsahuje přesně $K$ objektů se sledovanou vlastností.

Označme $N$ velikost populace, $K$ počet objektů s danou vlastností v populaci, $n$ počet výběrů, a $x$ počet pozorovaných úspěšných výsledků. Náhodná veličina $X$ se řídí hypergeometrickým rozdělením, jestliže její pravděpodobnostní funkce má tvar

$$\mathbb{P}(X = x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}} \quad x = 0, 1, 2, \ldots, n. \tag{3}$$

# Hypergeometrické rozdělení

Čtyři funkce pro práci s hypergeometrickým rozdělením v R:

- `rhyper(N, m, n, k)`, obecně označuje funkci generování náhodných čísel při zadaných parametrech a velikosti vzorku,
- `phyper(x, m, n, k)` definuje hypergeometrickou distribuční funkci,
- `dhyper(x, m, n, k)` definuje pravděpodobnostní funkci hypergeometrického rozdělení,
- `qhyper(N, m, n, k)` je kvantilová funkce hypergeometrického rozdělení, která se používá k určení posloupnosti pravděpodobností mezi 0 a 1.

Zde `x` představuje soubor hodnot, `m` velikost populace, `n` počet vzorků, `k` počet položek v populaci a `N` hypergeometricky rozdělené hodnoty.

# Negativní binomické rozdělení

Negativní binomické rozdělení je diskrétní rozdělení pravděpodobnosti, které modeluje počet úspěchů v posloupnosti nezávislých a shodně rozdělených Bernoulliho pokusů před výskytem určitého (nenáhodného) počtu neúspěchů (označených $n$).

Označíme-li počet úspěchů $x$ a pravděpodobnost úspěchu $p$, má pravděpodobnostní funkce záporného binomického rozdělení tvar:

$$\mathbb{P}(X = x) = \binom{n + x - 1}{n - 1} p^x q^n, \quad x = 0, 1, 2, \ldots \tag{4}$$

kde $q + p = 1$, $p > 0, q > 0$.

# Negativní binomické rozdělení

Čtyři funkce pro práci s negativním binomickým rozdělením v R:

- `rnbinom(N,n,prob)`, kde `n` je počet pokusů, `N` je velikost vzorku, `prob` je pravděpodobnost úspěchu. Tato funkce generuje $N$ náhodných veličin s danou pravděpodobností.
- `pnbinom(x, n, p)`, slouží k výpočtu hodnoty distribuční funkce záporného binomického rozdělení. Zde `x` je počet selhání před `n`tým úspěchem a `p` je pravděpodobnost úspěchu.
- `dnbinom(x, n, p)` je pravděpodobnost `x` selhání před `n`tým úspěchem (všimněte si rozdílu), když pravděpodobnost úspěchu je `p`.
- `qnbinom(x, n, p)` se používá k výpočtu hodnoty kvantilové funkce negativního binomického rozdělení. Zde `x` je vektor požadovaných úrovní kvantilů, `n` je celkový počet pokusů a `p` je pravděpodobnost úspěchu na pokus.

# Poissonovo rozdělení

Poissonovo rozdělení je diskrétní rozdělení pravděpodobnosti, které vyjadřuje pravděpodobnost výskytu určitého počtu událostí v pevném časovém nebo prostorovém intervalu za předpokladu, že se tyto události vyskytují známou konstantní střední rychlostí a nezávisle na čase, který uplynul od poslední události.

Jestliže náhodná veličina $X$ má Poissonovo rozdělení s parametrem $\lambda > 0$ (průměrný počet událostí), má její pravděpodobnostní funkce tvar

$$\mathbb{P}(X = x) = \frac{\mathrm{e}^{-\lambda}\lambda^x}{x!}, \quad x = 0, 1, 2, \ldots \tag{5}$$

# Poissonovo rozdělení

Čtyři funkce pro práci s Poissonovým rozdělením v R:

- `dpois(x,l)` vypočítá hodnotu pravděpodobnostní funkce $\mathbb{P}(X = x)$ Poissonova rozdělení s parametrem $\lambda$ implementovaným jako argument `l`.
- `ppois(x,l)` počítá distribuční funkci náhodné veličiny, která se řídí Poissonovým rozdělením. Určuje pravděpodobnost $\mathbb{P}(X \leq x)$, argument `l` je parametr rozdělení. Při zadání dalšího argumentu `lower.tail=FALSE` dostaneme pravděpodobnost $\mathbb{P}(X > x)$.
- `rpois(k,l)` slouží ke generování náhodných čísel z daného Poissonova rozdělení, `k` je počet potřebných náhodných čísel a `l` je parametr rozdělení.
- `qpois(q,l)` se používá pro generování kvantilů daného Poissonova rozdělení, `q` je vektor potřebných kvantilových úrovní a `l` je parametr rozdělení.

# Spojité rozdělení

Je to rozdělení pravděpodobnosti, jehož nositelem je nespočetná množina.

Rozdělení je jednoznačně charakterizováno distribuční funkcí

$$F(x) = \mathbb{P}(x \leq x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t.$$

# Spojitá rozdělení

Uveďme si některé z nich:

- rovnonoměrné rozdělení,
- exponenciální rozdělení,
- normální rozdělení,
- Studentovo rozdělení $t$,
- Chí-kvadrát rozdělení,
- Fisherovo rozdělení F.

V R je implementováno mnoho dalších rozdělení.

# Spojitá rozdělení

Uveďme si některé z nich:

- rovnonoměrné rozdělení,
- exponenciální rozdělení,
- normální rozdělení,
- Studentovo rozdělení $t$,
- Chí-kvadrát rozdělení,
- Fisherovo rozdělení F.

V R je implementováno mnoho dalších rozdělení.

Budeme se zabývat, třemi „modrými“ rozděleními.

# Rovnoměrné rozdělení

Spojité rovnoměrné rozdělení popisuje experiment, v němž je možný jakýkoli výsledek, který leží mezi určitými hranicemi.

Přesněji řečeno, náhodná veličina $X$ má rovnoměrné rozdělení s parametry $a$ a $b$, $a < b$, pokud má její funkce hustoty tvar: $vfill

$$f(x) =, \begin{cases} \frac{1}{b-a} & x \in \langle a; b \rangle \\ 0 & \text{jinak} \end{cases} \tag{6}$$

# Rovnoměrné rozdělení

Čtyři funkce pro práci s rovnoměrným rozdělením v R:

- `dunif()`, která definuje funkci hustoty, jejímiž argumenty jsou vektor `x` a parametry `min` a `max` rozdělení,
- `punif()`, která definuje distribuční funkci, jejími argumenty jsou vektor `x` a parametry `min` a `max` distribuce,
- `qunif()`, která poskytuje kvantilovou funkci, jejímiž argumenty jsou kvantily `q` a parametry `min` a `max` rozdělení,
- `runif()`, který generuje náhodné hodnoty proměnné, jeho argumenty jsou velikost vzorku `n` a parametry `min` a `max` rozdělení.

# Exponenciální rozdělení

Exponenciální rozdělení popisuje dobu čekání mezi událostmi v Poissonově procesu, tj. procesu, ve kterém se události vyskytují kontinuálně a nezávisle s konstantní průměrnou četností.

Náhodná veličina $X$ se řídí exponenciálním rozdělením s parametrem $\lambda > 0$ (obvykle se udává jako četnost), pokud má její funkce hustoty tvar:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{7}$$

Pokud je parametr $\lambda$ interpretován jako četnost, je průměrná čekací doba $\frac{1}{\lambda}$.

# Exponenciální rozdělení

Čtyři funkce pro práci s exponenciálním rozdělením v jazyce R:

- `dexp()`, která představuje funkci hustoty, jejímiž argumenty jsou vektor `x` a parametr `rate` rozdělení,
- `pexp()`, což je distribuční funkce, jejímiž argumenty jsou vektor `x` a parametr `rate` distribuce,
- `qexp()`, která určuje kvantilovou funkci, jejími argumenty jsou kvantily `q` a parametr `rate` rozdělení,
- `rexp()`, která generuje náhodné hodnoty proměnné, jejími argumenty jsou velikost vzorku `n` a parametr `rate` rozdělení.

# Normální rozdělení

Normální (neboli Gaussovo) rozdělení je typem spojitého rozdělení pravděpodobnosti, jehož funkce hustoty je následující.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{8}$$

Parametr $\mu$ je střední hodnota rozdělení (a také jeho medián a modus), parametr $\sigma$ je jeho směrodatná odchylka.

Normální rozdělení je důležité díky centrální limitní větě, která říká, že populace všech možných vzorků velikosti $n$ z populace se střední hodnotou $\mu$ a rozptylem $\sigma^2$ konverguje normálnímu rozdělení se střední hodnotou $\mu$ a $\frac{\sigma^2}{n}$, jak $n$ roste do nekonečna.

# Normální rozdělení

Čtyři funkce pro práci s normálním rozdělením v R:

- `dnormf()`, což je funkce hustoty, jejímiž argumenty jsou vektor `x` a parametry `mean` a `sd` rozdělení,
- `pnorm()`, která představuje distribuční funkci, jejími argumenty jsou vektor `x` a parametry `mean` a `sd` rozdělení
- `qnorm()`, která představuje kvantilovou funkci, jejímiž argumenty jsou kvantily `q` a parametry `mean` a `sd` rozdělení,
- `rnorm()`, který generuje náhodné hodnoty proměnné, jeho argumenty jsou velikost vzorku `n` a parametry `mean` a `sd` rozdělení.

# Statistika a programování v R

## IV. Programování v R

# Funkce

Téměř všechny akce v jazyce R se provádějí pomocí funkcí.

Je implementována bohatá škála vestavěných funkcí.

Uživatel může definovat další funkce

Vestavěné funkce lze rozdělit na

- matematické funkce,
- řetězcové functions,
- specializované statistické a pravděpodobnostní funkce,
- další užitečné funkce.

# Matematické funkce

O některých z nich jsme se již zmínili v lekci 1.

Zde uvedeme několik dalších podrobností

Logaritmická funkce `log()` vypočítá přirozený logaritmus jako výchozí hodnotu.

Pro získání logaritmu s libovolným základem je třeba deklarovat argument `base` funkce `log()`.

```
1 > log(4)
2 [1] 1.386294
3 > log(4,base=2)
4 [1] 2
```

# Řetězcové funkce

Funkce `nchar()` určuje velikost každého prvku vektoru znaků.

```
1  > z<-c("yellow","black","white")
2  > nchar(z)
3  [1] 6 5 5
4  > str<-"This␣is␣a␣long␣string"
5  > nchar(str)
6  [1] 21
```

# Elementární statistické funkce

- `mean()` Průměrná hodnota vzorku.
- `median()` Medián vzorku.
- `sd()` Směrodajná odchylka.
- `var()` Výběový rozptyl .
- `mad()` Absolutní odchylka mediánu.
- `quantile()` Výběrové kvantily, implicitní jsou kvartily.
- `range()` Rozsah hodnot.
- `sum()` Součet prvků vektoru.
- `min()` Minimum.
- `max()` Maximum.

# Elementární statistické funkce – `mean()` nepovinné argumenty

`trim`, který určuje procento nejvyšších a nejnižších hodnot, které jsou z výpočtu vynechány, a vrací tak ořezaný průměr.

Druhý nepovinný argument `na.rm` je logická hodnota, která určuje, zda se mají před pokračováním výpočtu odstranit hodnoty `NA`.

```
1  > x<-c(1,3,5,10,12)
2  > mean(x)
3  [1] 6.2
4  > mean(x,trim=0,2)
5  [1] 6
```

```
1  > x<-c(1,5,2,12,NA,3,6)
2  > mean(x)
3  [1] NA
4  > mean(x,na.rm=TRUE)
5  [1] 4.833333
6  > mean(x,na.rm=TRUE,trim=0.17)
7  [1] 4
```

# Elementární statistické funkce – `mad()`

Absolutní mediánová odchylka je robustní míra variability jednorozměrného vzorku kvantitativních dat.

Pro vzorek $X_1, \ldots, X_n$ je definován vzorcem:

$$\mathrm{MAD}(X) = \mathrm{median}\{|X_i - \overline{X}|\}$$

```
1  > mad(delay)
2  [1] 13.3434
```

# Užitečné funkce – seq()

Funkce `seq()` generuje posloupnost čísel začínající `from` a končící `to`. Poslední argument `by` určuje krok posloupnosti.

```
> seq(10)
 [1] 1 2 3 4 5 6 7 8 9 10
> seq(5,15)
 [1] 5 6 7 8 9 10 11 12 13 14 15
> seq(5,15,2)
[1] 5 7 9 11 13 15
```

# Užitečné funkce – rep()

Funkce rep() má dva argumenty, vektor x, který se má opakovat, a počet cyklů opakování n.

```
> rep(1,10)
 [1] 1 1 1 1 1 1 1 1 1 1
> rep(c(1,3),4)
[1] 1 3 1 3 1 3 1 3
> rep("hello",3)
[1] "hello" "hello" "hello"
```

# Užitečné funkce – `sort()` a `order()`

Funkce `sort()` a `order()` jsou spojeny s uspořádáním prvků vektoru `x`.

`sort()` poskytuje vzestupně seřazené hodnoty, zatímco `order()` poskytuje indexy seřazených komponent v původním vektoru.

```
1  > x<-c(5,2,10,3,7,8)
2  > sort(x)
3  [1]  2  3  5  7  8 10
4  > order(x)
5  [1]  2  4  1  5  6  3
```

# Užitečné funkce – `rev()`

Dává vektor `x` v opačném pořadí prvků

```
> rev(x)
[1]  8  7  3 10  2  5
> rev(sort(x))
[1] 10  8  7  5  3  2
```

# Podmíněné příkazy – `if`

`if()` příkaz provádí operace na základě jednoduché podmínky

`if (podmínka) {příkaz, který se provede, pokud podmínka platí}`

Více než jeden výrok musí být v závorkách

```
1  > x<-5
2  > if(x%%2){print("Odd number")}
3  [1] "Odd number"
4  > x<-6
5  > if(x%%2){print("Odd number")}
6  >
```

# Podmíněné příkazy – switch

`switch()` testuje výraz oproti prvkům seznamu. Každá hodnota v seznamu se nazývá case

Syntaxe funkce `switch()`:

```
switch (výraz, seznam)
```

```
1 > x<-10
2 > switch(x%%2+1, "even", "odd")
3 [1] "even"
4 > x<-9
5 > switch(x%%2+1, "even", "odd")
6 [1] "odd"
```

# Podmíněné příkazy – switch

Pokud je výrazem řetězec znaků, funkce switch() vrátí hodnotu na základě názvu prvku.

```
> x <- "a"
> switch(x, "a"="apple", "b"="banana", "c"="cherry")
[1] "apple"
> x <- "c"
> switch(x, "a"="apple", "b"="banana", "c"="cherry")
[1] "cherry"
```

# Cyklus – `for`

Cyklus `for` nám umožňuje pevný počet opakování příkazu nebo bloku příkazů.

Obecná syntaxe cyklu `for` je následující:

```
for (val in sequence)
{
příkaz
}
```

kde `sequence` je vektor a `val` nabývá během cyklu každé z hodnot sequence.

# Uživatelem definovaná funkce

Obecná struktura funkce je

```
myfunction_name <- function(arg1, arg2, ... ){
příkazy
return(objekt)
}
```

# Uživatelem definovaná funkce

Jednotlivé složky funkce jsou:

- **Název funkce**, což je skutečný název funkce. Je uložen v prostředí R jako objekt s tímto názvem.
- **Argumenty**, což jsou zástupné znaky. Při volání funkce předáváme hodnoty argumentů. Argumenty jsou nepovinné, to znamená, že funkce nemusí obsahovat žádné argumenty. Argumenty mohou mít také výchozí hodnoty.
- **Tělo funkce**, která obsahuje sadu příkazů definujících, co funkce dělá. Tělo funkce se nachází uvnitř složených závorek {}.
- **Návratová hodnota**, což je poslední výraz v těle funkce, který se vyhodnocuje.

# Spouštění skriptů v R

Skript R je jednoduše textový soubor obsahující (téměř) stejné příkazy, jaké byste zadali při psaní

Lze jej vytvořit v libovolném jednoduchém textovém editoru a uložit s příponou `.R`.

Pro spuštění skriptu v systému Linux existují v zásadě dva příkazy.

```
Rscript filename.R
```

který je preferován. Starší příkaz je

```
R CMD BATCH název souboru.R
```

# Statistika a programování v R

## V. Základní grafika v R

# Bodové grafy

Bodový (také korelační) graf zobrazuje hodnoty dvou různých číselných proměnných.

Poloha každého bodu odpovídá jednotlivým hodnotám dat na vodorovné a svislé ose.

Nejčastější aplikace a použití bodových grafů jsou:

1. Demonstrace vztahu mezi dvěma proměnnými.
2. Identifikace korelačních vztahů.
3. Identifikace datových modelů.

# Bodové grafy

Vytvoříme je jednoduše pomocí funkce `plot()`.

Nejjednodušeji funkce přijímá dva argumenty `x` a `y`.

Tyto proměnné jsou vektory, které obsahují hodnoty, jež chceme vykreslit.

Délka vektorů musí být stejná.

# Jak uložit obrázek

Případně můžeme výstup z obrazovky přesměrovat do souboru.

Můžeme použít funkce

| | |
|---|---|
| `pdf()` | Vektor `pdf` formát, nejlepší volba při použití s `pdflatex` |
| `svg()` | Vektorový `svg` formát, snadná změna velikosti. |
| `postscript()` | Vektorový postscriptový formát `ps`, snadná změna velikosti. |
| `png()` | Bitmapový formát s vysokým rozlišením, nelze bez ztráty měnit velikosti. |
| `jpeg()` | Komprimovaný formát bitmapy, nemění velikost beze ztrát. |
| `bmp()` | Bitmapový formát s vysokým rozlišením, nemění velikost bezeztrátově. |
| `tiff()` | Bitmapový formát s vysokým rozlišením, nemění velikost bezeztrátově. |

# Možnosti ukládání grafů

| | |
|---|---|
| `filename` | Název uloženého souboru, v případě potřeby s úplnou cestou. |
| `width` | Šířka výsledného grafu, výchozí hodnota 7 in. |
| `height` | Výška výsledného grafu, výchozí hodnota 7 in. |
| `res` | rozlišení obrázku, platí pro formáty bitmap, výchozí 72 dpi. |
| `units` | Jednotky míry. |
| `bg` | Barva pozadí. |
| `fg` | Barva popředí. |
| `family` | Použité písmo (výchozí Helvetica). |

# Modifikace grafu – značkovací body

Značka bodu je dána hodnotou argumentu `pch` funkce `plot()`.

Možné hodnoty

| | | | | |
|---|---|---|---|---|
| □ pch=0 | ○ pch=1 | △ pch=2 | + pch=3 | × pch=4 |
| ◇ pch=5 | ▽ pch=6 | ⊠ pch=7 | ✳ pch=8 | ⨁ pch=9 |
| ⊕ pch=10 | ✡ pch=11 | ⊞ pch=12 | ⊗ pch=13 | ⧄ pch=14 |
| ■ pch=15 | ● pch=16 | ▲ pch=17 | ♦ pch=18 | ● pch=19 |
| • pch=20 | ○ pch=21 | □ pch=22 | ◇ pch=23 | △ pch=24 |

# Modifikace grafu – typ spojnice

Typ spojnice bodů se nastavuje pomocí argumentu `type` funkce `plot()`.

Možné hodnoty

| | |
|---|---|
| `p` | Bodový graf, výchozí hodnota. |
| `l` | Souvislá čára. |
| `b` | Souvislá čára s body. |
| `c` | Části spojitých čar s vynechanými body. |
| `o` | Části souvislých čar, body překresleny. |
| `h` | Graf podobný histogramu. |
| `s` | Schodovitý graf. |

# Modifikace grafu – styl spojnice

Styl čáry se nastavuje pomocí argumentu `lty` funkce `plot()`.

Možné hodnoty

| | | | |
|---|---|---|---|
| 1 | Tlustá čára (výchozí). | 2 | Čárkovaná čára. |
| 3 | Tečkovaná čára. | 4 | Tečky a čárky. |
| 5 | Dlouhé čárky. | 6 | Dlouhá a krátká dvojité přerušované čárky. |

Šířka čáry se nastavuje pomocí argumentu `lwd` funkce `plot()`.

# Modifikace grafu – barvení

Barvy můžeme upravit pomocí

- názvu barev položek, například `col=red`
- číslo barvy položky, například `col=636`
- podle hexadecimálního kódu (v režimu RGB), například `col="#FFCC00"`

Seznam dostupných barev lze získat jako výstup funkce `colors()`.

# Modifikace grafu – barvení

Další možnosti obarvení jsou

| | | | |
|---|---|---|---|
| `col.axis` | Barva popisků os. | `col.lab` | Barva označení os. |
| `col.main` | Barva hlavního nadpisu. | `col.sub` | Barva podnadpisu. |
| `bg` | Barva výplně znaku. | `fg` | Barva podkladu grafu. |

# Modifikace grafu – barvení

Barvy jako vektory

Hodnotu argumentu `col` můžeme nastavit jako vektor.

Barvy z vektoru se pak pravidelně střídají.

Můžeme také použít funkci `rainbow()` s předem definovanou posloupností barev.

# Úpravy grafu – názvy a titulky

Základní kreslicí funkce v R obsahují argument `main`, který umožňuje přidat ke grafu nadpis.

Můžeme také použít argument `sub` pro přidání podnadpisu, který bude umístěn pod grafem.

Alternativním způsobem, jak do grafu přidat nadpis a podnadpis, je použití funkce `title()`.

# Úprava grafu – přidání textu do grafu

Do nakresleného grafu můžeme přidat libovolný text pomocí funkcí `text()` a `mtext()`.

Funkce `text()` umístí zadaný text na libovolné místo v kreslicí oblasti, funkce `mtext()` umístí text na okraje.

Funkce `text()` přijímá dva další argumenty:

- `location` definuje souřadnice `x` a `y`, kde bude text umístěn. Souřadnice musí být zadány jako první dva argumenty funkce.
- `pos` určuje umístění ve vztahu k aktuální poloze, 1=dolů, 2=doleva, 3=nahoru a 4=doprava. Definování pozice jako `locator(1)` umožňuje umístění textu pomocí myši.

# Úprava grafu – uzpůsobení os

Pokud chceme odstranit rámeček grafu, nastavíme argument kreslicí funkce `axes=FALSE`.

Nové osy přidáme pomocí funkce `axes()`.

Argument funkce `axis()` určuje stranu grafu, na kterou bude osa přidána.

Jako obvykle čísla určují strany: 1=spodní, 2=levá, 3=horní a 4=pravá.

Zkusme

```
1 > plot(sort(x),y[order(x)],pch=17,type="b",col=30,axes=FALSE)
2 > axis(1)
3 > axis(2)
```

# Další nastavení os

Můžeme také:

- určit počet značek se zadanými počátečními a koncovými hodnotami,
- upravit délku a orientaci značek,
- otáčet popisky značek,
- přizpůsobit popisy značek,
- odstranit značky,
- přidat menší značky pomocí `Hmisc` .

# Další nastavení os – délka a orientace značek

Argument `tck` umožňuje nastavit délku a orientaci dělících značek.

Jeho kladná hodnota orientuje značky dovnitř kreslící plochy, zatímco záporné hodnoty orientují značky vně kreslící plochy. Čím větší je absolutní hodnota, tím delší jsou značky. Výchozí hodnota je `tck=-0,05`.

Rotace je omožněna argumentem `las`, který může nabývat jedné ze čtyř hodnot:

- `las=0` popisky jsou rovnoběžné s osou (výchozí),
- `las=1` všechny popisky jsou vodorovné,
- `las=2` popisky jsou kolmé na osu,
- `las=3` všechny popisky jsou svislé.

# Rozsah os a přizpůsobení

Rozsah hodnot pro osy lze definovat pomocí nepovinných argumentů `xlim` a `ylim` funkce `plot()`.

Hranice jsou zadány jako vektory ve tvaru `c(start,end)`

Můžeme také transformovat osy do logaritmického měřítka tak, že nastavíme argument `log` na hodnotu osy, kterou plánujeme přizpůsobit.

`log="x "` nastaví logaritmickou stupnici na osu `x`,
`log="y "` nastaví logaritmickou stupnici na osu `y` a
`log="xy "` transformuje obě osy do logaritmického měřítka.

# Duální svislé osy

**Príklad.**
Do jednoho grafu chceme zakreslit dvě charakteristiky zdravotního stavu pacientů, teplotu a krevní tlak.

# Duální svislé osy

**Príklad.**
Do jednoho grafu chceme zakreslit dvě charakteristiky zdravotního stavu pacientů, teplotu a krevní tlak.

V proměnných y a z máme uloženy údaje o 100 pacientech, přičemž proměnná x obsahuje posloupnost identifikátorů pacientů, čísla od 1 do 100.

# Křivky

Jednou z mnoha užitečných funkcí v R je `curve()`.

Je to malá šikovná funkce, která umožňuje vykreslovat křivky, např. grafy funkcí.

Funkce `curve()` přijímá jako první argument výraz v syntaxi R.

Například,

```
curve(x^2)
curve(x^2,xlim=c(-2,2),col="red",lwd=2)
```

# Zobrazení dvou nebo více křivek v jednom grafu

Použijeme funkci `curve()` s argumentem `add=TRUE`.

Například

```
curve(x^2)
curve(sqrt(x),col="red",lwd=2,add=TRUE)
```

# Přidání legendy

Funkce `legend()` umožňuje přidat ke grafům legendu.

Některé argumenty:

- `x,y` pozice v kreslicí oblasti definovaná souřadnicemi v grafu,
- `legend`vektor řetězců pro popis v legendě,
- `col` vektor barev použitých v grafu,
- `pch` vektor tvarů značek použitých v grafu,
- `lty` vektor typů čar použitých v grafu,
- `ncol` počet sloupců použitých v legendě, výchozí hodnota je jeden sloupec.

# Sloupcové grafy

Sloupcový graf zobrazuje kategoriální data pomocí obdélníkových sloupců, jejichž výška nebo délka je úměrná hodnotám, které představují.

R používá funkci pro vytváření sloupcových grafů

```
barplot(H,xlab,ylab,title, names.arg,col)
```

Parametry použité ve funkci jsou následující:

- `H` je vektor nebo matice obsahující číselné hodnoty použité ve sloupcovém grafu,
- `xlab` je označení osy x,
- `ylab`je označení osy y,
- `title` je nadpis sloupcového grafu,
- `names.arg` je vektor popisů, které se zobrazují pod každým sloupcem,
- `col` slouží k přiřazení barev sloupcům v grafu.

# Sloupcové grafy – obarvení a popisky sloupců

Pro přiřazení názvů sloupcům používáme parametr `names.arg` sloupcového grafu.

Dále definujeme hodnoty parametrů

- `xlab` a `ylab` pro názvy os,
- `col` a `border` pro obarvení sloupců, a
- `main` pro definování nadpisu grafu

Je to podobné jsko u funkce `plot()`.

# Histogramy

Histogram je zobrazení přibližného rozložení číselných dat.

Zobrazuje četnosti hodnot rozdělených do intervalů.

Histogram je podobný sloupcovému grafu s tím rozdílem, že sdružuje hodnoty do souvislých intervalů.

Histogramy poskytují přibližný obraz hustoty základního rozdělení dat.

# Histogramy

Histogram lze vytvořit pomocí funkce `hist()` v R.

```
barplot(H,xlab,ylab,title, names.arg,col)
```

Parametry použité ve funkci jsou následující:

- `data` je vektor obsahující číselné hodnoty použité v histogramu,
- `main` je nadpis grafu,
- `col` slouží k nastavení barvy sloupců,
- `border` slouží k nastavení barvy rámečku každého sloupce,
- `xlab` určuje popis osy x,
- `xlim` určuje rozsahu hodnot na ose x,
- `ylim` slouží k určení rozsahu hodnot na ose y,
- `breaks` slouží k určení šířky každého sloupce.

# Kruhové diagramy

Kruhový diagram je graf pro jednu kategoriální proměnnou a je alternativou sloupcového grafu.

Kruhový diagram (nebo koláčový diagram) je graf ve tvaru kruhu rozděleného na výseče, jež znázorňují poměry mezi veličinami.

V kruhovém diagramu je délka oblouku každého dílku (a tedy i jeho středový úhel a plocha) úměrná množství, které představuje.

# Kruhové diagramy

Základní syntaxe pro vytvoření kruhového diagramu v prostředí R je následující:

```
pie(data, labels, radius, main, col, clockwise)
```

Význam argumentů:

- `data` je vektor obsahující číselné hodnoty použité v kruhovém diagramu,
- `labels`se používá k popisu dílků,
- `radius` označuje poloměr kruhu diagramu (hodnota mezi $-1$ a $+1$),
- `main` označuje nadpis grafu,
- `col` označuje paletu barev,
- `clockwise` je logická hodnota, která určuje, zda jsou plátky vykresleny ve směru nebo proti směru hodinových ručiček.

# Kruhový diagram – úprava barev

Pro změnu barev v grafu použijeme funkci `rainbow()`, která definuje paletu barev.
Jeho argumenty jsou následující:

- `n` počet barev ($\geq 1$), které mají být v paletě,
- `s,v` „sytost“ a „hodnota“ pro přidání popisu k barvám
- `start` (upravený) odstín v $\langle 0; 1\rangle$, od kterého začíná vybraná duha,
- `end` (uopravený) odstín v $\langle 0; 1\rangle$, ve kterém duha končí,
- `gamma` gama korekce pro každou barvu, `(r,g,b)` v prostoru RGB (se všemi hodnotami v $\langle 0; 1\rangle$), výsledná barva odpovídá $(r^\gamma, g^\gamma, b^\gamma)$,
- `alfa` průhlednost, číslo ve tvaru $\langle 0; 1\rangle$, (0 znamená průhledný a 1 znamená neprůhledný).

# Kruhový diagram–použití `rainbow()`

```
1 description<-paste(labels,"\n",data,sep="")
2 pie(data,description,main="Monthly␣expenses",
3 col=rainbow(length(data)))
```

**Poznámka**

Změnili jsme také popisky. K jejich názvům jsme přidali číselné hodnoty.

# Vějířový graf

Užitečnou alternativou ke kruhovým diagramům je `fan.plot()` definovaný v balíčku `plotrix`.

Umožňuje vizuálně porovnat sektory kruhového diagramu.

Vějířový graf můžeme přizpůsobit zadáním dalších argumentů:

- `max.span` úhel maximálního sektoru v radiánech. Ve výchozím nastavení se `data` škáluje tak, aby se součet rovnal $2\pi$.
- `ticks` počet políček, která by se objevila, kdyby byly sektory v kruhovém diagramu. Výchozí nastavení je bez políček.

# Krabicový graf

V popisné statistice je krabicový graf nebo boxplot typem grafu, který se často používá při vysvětlování analýzy dat.

Krabicové grafy znázorňují rozptýlení číselných dat a šikmost zobrazením kvartilů (nebo percentilů) a průměrů dat.

Je také užitečný při porovnávání rozložení dat v různých souborech dat, kdy pro každý z nich nakreslíme krabicový graf.

# Krabicový graf

Krabicový grafse skládá ze dvou částí, boxu a sady „whiskers“.

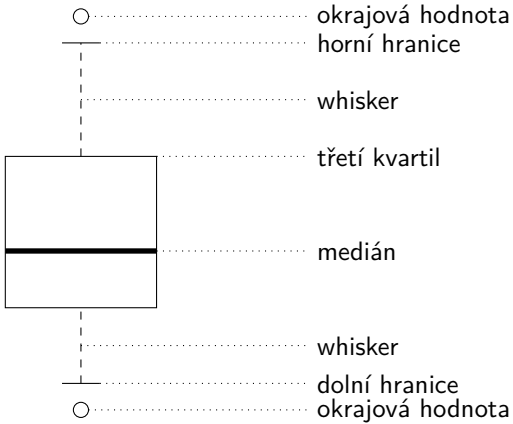
Box zobrazuje rozmezí od dolního kvartilu po horní kvartil, přičemž vodorovná čára vedená uvnitř boxu označuje medián.

Whiskery mohou několik odlehlých hodnot mezi pozorovanými daty dle jedné z alternativ:

- minimum a maximum všech údajů,
- s jednou směrodatnou odchylkou nad a pod průměrem dat,
- 9. percentil a 91. percentil,
- 2. percentil a 98. percentil.

Všechna data, která nejsou zahrnuta mezi whiskery, by měla být vykreslena jako okrajové hodnoty označené tečkou, malým kroužkem nebo hvězdičkou, což se však někdy vynechává.

# Krabicový graf



# Q-Q graf

Kvantilový graf (zkráceně Q-Q graf) je grafický nástroj, který nám pomáhá posoudit, zda soubor dat věrohodně odpovídá nějakému teoretickému rozdělení, například normálnímu nebo exponenciálnímu rozdělení.

Pokud například provádíme statistickou analýzu, která předpokládá, že naše závislá proměnná je normálně rozdělena, můžeme k ověření tohoto předpokladu použít normální Q-Q graf.

Jedná se pouze o vizuální kontrolu, nikoli o exaktní důkaz, ale umožňuje nám na první pohled zjistit, zda je náš předpoklad věrohodný, a pokud ne, jak je předpoklad porušen a které hodnoty dat k porušení přispívají.

# Q-Q graf

Q-Q graf je v podstatě bodový graf vytvořený vynesením dvou sad kvantilů proti sobě.

Pokud obě sady kvantilů pocházejí ze stejného rozdělení, nachází se body přibližně na přímce.

Q-Q grafy vezmou naše vzorky, seřadí je vzestupně a pak je vynesou do grafu proti kvantilům navrženého teoretického rozdělení.

Počet kvantilů je zvolen tak, aby odpovídal velikosti našeho vzorku.

# Více grafů v jednom obrázku

V prostředí R můžeme grafy kombinovat užitím grafických parametrů `mfrow` a `mfcol`.

Stačí zadat vektor, který určuje počet řádků a počet sloupců, které plánujeme vytvořit.

Rozhodnutí o tom, který parametr grafu použijeme, závisí na tom, jak chceme mít grafy uspořádané:

- `mfrow` grafy budou uspořádány po řádcích,
- `mfcol` grafy budou uspořádány po sloupcích.

Toto nastavení se používá jako argument funkce `par()`, která upravuje parametry grafického zařízení.

# Statistika a programování v R

## VI. Výběrové charakteristiky

# Průměr

Průměr dané množiny hodnot $x_1, x_2, \ldots, x_n$ je dán vztahem:

$$\overline{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}. \tag{9}$$

V jazyce R je implementován jako funkce `mean()`.

Její použití je velmi jednoduché.

# Průměr

Na internetových stránkách slovenské centrální banky https://www.nbs.sk/en/monetary-policy/macroeconomic-database nalezneme makroekonomickou databázi

Můžeme si stáhnout například soubor `.csv`, jenž obsahuje počty registrací nových osobních automobilů ve zvoleném časovém období.

Tento soubor je implicitně uložen jako `makrostat.csv`.

Vzhledem k použití čárky jako oddělovače desetinných míst načteme data pomocí funkce `read.csv2()`.

# Průměr

Pro výpočet průměru musíme použít `as.numeric()`, protože `cars[1,]` poskytuje hodnoty ve formátu seznamu.

Pro získání průměrné hodnoty měsíčního počtu nově registrovaných osobních automobilů (v tisících) v letech 2017-18 tedy použijeme kód:

```
1 cars<-read.csv2("macrostat.csv",header=FALSE,sep=";")
2 mean(as.numeric(cars[1,]))
3 [1] 8.090125
```

# Průměr

Často se stává, že hodnoty statistického znaku, který nás zajímá, jsou uspořádány v posloupnosti absolutních četností.

V tomto případě upravíme vztah (9) pro výpočet výběrového průměru do tvaru:

$$\overline{x} = \frac{x_1 \cdot n_1 + x_2 \cdot n_2 + \cdots + x_k \cdot n_k}{n_1 + n_2 + \cdots + n_k} = \frac{\sum_{i=1}^{k} x_i \cdot n_i}{\sum_{i=1}^{k} n_i}. \tag{10}$$

kde $x_i$ představují hodnoty proměnné a $n_i$ jejich absolutní četnosti.

# Průměr

V tomto případě musíme definovat vlastní funkci pro výpočet střední hodnoty.

Jako vstupní hodnoty zadáme dva vektory. První vektor obsahuje hodnoty, kterých náhodná veličina nabývá, a druhý je vektor jejich četností.

Před provedením výpočtu podle vztahu (10) je nutno ověřit, zda mají oba vektory stejnou délku.

# Medián

Medián lze charakterizovat jako střední hodnotu, pokud jsou pozorované hodnoty seřazeny od nejmenší po největší.

Formálně můžeme říci, že pravdľpodobnost, že hodnota proměnné je větší než medián, je rovna pravděpodobnosti, že její hodnota je menší než medián, a tudíž se tato pravděpodobnost rovná $\frac{1}{2}$.

Máme-li vzorek hodnot $x_1, x_2, \ldots, x_n$, je medián $\tilde{x}$ dán vztahy

$$\tilde{x} = \begin{cases} \left(\frac{n+1}{2}\right)\text{-tá uspořádaná odnota} & n \text{ je liché} \\ \frac{1}{2}\left(\left(\frac{n}{2}\right)\text{-tá} + \left(\frac{n}{2}\right)\text{-tá uspořádaná hodnota}\right) & n \text{ je sudé} \end{cases} \tag{11}$$

# Kvantily

Pro určeníní kvantilů je v R implementována funkce `quantile()`. Bez zadání volitelných parametrů dává minimum vzorku, první kvartil, medián, třetí kvartil a maximum vzorku.

Můžeme si to ilustrovat na datech COVID-19, stažených z oficiálních stránek slovenské vlády https://korona.gov.sk.

```
1  data<-read.csv("https://mapa.covid.chat/export/csv",
2     header=T,sep=";")
3  > quantile(data[,4])
4     0%    25%    50%    75%   100%
5      0     30    232   1737  15278
6  >
```

# Variační rozpětí

Variační rozpětí závisí pouze na dvou hodnotách ve vzorku – na největší a nejmenší hodnotě.

Je definováno jako rozdíl

$$R = \max\{x_1, \ldots, x_n\} - \min\{x_1, \ldots, x_n\} \tag{12}$$

Je dobře použitelné pro rychlou orientaci v rozsahu variability daného vzorku.

# Variační rozpětí

Variační rozpětí je v prostředí jazyka R výstupem funkce `range()` I Jejím výstupem jsou dvě hodnoty - největší a nejmenší hodnota ve vzorku.

Abychom mohli vyjádřit rozsah variace jako jedinou hodnotu podle definice (12), použijeme funkce `max()` a `min()`.

# Mezikvartilové rozpětí

Mezikvartilové rozpětí je mírou statistického rozptylu.

Je definováno jako rozdíl mezi horním a dolním kvartilem vzorku.

Přiřadíme-li těmto kvartilům hodnoty $Q_3$ a $Q_1$, můžeme mezikvartilové rozpětí $IQR$ vyjádřit takto

$$IQR = Q_3 - Q_1. \tag{13}$$

# Střední absolutní odchylka

Střední absolutní odchylka *MAD* souboru dat je průměrná vzdálenost každého datového bodu od průměru.

Pro vzorek $x_1, \ldots, x_n$ je definována vzorcem

$$MAD = \frac{1}{n} \sum_{i=1}^{n} |x_i - \overline{x}| . \tag{14}$$

# Rozptyl a standardní odchylka

Rozptyl je nejznámější a nejčastěji používanou mírou variability vzorku.

Máme-li vzorek $x_1, \ldots, x_n$, definujeme výběrový rozptyl $s^2$ podle vzorce

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2 . \tag{15}$$

# Rozptyl a standardní odchylka

V případě dat uspořádaných v posloupnosti absolutních četností $n_1, \ldots, n_k$, $\sum_{i=1}^{k} n_i = n$, hodnot $x_1, \ldots, x_k$, upravíme vzorec (15) do tvaru

$$s^2 = \frac{1}{n} \sum_{i=1}^{k} n_i \left(x_i - \overline{x}\right)^2 . \tag{16}$$

# Rozptyl a standardní odchylka

Funkce `var()` a `sd()` musíme používat opatrně.

Jejich výsledkem je nestranný odhad rozptylu a směrodatné odchylky celé populace.

Chceme-li vypočítat výběrový rozptyl podle vztahu (15), musíme definovat vlastní funkci, což ilustrujeme v následujícím zdrojovém kódu.

# Rozptyl a standardní odchylka

```
> variance<-function(x) sum((x-mean(x))^2)/length(x)
> stdev<-function(x) sqrt(variance(x))
> variance(x)
[1] 14.40816
> stdev(x)
[1] 3.795809
> var(x) # porovnejte vysledky
[1] 16.80952
> sd(x)
[1] 4.099942
```

# Variační koeficient

Variační koeficient je statistická míra relativního rozptylu datových bodů ve vztahu k průměru.

Variační koeficient $CV$ je definován jako poměr směrodatné odchylky $s$ k průměru $\overline{x}$.

$$CV = \frac{s}{\overline{x}}. \tag{17}$$

Variační koeficient se často vyjadřuje v procentech.

# Variační koeficient

Variační koeficient není v jazyce R implementován jako funkcie

Lze ho určit pomocí známých funkcí nebo definovat funkci vlastní

```
1 > cv<-function(x) variance(x)/mean(x) * 100
2 > cv(x)
3 [1] 157.5893
```

# Šikmost

Šikmost je mírou asymetrie rozdělení nebo souboru dat.

Šikmost $\gamma_1$ definujeme jako

$$\gamma_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^3}{s^3}. \tag{18}$$

# Šikmost

V závislosti na hodnotě $\gamma_1$ existují 3 typy šikmosti.

Pokud $\gamma_1 > 0$, hovoříme o pravostranné šikmosti. To znamená, že většina dat je menší než průměr.

Na druhé pokud $\gamma_1 < 0$, hovoříme o levostranné šikmosti. V tomto případě je většina hodnot v souboru dat větší než průměr.

A konečně nulová šikmost $\gamma_1 = 0$ představuje symetrické rozdělení dat.

# Špicatost

Špičatost měří ostrost vrcholu v rozdělení dat.

Špičatost $\gamma_2$ definujeme jako

$$\gamma_2 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^4}{s^4}. \tag{19}$$

# Špicatost

Podobně jako v případě šikmosti i zde existují tři typy špičatosti.

Pokud je $\gamma_2 = 3$, hovoříme o normální špičatosti rozdělení. Hodnota $\gamma_2$ je srovnávána s hodnotou 3, protože špičatost normálního rozdělení je rovna 3. Porovnáváme špičatost vzorku s Gaussovou křivkou.

V případě $\gamma_2 < 3$ je rozdělení plošší než normální rozdělení .

V opačném případě, kdy $\gamma_2 > 3$, má analyzované rozdělení špičatější vrchol než normální rozdělení.

# Statistika a programování v R

## VII. Odhady parametrů

# Odhady parametrů

Jedním z cílů statistické analýzy je odhadnout parametry původního rozdělení, ze kterého pochází náhodný výběr.

Rozlišujeme dva typy odhadů:

- bodové odhady, které poskytují odhad přesných hodnot parametrů,
- intervaly spolehlivosti, které představují intervaly, které obsahují hodnotu parametru s danou pravděpodobností (uvádí se jako hladina spolehlivosti).

# Bodové odhady

Pro odhad hodnoty parametru se snažíme vybrat takovou charakteristiku, která nejlépe aproximuje parametr $\Theta$.

Kvalitu odhadu určují jeho vlastnosti:

- Nevychýlený odhad parametru $\theta$ je takový odhad $T$, že platí rovnost $\mathbb{E}(T) = \theta$.
- Konzistentní odhad může být charakterizován rostoucí přesností se zvyšující se velikostí vzorku. Formálně můžeme psát $\lim_{n\to\infty} T_n = \Theta$, kde indexy představují velikost vzorku použitého při odhadu.
- Efektivní odhad lze interpretovat jako nejlepší možný odhad. Nepřesnost se měří střední kvadratickou chybou $T$, tzn. $MSE(T) = \mathbb{E}\left((T - \Theta)^2\right)$. Efektivní odhad tuto hodnotu minimalizuje.

# Bodové odhady

Tyto vlastnosti ovlivňují provádění charakteristik vzorků v jazyce R.

Podívejme se na průměr vzorku. Pokud máme náhodný vzorek $X_1, \ldots, X_n$ z rozdělení se střední hodnotou $\mu$, můžeme vypočítat:

$$\mathbb{E}(\overline{x}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}(X_i) = \frac{1}{n} n\mu = \mu.$$

Takže jsme ukázali, že průměr je nevychýlený odhad průměru vzorku.

# Bodové odhady

Nyní se podívejme na výběrový rozptyl. Použijeme nerovnosti

$$\sum_{i=1}^{n}(X_i - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu + \mu - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu)^2 - n(\overline{X} - \mu)^2,$$

dostaneme

$$\begin{aligned}\mathbb{E}\left(\sum_{i=1}^{n}(X_i - \overline{X})^2\right) &= \mathbb{E}\left(\sum_{i=1}^{n}(X_i - \mu)^2 - n(\overline{X} - \mu)^2\right) \\ &= \sum_{i=1}^{n}\mathbb{D}\left(\overline{X_i}\right) - n\mathbb{D}\left(\overline{X}\right) \\ &= n\sigma^2 - \frac{n\sigma^2}{n} = (n-1)\sigma^2,\end{aligned}$$

# Bodové odhady

Nyní se podívejme na výběrový rozptyl. Použijeme nerovnosti

$$\sum_{i=1}^{n}(X_i - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu + \mu - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu)^2 - n(\overline{X} - \mu)^2,$$

a proto

$$\mathbb{E}\left(s^2\right) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2\right) = \frac{n-1}{n}\sigma^2.$$

Vidíme tedy, že výběrový rozptyl není nevychýlený odhad náhodného rozptylu.

# Bodové odhady

Nyní se podívejme na výběrový rozptyl. Použijeme nerovnosti

$$\sum_{i=1}^{n}(X_i - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu + \mu - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu)^2 - n(\overline{X} - \mu)^2,$$

Abychom získali nevychýlený odhad, musíme vynásobit výběrový rozptyl $\frac{n}{n-1}$. Získáme tak nevychýlený odhad rozptylu

$$s_{n-1}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

Tento výsledek vysvětluje, proč je funkce `var()` implementována odlišně než výběrový rozptyl. Představuje nevychýlený odhad původní náhodné proměnné.

# Bodové odhady

## Metody

V tomto kurzu představíme dvě metody konstrukce bodových odhadů:

- metoda momentů,
- metoda maximální pravděpodobnosti.

Předpokládejme, že máme vzorek $X_1, \ldots, X_n$ z rozdělení, který závisí na vektoru parametrů $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$.

# Metoda momentů

Předpokládejme dále, že existují všechny momenty $\nu_k = \mathbb{E}\left(X_i^k\right)$, $k = 1, \ldots, m$.

Momenty vzorku $v_k$ jsou definovány jako $v_k = \frac{1}{n}\sum_{i=1}^{n} X_i^k$ pro $k = 1, \ldots, m$.

Principem metody momentů je rovnost teoretického momentu a momentu vzorku.

# Metoda momentů

Znamená to odhadnout momenty pro $\theta_1, \theta_2, \ldots, \theta_k$, označení $\widehat{\theta}_1, \widehat{\theta}_2, \ldots, \widehat{\theta}_k$, které jsou definovány jako řešení (pokud existuje) rovnic:

$$\nu_k = v_k, \quad k = 1, \ldots, m.$$

Alternativně namísto libovolného $r$-tého momentu můžeme použít $r$-tý centrální moment definovaný jako $\mu_r = \mathbb{E}\left((X - \mathbb{E}(X))^r\right)$ a $r$-tý centrální moment vzorku $m_r = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^r$.

# Metoda momentů

Metodu ilustrujeme na případě rovnoměrného rozdělení s parametry $a$ a $b$. Je známo, že momenty rovnoměrně rozložené náhodné proměnné $X$ jsou

$$\mathbb{E}(X) = \frac{a+b}{2} \quad \text{a} \quad \mathbb{D}(X) = \frac{(b-a)^2}{12}.$$

Určit odhady parametrů $a$ a $b$ představuje vyřešit rovnice:

$$\overline{x} = \frac{a+b}{2},$$
$$s^2 = \frac{(b-a)^2}{12}.$$

# Metoda momentů

Řešením těchto rovnic dostaneme:

$$a = \overline{x} - \sqrt{3}s,$$
$$b = \overline{x} + \sqrt{3}s.$$

Nyní jsme připraveni implementovat tyto odhady v jazyce R. Použijeme předem definované funkce `variation()` resp. `stdev()`.

# Metoda momentů

```
1 > x<-runif(1000,1,3) #generating the random sample
2 > a<-mean(x)-sqrt(3)*stdev(x)
3 > b<-mean(x)+sqrt(3)*stdev(x)
4 > a
5 [1] 1.018323 #can differ for other samples
6 > b
7 [1] 3.033395 #can differ for other samples
```

# Metoda maximální pravděpodobnosti

Tato metoda je založena na maximalizaci pravděpodobnostní funkce.

Bod v prostoru parametrů, který maximalizuje funkci pravděpodobnosti, se považuje za odhad maximální pravděpodobnosti.

Formálně předpokládejme, že $X_1, \ldots, X_n$ jsou nezávislé náhodné proměnné se stejným rozdělením s hustotou $f(x, \boldsymbol{\theta})$. Sjednocená hustota je součinem těchto jednorozměrných funkcí hustoty:

$$L(x_1, \ldots, x_n; \boldsymbol{\theta}) = \prod_{i=1}^{n} f(x, \boldsymbol{\theta}).$$

# Metoda maximální pravděpodobnosti

Právě představená funkce $L(x_1, \ldots, x_n; \boldsymbol{\theta})$ se nazývá pravděpodobná funkce.

Abychom získali odhady parametrů, maximalizujeme tuto funkci standardním procesem známým z matematické analýzy.

Pro usnadnění práce maximalizujeme místo funkce $L(x_1, \ldots, x_n; \boldsymbol{\theta})$ její logaritmus $\ln L(x_1, \ldots, x_n; \boldsymbol{\theta})$ .

Přirozený logaritmus je rostoucí funkce, proto má extrémy a navíc převádí součin na součet funkcí.

# Metoda maximální pravděpodobnosti

Metodu ilustrujeme na problému odhadu pravděpodobnosti $p$ nějaké náhodné události $A$.

Tuto situaci můžeme interpretovat jako výsledek alternativní náhodné proměnné, která je indikátorem náhodné události $A$.

Potom máme $\mathbb{P}(X = 1) = p$ a $\mathbb{P}(X = 0) = 1 - p$.

Provádíme $n$ náhodných pokusů a pozorujeme výskyt události $A$.

Takže dostaneme vzorek $X_1, \ldots, X_n$ náhodných proměnných s hustotami $f(x_i, p) = p^{x_i}(1-p)^{1-x_i}$, kde $x_i \in \{0, 1\}$.

# Metoda maximální pravděpodobnosti

Odpovídající pravděpodobnostní funkce má tvar

$$L(x,p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^{\sum_{i=1}^{n} x_i}(1-p)^{n-\sum_{i=1}^{n} x_i}.$$

Abychom mohli určit odhad $p$, maximalizujeme funkci $L(x,p)$ vzhledem k parametru $p$. Abychom toho dosáhli, používáme přirozený logaritmus pravděpodobnostní funkce

$$\ln(L(x,p)) = \sum_{i=1}^{n} x_i \ln p + \left(n - \sum_{i=1}^{n} x_i\right) \ln(1-p),$$

# Metoda maximální pravděpodobnosti

Jeho derivaci vzhledem k $p$ položíme rovnou 0:

$$\frac{\mathrm{d} \ln L(x,p)}{\mathrm{d}x} = \frac{\sum_{i=1}^{n} x_i}{p} - \frac{n - \sum_{i=1}^{n} x_i}{1-p} = 0.$$

Vynásobením rovnice $\frac{1}{n}$ dostaneme

$$\overline{x} \cdot \frac{1}{p} - (1 - \overline{x}) \cdot \frac{1}{1-p} = 0,$$

a řešením je

$$p = \overline{x}.$$

# Metoda maximální pravděpodobnosti

Abychom potvrdili, že $p = \overline{x}$ skutečně maximalizuje funkci pravděpodobnosti, musíme ověřit derivaci druhého řádu.

$$\frac{\mathrm{d}^2 \ln L(x,p)}{\mathrm{d}x^2} = -\frac{\overline{x}}{p} + (1-\overline{x}) \cdot \frac{1}{(1-p)^2}.$$

Nahrazením $p = \overline{x}$ dostaneme

$$\left(\frac{\mathrm{d}^2 \ln L(x,p)}{\mathrm{d}x^2}\right)_{p=\overline{x}} = -\frac{1}{1-\overline{x}} < 0.$$

Potom máme nejpravděpodobnější odhad $p = \overline{x}$.

# Metoda maximální pravděpodobnosti

Tento přístup můžeme použít k určení toho, do jaké míry je házení mincí spravedlivé.

Nejprve vygenerujeme vzorek 0 a 1, což znamená hod rub nebo líc. Abychom získali vzorek neférové mince, deklarujeme vektor pravděpodobností `prob`, jak můžeme vidět na zdrojovém kódu:

```
1 > x<-sample(c(0,1),1000,replace=TRUE,prob=c(2/3,1/3))
2 > mean(x)
3 [1] 0.304
```

# Intervaly spolehlivosti

Interval spolehlivosti může být definován jako rozsah odhadů pro neznámý parametr, který obsahuje skuteční hodnotu parametru s danou pravděpodobností.

Tato pravděpodobnost, že se parametr nachází v daném intervalu, se uvádí jako hladina spolehlivosti.

Nejběžnější hladina spolehlivosti používaná v praxi je 95 %, ale často se používají i jiné úrovně (například 90 % nebo 99 %).

# Intervaly spolehlivosti

Formálně ať $\mathbf{X} = (X_1, \ldots, X_n)$ je náhodný vzorek s rozdělením, které závisí na neznámém parametru $\theta$.

Interval spolehlivosti pro parametr $\theta$ s hladinou spolehlivosti $\alpha$ je interval s náhodnými koncovými body $(u(\mathbf{X}); v(\mathbf{X}))$ určený dvojicí náhodných proměnných $u(\mathbf{X})$ a $v(\mathbf{X})$ s vlastností:

$$\mathbb{P}\left(u(\mathbf{X}) < \theta < v(\mathbf{X})\right) = \alpha.$$

# Intervaly spolehlivosti

Odvození intervalu spolehlivosti ilustrujeme na normálním rozdělení.

Předpokládejme nejprve, že $X_1, \ldots, X_n$ je náhodný vzorek z normálního rozdělení $N(\mu, \sigma^2)$, jehož standardní odchylka $\sigma$ je známa.

Chceme najít interval spolehlivosti pro střední hodnotu $\mu$. Protože $\overline{X}$ má normální rozdělení $N\left(\mu, \frac{\sigma^2}{n}\right)$, máme

$$\mathbb{P}\left(\left|\frac{\overline{X} - \mu}{\frac{\sigma}{\sqrt{n}}}\right| < c\right) = 2\Phi(c) - 1, \text{ pro všechny } c > 0,$$

# Intervaly spolehlivosti

Je to ekvivalentní

$$\mathbb{P}\left(\overline{X} - c\frac{\sigma}{\sqrt{n}} < \mu < \overline{X} + c\frac{\sigma}{\sqrt{n}}\right) = 2\Phi(c) - 1, \text{ pro všechny } c > 0.$$

Z posledního vztahu vyplývá, že interval spolehlivosti pro průměr s hladinou spolehlivosti $\alpha = 2\Phi(c) - 1$ má tvar

$$\left(\overline{X} - c\frac{\sigma}{\sqrt{n}}; \overline{X} + c\frac{\sigma}{\sqrt{n}}\right).$$

Vezmeme-li v úvahu, že $c = \Phi^{-1}\left(\frac{\alpha+1}{2}\right)$, můžeme interval spolehlivosti zapsat ve tvaru

$$\left(\overline{X} - \Phi^{-1}\left(\frac{\alpha+1}{2}\right)\frac{\sigma}{\sqrt{n}}; \overline{X} + \Phi^{-1}\left(\frac{\alpha+1}{2}\right)\frac{\sigma}{\sqrt{n}}\right).$$

# Intervaly spolehlivosti

Ke zjištění hranic intervalu spolehlivosti používáme kvantilovou funkci `qnorm()`.

```
x<-rnorm(100,1,2) # we generate some sample
> n<-length(x)
> sample.mean<-mean(x)
> sample.sd<-2 #  standard deviation is known
> alpha<-0.95 # seting the confidence level 95\%
> c<-qnorm((alpha+1)/2,0,1)
> margin<-c*sample.sd/sqrt(n)
> lower.bound<-sample.mean-margin
> upper.bound<-sample.mean+margin
> print(c(lower.bound,upper.bound))
[1] 0.8149884 1.5989740
```

# Intervaly spolehlivosti

Nyní se podívejme, jak se změní interval spolehlivosti, pokud standardní odchylka není známa.

Pak musíme použít nevychýlený odhad standardní odchylky $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\overline{X} - X_i)^2$, $s = \sqrt{s^2}$. Potom náhodná proměnná

$$T = \frac{\overline{X} - \mu}{s}\sqrt{n},$$

má Studentovo $t$-rozdělení s $n - 1$ stupni volnosti. Potom dostaneme uvedený interval spolehlivosti

$$\left(\overline{X} - c\frac{s}{\sqrt{n}}; \overline{X} + c\frac{s}{\sqrt{n}}\right).$$

# Intervaly spolehlivosti

Hodnota $c$ je odpovídající kvantil Studentova rozdělení, proto v R použijeme funkci `qt()`.

```
> n<-length(x) # we use previously generated sample
> sample.mean<-mean(x)
> sample.sd<-sd(x) # estimate of the standard deviation
> alpha<-0.95
> c<-qt((alpha+1)/2,df=n-1)
> margin<-c*sample.sd/sqrt(n)
> lower.bound<-sample.mean-margin
> upper.bound<-sample.mean+margin
> print(c(lower.bound,upper.bound))
[1] 0.8118763 1.6020861
```

# Intervaly spolehlivosti

Nyní se podívejme na interval spolehlivosti pro rozptyl normálního rozdělení.

Pokud $s^2$ je nevychýlený odhad rozptylu, pak náhodná proměnná

$$Y = \frac{(n-1)s^2}{\sigma^2} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{\sigma^2} = \sum_{i=1}^{n}\left(\frac{x_i - \overline{x}}{\sigma}\right)^2$$

se řídí rozdělením $\chi^2(n-1)$.

# Intervaly spolehlivosti

Pro interval spolehlivosti dostaneme

$$\mathbb{P}(c_1 < \chi^2 < c_2) = \alpha$$
$$\mathbb{P}\left(c_1 < \frac{(n-1)s^2}{\sigma^2} < c_2\right) = \alpha$$
$$\mathbb{P}\left(\sigma \in \left(\frac{(n-1)s^2}{c_2}; \frac{(n-1)s^2}{c_1}\right)\right) = \alpha,$$

kde $c_1$ a $c_2$ jsou kritické hodnoty rozdělení $\chi^2(n-1)$.

# Intervaly spolehlivosti

Při výpočtu musíme respektovat, že rozdělení $\chi^2$ není symetrické, což je důležité pro kritické hodnoty $c_1$ a $c_2$.

```
1  > n<-length(x) # the sample already generated sooner
2  > sample.var<-var(x)
3  > c1<-qchisq(1-(alpha+1)/2,df=n-1)#consequent of asymmetry
4  > c2<-qchisq((alpha+1)/2,df=n-1)#consequent of asymmetry
5  > lower.bound<-sample.var*(n-1)/c2
6  > upper.bound<-sample.var*(n-1)/c1
7  > print(c(lower.bound,upper.bound))
8  [1] 3.056626 5.350767
```

# Intervaly spolehlivosti

V posledním příkladu si ukážeme interval spolehlivosti pro pravděpodobnost náhodné události.

Nevychýlený odhad pravděpodobnosti $p$ výskytu náhodné události je $\hat{p} = \frac{m}{n}$, kde $m$ je počet výskytů pozorované události v řadě $n$ náhodných pokusů.

Víme, že $\mathbb{E}(\hat{p}) = p$ a $\mathbb{D}(\hat{p}) = \frac{pq}{n}$, kde $q = 1 - p$.

Pro velké $n$ platí přibližná rovnost $\frac{pq}{n} \approx \frac{\hat{p}\hat{q}}{n}$.

# Intervaly spolehlivosti

Potom náhodná proměnná

$$Z = \frac{\frac{m}{n} - p}{\sqrt{\frac{\hat{p}\hat{q}}{n}}}$$

se řídí standardizovaným normálním rozdělením $N(0, 1)$. Potom interval spolehlivosti pro pravděpodobnost $p$ s hladinou spolehlivosti $\alpha$ můžeme zapsat jako

$$\left( \hat{p} - \Phi^{-1}\left(\frac{\alpha+1}{2}\right) \cdot \sqrt{\frac{\hat{p}\hat{q}}{n}}; \hat{p} + \Phi^{-1}\left(\frac{\alpha+1}{2}\right) \cdot \sqrt{\frac{\hat{p}\hat{q}}{n}} \right).$$

# Intervaly spolehlivosti

**Example**

Předpokládejme, že se uskutečnil průzkum na 250 náhodně vybraných lidech, aby se zjistilo zda vlastní tablet. Z 250 dotázaných 98 uvedlo, že vlastní tablet. Pomocí 95 % hladiny spolehlivosti vypočtěte odhad intervalu spolehlivosti pro skutečný podíl lidí, kteří vlastní tablet.

# Děkuji za pozornost.


Statistika a programování v R

Aleš Kozubík