Moderní přístup k aplikaci matematických dovedností v přírodovědných a ekonomických oborech
reg. č.: CZ.1.07/2.2.00/28.0168

Základy analýzy dat v softwaru *Mathematica*

Ondřej Vencálek

Olomouc 2015

Oponenti: RNDr. Tomáš Fürst, Ph.D.
RNDr. Bohumír Procházka, CSc.

Tato publikace neprošla redakční jazykovou úpravou.

Olomouc 2015
1. vydání

ISBN 978-80-244-4474-1

Obsah

1. Práce s daty 6

1.1 Jak vypadají data 6
1.2 Práce s vektory 9
1.3 Práce s maticemi 14
1.4 Řekněte si Mathematice o data! Třeba o rybách 18
1.5 Řekněte si o data z internetu! Třeba o taxonomii 25

2. Popis rozdělení jednoho znaku 27

2.1 Kvalitativní znak 27
2.2 Kvantitativní znak 33
2.3 ... má však cenné údaje 48

3. Popis vztahu dvou a více znaků 49

3.1 Dva kvalitativní znaky 49
3.2 Kvalitativní a kvantitativní znak 54
3.3 Dva kvantitativní znaky 59
3.4 Popis vztahu vícero znaků 64
3.5 Dodatek 68

4. Závěr 70

5. Použitá literatura 72

Motto:

Tak tedy o tomhle v knihách žádná řeč nebyla ...
(They didn't say anything about this in the books ...)

James Herriot

Základním kurzem statistiky dnes projde značná část (zřejmě můžeme říci většina) studentů přírodovědných, technických, ekonomických, medicínských i jiných oborů na vysokých školách. Vyučující těchto základních kurzů si kladou různé cíle.

Statistika pro nematematiky

V kurzech pro „nematematiky" si vyučující mnohdy vystačí s tím, že studenty chtějí zbavit „strachu z matematiky" a ukázat jim, že statistika „může být k něčemu dobrá". A to věru není jednoduchý úkol. Člověk, který není běžně v situaci, kdy potřebuje analyzovat číselné údaje, brzy zapomene, co že je to vlastně ten „t-test", o kterém slyšel ve škole. To samozřejmě vůbec nevadí. Není smyslem výuky studenta přinutit, aby si tohle jednou provždy zapamatoval. Základní kurz statistiky však může trvale změnit přístup studenta k informacím. Právě takovýto kurz by totiž měl vést k poznání, že statistika (ostatně jako mnoho jiných "nástrojů"), může být dobrým pomocníkem, ale stejně tak dobře může být zneužita k manipulaci. Nejen vysokoškolsky vzdělaní lidé jsou dnes prakticky denně vystaveni toku informací, z nichž mnohé mají (alespoň zčásti) číselnou podobu. Úspěšný absolvent základního kurzu statistiky si klade otázky typu *Co vlastně znamená číslo, kterým se mě někdo snaží přesvědčit o „své pravdě"? Jaké jsou nejistoty v závěrech výzkumu, který mám před sebou?*

Statistika pro matematiky

Základní kurzy statistiky pro matematiky se od těch pro nematematiky liší v tom, že jsou daleko více zaměřeny na „technické podrobnosti" – jsou plné matematických definic, vět a důkazů. Studenti bývají tím vším poněkud zahlceni a nezřídka jim pak unikají ideje definic a praktický smysl uváděných vět. Kvalitní kurzy pro nematematiky jsou plné detailně komentovaných příkladů. Od studenta matematiky jako by se občas očekávalo, že příklady ani nepotřebuje. Když je pak postaven do situace, kdy má zpracovat konkrétní datovou sadu, nezbude mu, než si povzdechnout „tak tedy o tomhle v knihách žádná řeč nebyla".

Cíl autora

Cíl autora této publikace je skromný: poskytnout studentům i absolventům základního kurzu statistiky materiál obsahující komentované zpracování konkrétní datové sady. Takovéto zpracování je dnes prakticky nemyslitelné bez použití počítače a specializovaného softwaru pro analýzu dat. Takovýchto softwarových produktů je celá řada. Software *Mathematica*, se kterým budeme pracovat, je jen jednou z možností při výběru prostředí pro analýzu dat. Tato publikace však rozhodně není „manuálem" k softwaru *Mathematica*.

Příklady, kam se jen podíváš (?)

Sběr dat určených k analýze je dosti často časově i finančně náročná záležitost. Není proto běžné shromážděná data veřejně bezplatně poskytovat. Existuje však několik webových stránek obsahujících (veřejně přístupné) datové sady určené dnes převážně pro potřeby výuky. Uveďme zde tři takovéto stránky:

- http://www.umass.edu/statdata/statdata/index.html (University of Massachusetts Amherst)
- http://lib.stat.cmu.edu/DASL/allmethods.html (systém StatLib, Carnegie Mellon University)
- http://www.amstat.org/publications/jse/jse_data_archive.htm (Journal of Statistics Education)

Na výše uvedených stránkách lze najít velké množství datových sad, na kterých si zájemci mohou procvičovat a zdokonalovat své praktické schopnosti analýzy dat.

Na co se může čtenář těšit

Pojednání o základech analýzy dat je rozděleno do tří kapitol. V kapitole 1 je pojednáno o práci s daty. Nejprve si ukážeme, co vše je třeba se naučit, než vůbec můžeme s analýzou dat

začít. Jde především o práci s číselnými vektory a s maticemi. Pokročilejší čtenář může odhalovat možnosti softwaru *Mathematica* v kapitolkách 1.4 a 1.5, čtenář-začátečník může tyto dvě kapitolky vynechat. Kapitola 2 je věnována možnostem popisu jednoho statistického znaku. Jejím základem jsou části pojednávající o znacích kvalitativních a kvantitativních. Kapitolka 2.3 je spíše jen poznámkou na okraj. Kapitola 3 pak pojednává o popisu vztahu dvou znaků. Její závěrečná část pak obsahuje krátkou zmínku o (zejména grafických) možnostech při analýze vztahu více než dvou znaků.

Jak s textem pracovat

Čtení tohoto textu by mělo přímo čtenáře vybízet k tomu, aby si jednotlivé postupy sám vyzkoušel. Čtenář by přitom měl aktivně pracovat s nápovědou (helpem), neboť ne vždy jsou jednotlivé příkazy detailně komentovány a tak je třeba některé detaily dohledat právě v nápovědě k jednotlivým příkazům. Pokud se s *Mathematicou* setkáváte poprvé, pak věžte, že základní nápovědu je možno získat například prostřednictvím hlavního menu, které pod heslem Help nabízí Documentation Center – tématicky uspořádanou nápovědu s možností vyhledávat různá hesla. Nápovědu ke konkrétnímu příkazu lze získat tak, že název příkazu napíšete, kliknete na něj pravým tlačítkem a zvolíte možnost Get Help. *Mathematica* v současnosti nemá česky psanou nápovědu.

1 Práce s daty

K základním dovednostem nutným pro zpracování dat patří schopnost načíst data z databáze (například v podobě tabulky v Excelu) do softwaru pro výpočty (v našem případě softwaru *Mathematica*). V této kapitole si osvojíme právě tuto dovednost, stejně jako dovednost s daty dále pracovat. Navíc se dozvíme, že *Mathematica* poskytuje přístup k některým zajímavým datovým souborům. Zmíníme také možnost načíst data z internetu.

1.1 Jak vypadají data

Jak vypadají typická data? Tato otázka je samozřejmě dosti široká. Data mohou mít různý původ – může jít například o záznamy fyzikálních měření, o záznamy vyšetření pacientů, o údaje získané na základě průzkumu veřejného mínění, nebo o záznamy vývoje hodnot nějaké ekonomické veličiny (jako je např. HDP) v čase. Tento výčet ani zdaleka není kompletní a přitom již z něj je zřejmé, že statistik se může setkávat s daty napříč prakticky všemi oblastmi lidského vědění – od fyziky, přes medicínu až např. k sociologii či ekonomii. Diskuse o vhodnosti použití kvantitativních metod (tedy hrubě řečeno metod popisu reality za pomocí číselných údajů) v různých oblastech je přitom stále otevřená. Zatímco například fyzika je „bez čísel“ prakticky nemyslitelná, v jiných vědních (a nevědních) disciplínách můžeme o roli čísel a jejich využití s úspěchem pochybovat.

Přes značnou různorodost oborů, kterých se mohou analyzovaná data týkat, je jejich struktura většinou dosti podobná. Představme si nějaký objekt našeho zájmu – například jednoho konkrétního člověka, jeden konkrétní stát, jeden konkrétní vzorek oceli nebo jedno konkrétní auto. U tohoto objektu nás zajímá nějaká jeho vlastnost. Předpokládejme, že tuto vlastnost můžeme vyjádřit číslem. Např. výška člověka (v centimetrech), rozloha státu (v km^2), jednoosá pevnost v tlaku (v MPa), stáří auta (v letech). Když na jednom objektu našeho zájmu sledujeme vícero vlastností (veličin), pak záznamy o těchto vlastnostech tvoří **vektor**. Například u jednoho člověka změříme výšku, váhu, systolický a diastolický tlak. Vektor však také získáme, budeme-li jednu vlastnost zjišťovat na vícero objektech našeho zájmu (změříme výšku dvaceti různých lidí). Máme-li vícero objektů zájmu a u každého sledujeme vícero vlastností, jeden vektor už nám nestačí – máme vícero (stejně dlouhých) vektorů a ty uspořádáme „do tabulky“ – **matice**. Tradiční uspořádání matice je takové, že jednotlivé řádky matice odpovídají záznamům o jednotlivých objektech našeho zájmu, zatímco jednotlivé sloupce odpovídají jednotlivým sledovaným veličinám. Poznamenejme, že statistici mají vlastní terminologii, a tak místo „objekt zájmu“ říkají „statistická jednotka“ a místo „sledovaná veličina“ někdy říkají „statistický znak“.

Příklad – rybářská data

Podívejme se nyní na jedna konkrétní data. Jde o záznamy o 159 ulovených rybách, z nichž u každé bylo sledováno 8 vlastností – druh (podle biologické klasifikace), hmotnost (v gramech), délka těla od špičky hlavy po konec ošupení (v cm), délka těla od špičky hlavy po zářez ocasní ploutve (v cm), celková délka těla od špičky hlavy po konec ocasní ploutve (v cm), „výška“ ryby (udává se jako procento celkové délky těla), „šířka“ („tloušťka“) ryby (udává se jako procento celkové délky těla), pohlaví ryby. Ilustrativní obrázek k jednotlivým rozměrům je společně s detailním popisem dat (v angličtině) uveden na stránce

`http://www.amstat.org/publications/jse/datasets/fishcatch.txt`.

Samotná data pak najdeme na stránce

`http://www.amstat.org/publications/jse/datasets/fishcatch.dat.txt`.

(Tyto odkazy jsou platné k červenci 2013. Pokud by v budoucnu došlo ke změně adresy stránek s těmito daty, bylo by zřejmě nutné data nalézt pomocí některého z běžných vyhledávačů (jako je např. google) pomocí klíčových slov „fishcatch data Laengelmavesi“).

Než se pustíme do práce s daty, jejímž prvním krokem bude načtení dat, bylo by možná dobré dozvědět se, jakých druhů ryb se data týkají. V souboru fishcatch.txt jsou pouze anglické a latinské názvy (názvy ve finštině či švédštině asi většině čtenářů nebudou příliš platné). Ve čtvrté části této kapitoly si ukážeme, jak široké je uplatnění programu *Mathematica* – nejen, že umí „počítat“, ale zná také ryby! Nechť tedy zvídavý čtenář nalistuje kapitolku 1.4, která mu nabídne pohled do němých tváří těch, o kterých bude v následujícím textu řeč.

Data načteme pomocí příkazu **`Import[]`**. Pokud se se softwarem *Mathematica* setkáváte poprvé, pak věřte, že se všechny příkazy v tomto softwaru píší „s velkým písmenem na začátku“. *Mathematica* je programovacím jazykem, kde význam jednotlivých „slov“ závisí na tom, jak jsou napsány, zda velkými či malými písmeny (v anglické terminologii se tato vlastnost nazývá „case sensitivity“). Argument příkazu je vždy uveden v hranatých závorkách.

Z důvodu úspory místa zde data nebudeme přetiskovat.

In[1]:=
```
ryby =
  Import["http://www.amstat.org/publications/jse/datasets/fishcatch.dat.txt",
   "Data"];
```

Pro zobrazení dat na obrazovku počítače lze použít příkaz

```
ryby // TableForm
```

Datová matice má 159 řádků a 9 sloupců. Jeden řádek představuje údaje o jedné chycené rybě. První sloupec je „identifikátor ryby“ – každá ryba má své číslo. První má jedničku, druhá dvojku, ..., poslední má číslo 159. Dalších osm sloupců pak představuje postupně všech osm sledovaných veličin.

Zapamatujme si, že

- Data mají obvykle podobu datové matice,
- jejíž řádky představují jednotlivé objekty našeho zájmu
- a jejíž sloupce představují jednotlivé sledované veličiny.

Řádková data – ryba číslo 1

Jak již bylo řečeno, data v jednotlivých řádcích představují záznamy o jednotlivých objektech našeho zájmu. Zobrazme nyní (všechny) údaje o „rybě číslo 1“:

In[2]:=
```
ryby[[1, All]]
```

Out[2]=
```
{1, 1, 242., 23.2, 25.4, 30., 38.4, 13.4, NA}
```

Jak bylo uvedeno výše, první sledovaná veličina je druh ryby. Druh ryby zřejmě není číselná informace. Co tedy znamená druh 1? Potřebujeme vysvětlení, jakým způsobem je slovní informace převedena na číslo. Toto vysvětlení je obsaženo v souboru fishcatch.txt v odstavci „VARIABLE

DESCRIPTIONS“. Zde se dozvíme, že druh číslo 1 je latinsky „abramis brama“, tedy česky „cejn velký“. Dále vidíme, že ryba číslo 1 vážila 242 gramů, měřila 23,2 cm od špičky hlavy po konec ošupení, 25,4 cm od špičky hlavy po zářez ocasní ploutve, celková délka jejího těla byla 30,0 cm. „Výška“ ryby byla 38,4 % celkové délky jejího těla, zatímco „šířka“ byla 13,4 % celkové délky těla. Kolik to však je v centimetrech? To zjistíme jednoduchým výpočtem 30 · (38,4 / 100), respektive 30 · (13,4 / 100) :

In[3]:=
```
ryby[[1, 6]] * ryby[[1, 7]] / 100
```

Out[3]=
```
11.52
```

In[4]:=
```
ryby[[1, 6]] * ryby[[1, 8]] / 100
```

Out[4]=
```
4.02
```

Ryba číslo 1 byla tedy 11,52 cm vysoká a 4,02 cm široká. Poznamenejme, že vzhledem k tomu, že údaje o délce ryb jsou uváděny s přesností na desetiny centimetru, nemají zde setiny centimetru zřejmě valný význam. Můžeme tedy říci, že ryba byla přibližně 11,5 cm vysoká a 4,0 cm široká.

Situace, kdy jednu veličinu potřebujeme nějakým způsobem upravit, například převést na jiné jednotky, je poměrně častá. Samotná úprava není nic složitého. Potřebujeme se však naučit, jak tuto úpravu udělat pro všechny řádky matice (pro všechny ryby) najednou. Tedy, potřebujeme vědět, jak z jednoho či dvou sloupců matice vytvořit sloupec nový.

Poslední údaj je údaj o pohlaví ryby. V souboru fishcatch.txt zjistíme, že 1 = male (sameček), 0 = female (samička). Zde je však uvedena hodnota NA. Jde o symbol reprezentující chybějící údaj. Tedy, u ryby číslo 1 není známo, zda jde o samečka či samičku. Symbol NA pro chybějící hodnoty se v různých obměnách (například n/a či N/A) vyskytuje běžně. Jde o zkratku z anglických slov „not available“ (případně not applicable, no answer).

Naše poznatky z prvního setkání s daty shrňme takto:

- První sloupec bývá často identifikátorem objektu našeho zájmu (např. číslo ryby, číslo měření, rodné číslo).
- Slovní informace může být převedena na číselnou, potřebujeme však informaci o tom, jakým způsobem je slovní popis převeden na čísla.
- Symbol NA znamená chybějící hodnotu.

Sloupcová data – celková délka těla

Statistická analýza se většinou více než o jednotlivé řádky zajímá o jednotlivé sloupce. Typické otázky, které se snažíme zodpovědět mají následující podobu:

- Jaký je vztah mezi váhou ryby a celkovou délkou jejího těla?
- Je tento vztah stejný pro všechny námi sledované druhy ryb?
- Je cejn velký (druh číslo 1) větší (ve smyslu celkové délky těla) než cejnek malý (druh číslo 4)?
- Jaká je typická celková délka ryb jednotlivých druhů?

Jde o otázky obecné. Nezajímá nás konkrétní ryba (konkrétní řádek), ale svou zkušenost, získanou na základě měření 159 ryb, se za pomocí statistických metod snažíme zobecnit.

Podívejme se tedy nyní na údaje o celkové délce těla všech 159 ryb. Tyto údaje jsou ve sloupci číslo 6.

In[5]:=
```
ryby[[All, 6]]
```

Out[5]=
```
{30., 31.2, 31.1, 33.5, 34., 34.7, 34.5, 35., 35.1, 36.2, 36.2, 36.2, 36.4, 37.3,
 37.2, 37.2, 38.3, 38.5, 38.6, 38.7, 39.5, 39.2, 39.7, 40.6, 40.5, 40.9, 40.6,
 41.5, 41.6, 42.6, 44.1, 44., 45.3, 45.9, 46.5, 28.7, 29.3, 30.8, 34., 39.6,
 43.5, 16.2, 20.3, 21.2, 22.2, 22.2, 22.8, 23.1, 23.7, 24.7, 24.3, 25.3, 25.,
 25., 27.2, 26.7, 26.8, 27.9, 29.2, 30.6, 35., 16.5, 17.4, 19.8, 21.3, 22.4,
 23.2, 23.2, 24.1, 25.8, 28., 29., 10.8, 11.6, 11.6, 12., 12.4, 12.6, 13.1,
 13.1, 13.2, 13.4, 13.5, 13.8, 15.2, 16.2, 34.8, 37.8, 38.8, 39.8, 40.5, 41.,
 45.5, 45.5, 45.8, 48., 48.7, 51.2, 55.1, 59.7, 64., 64., 68., 8.8, 14.7, 16.,
 17.2, 18.5, 19.2, 19.4, 20.2, 20.8, 21., 22.5, 22.5, 22.5, 22.8, 23.5, 23.5,
 23.5, 23.5, 23.5, 24., 24., 24.2, 24.5, 25., 25.5, 25.5, 26.2, 26.5, 27., 28.,
 28.7, 28.9, 28.9, 28.9, 29.4, 30.1, 31.6, 34., 36.5, 37.3, 39., 38.3, 39.4,
 39.3, 41.4, 41.4, 41.3, 42.3, 42.5, 42.4, 42.5, 44.6, 45.2, 45.5, 46., 46.6}
```

Je to změť čísel. Naším úkolem je se v té změti zorientovat a informaci v číslech obsaženou přehledně a srozumitelně prezentovat. *Jaká byla velikost největší ryby (z našeho souboru) a jaká nejmenší? Jaká velikost je „typická“?* Abychom byli schopni odpovědět, musíme se naučit, jak spočítat jednoduché číselné charakteristiky popisující soubor číselných hodnot.

Které z výše uvedených čísel odpovídají druhu cejn velký (druh číslo 1)? Abychom byli schopni odpovědět, musíme se naučit, jak vybrat část údajů splňující určitou podmínku – zde chceme vybrat z vektoru celkových délek tu část, která odpovídá těm řádkům (těm rybám), které mají ve sloupci týkajícím se druhu (sloupec číslo 2) hodnotu 1 odpovídající druhu cejn velký.

1.2 Práce s vektory

Z předešlé kapitolky již víme, že data mají často podobu datové matice. Jejími „základními stavebními kameny“ jsou vektory – jednotlivé řádky (případně sloupce). Naučme se tedy nyní pracovat s vektory v *Mathematice*. Dozvíme se:

- Jak zadat vektor
- Jak z vektoru vybrat jeho část – jeden či více prvků
- Jak spočítat nejdůležitější charakteristiky vektoru (jako je jeho délka)
- Jak upravovat vektor (jak jej uspořádat, jak přidávat, vypouštět či přepisovat prvky vektoru)

Jak zadat vektor

Přestože většina praktických analýz dat začíná načtením dat z (externího) souboru, je užitečné vědět, jak data zadat „ručně“, a tedy v první řadě jak zadat vektor. Základní způsob je následující:

In[6]:=
```
v1 = {7, 6, 2.2, 3}
```

Out[6]=
```
{7, 6, 2.2, 3}
```

Zadali jsme tak vektor délky 4 (má čtyři prvky) a uložili jej pod názvem v1.
Zadáváme-li vektor, jehož prvky jsou členy nějaké posloupnosti, můžeme si práci usnadnit pomocí příkazu **Range[]**:

In[7]:=
```
v2 = Range[5, 11]
```

Out[7]=
```
{5, 6, 7, 8, 9, 10, 11}
```

In[8]:=
```
v3 = Range[5, 11, 2]
```
Out[8]=
```
{5, 7, 9, 11}
```

Výběr jednoho prvku na dané pozici

Jednotlivé prvky můžeme vybrat pomocí dvojitých hranatých závorek. Vyberme například třetí prvek vektoru v1.

In[9]:=
```
v1[[3]]
```
Out[9]=
```
2.2
```

Při použití záporného znaménka jde o číslování od konce, např. poslední prvek vektoru v1 získáme následovně:

In[10]:=
```
v1[[-1]]
```
Out[10]=
```
3
```

Výběr více prvků na daných pozicích

Vyberme první a třetí prvek vektoru v1.

In[11]:=
```
v1[[{1, 3}]]
```
Out[11]=
```
{7, 2.2}
```

Vidíme, že uvnitř hranatých závorek je vlastně vepsán vektor – je to vektor celočíselných indexů pozic, které mají být vybrány. Výběr prvního a třetího prvku je tedy možné udělat tak, že definujeme vektor i1 obsahující indexy 1 a 3 a pak jej použijeme uvnitř hranatých závorek:

In[12]:=
```
i1 = {1, 3};
```

In[13]:=
```
v1[[i1]]
```
Out[13]=
```
{7, 2.2}
```

Výběr prvků splňujících podmínku

Dosti často je třeba vybrat z vektoru jen ty prvky, které splňují nějakou podmínku. V předešlé kapitolce jsme potřebovali z vektoru celkových délek rybích těl vzít jen tu část, která se týká druhu cejn velký. Začněme však s jednoduššími podmínkami. K výběru prvků splňujících podmínku slouží příkazy **`Select[]`** a **`Cases[]`**. Uveďme zde první z nich a vysvětleme si, jak se zapisuje podmínka.

Vyberme z vektoru v1 všechny jeho prvky, které jsou větší než 3.

In[14]:=
```
Select[v1, # > 3 &]
```
Out[14]=
```
{7, 6}
```

Symbolicky bychom strukturu předešlého zápisu mohli vyjádřit takto:

Select [*vektor*, *podminka*]

vektor je název vektoru, ze kterého vybíráme prvky, *podminka* má podobu funkce, jejíž hodnoty jsou logické hodnoty True/False. Budeme-li zadávat podmínku v podobě rovnosti či nerovnosti, budeme k tomu používat tzv. „pure function“, tedy „ryzí funkce“. Tento termín při práci se softwarem *Mathematica* obecně označuje způsob, jak zadat funkci, aniž bychom ji museli pojmenovávat. V našem případě chceme na prvky *vektoru* použít funkci, která určí, zda-li je pro ně splněna určitá

podmínka; konkrétně, zda jsou tyto prvky větší než 3. Funkci zapíšeme takto: # > 3 &. Symbol # zastupuje proměnnou, symbol & na konci zápisu signalizuje, že jde o pure function (ryzí funkci). Následující příklady snad poslouží k lepšímu pochopení role ryzích funkcí (pure functions):

```
In[15]:= #1 > 3 &[5]
Out[15]= True
```

Aplikovali jsme podmínku v podobě pure function na číslo 5. Výsledek je True (pravda), neboť platí 5 > 3.

```
In[16]:= #1 > 3 &[1]
Out[16]= False
```

Aplikovali jsme podmínku v podobě pure function na číslo 1. Výsledek je False (nepravda), neboť neplatí 1 > 3.

Řekněme si nyní ještě pár slov k zápisu podmínek v softwaru *Mathematica*. Rovnost v podmínce se píše pomocí dvojice ==, aby se tak odlišila od symbolu přiřazení =. Neostré nerovnosti se píší (jak je při práci s počítačem obvyklé) pomocí symbolů <=, >=. Podmínky lze spojovat logickými spojkami a zároveň (&&), nebo (||), negace (!). Více o podmínkách se lze dočíst v nápovědě k funkci **`Select[]`** případně v Documentation Center při zadání hesla „guide/TestingExpressions“. Čtenáři, který se s ryzími funkcemi (pure functions) setkává poprvé, doporučujeme nahlédnout do nápovědy k příkazu **`Select[]`** a vyzkoušet si příklady v ní uvedené.

Poznamenejme, že podmínka, která určuje výběr prvků z vektoru, nemusí mít vždy podobu rovnosti či nerovnosti. V některých případech chceme vybrat jen ty prvky, které patří do nějaké větší množiny, např. do množiny celých čísel nebo do množiny všech sudých čísel. Takové podmínky bychom mohli zadat například pomocí příkazu **`MemberQ[]`**.

Jednoduchým příkladem použití podmínky v podobě příslušnosti k množině je úkol vybrat všechna celá čísla z vektoru v1:

```
In[17]:= Select[v1, IntegerQ]
Out[17]= {7, 6, 3}
```

Jak spočítat nejdůležitější charakteristiky vektoru

Často potřebujeme zjistit délku vektoru. K tomu slouží příkaz **`Length[]`**.

```
In[18]:= Length[v1]
Out[18]= 4
```

Použití dalších příkazů jako **`Total[]`**, **`Tally[]`**, **`Count[]`** či **`Position[]`** si vysvětlíme až při aplikaci na sloupce datové tabulky.

Počítání s vektory

Vraťme se na moment k rybářským datům z kapitoly 1.1. Údaj o „výšce„ ryby, který je obsažen v 7. sloupci datové matice, je udáván jako procento celkové délky těla, tedy údaje uloženého v 6. sloupci datové matice. V kapitole 1.1 jsme si ukázali, jak pro jednu konkrétní rybu (rybu číslo 1) vypočítat „výšku“ ryby v centimetrech. Připomeňme použitý příkaz:

```
In[19]:= ryby[[1, 6]] * ryby[[1, 7]] / 100
Out[19]= 11.52
```

Existuje snadný způsob, jak takovýto převod udělat pro všechny ryby (řádky) najednou? Ano, existuje. Místo `ryby[[1,6]]`,což je jedno číslo, napišme `ryby[[All,6]]`, což je číselný vektor

odpovídající 6. sloupci datové matice, a podobně místo `ryby[[1,7]]` napišme `ryby[[All,7]]`. Budeme tedy násobit vektory. Jak takovéto násobení probíhá, není hned zřejmé. V softwaru *Mathematica* však použití některé ze základních matematických operací (jako je sčítání, odčítání, násobení, dělení, mocnění) na dva vektory (stejných délek) znamená, že se tato operace provede „člen po členu“. Výsledkem je tedy vektor stejné délky jako oba vstupní vektory.

```
In[20]:= vyska = ryby[[All, 6]] * ryby[[All, 7]] / 100
```

```
Out[20]= {11.52, 12.48, 12.3778, 12.73, 12.444, 13.6024, 14.1795, 12.67, 14.0049,
 14.2266, 14.2628, 14.3714, 13.7592, 13.9129, 14.9544, 15.438, 14.8604,
 14.938, 15.633, 14.4738, 15.1285, 15.9936, 15.5227, 15.4686, 16.2405,
 16.36, 16.3618, 16.517, 16.8896, 18.957, 18.0369, 18.084, 18.7542, 18.6354,
 17.6235, 8.3804, 8.1454, 8.778, 10.744, 11.7612, 12.354, 4.1472, 5.2983,
 5.5756, 5.6166, 6.216, 6.4752, 6.1677, 6.1146, 5.8045, 6.6339, 7.0334, 6.55,
 6.4, 7.5344, 6.9153, 7.3968, 7.0866, 8.8768, 8.568, 9.485, 6.8475, 6.5772,
 7.4052, 8.3922, 8.8928, 8.5376, 9.396, 9.7364, 10.3458, 11.088, 11.368,
 1.7388, 1.972, 1.7284, 2.196, 2.0832, 1.9782, 2.2139, 2.2139, 2.2044, 2.0904,
 2.43, 2.277, 2.8728, 2.9322, 5.568, 5.7078, 5.9364, 6.2884, 7.29, 6.396,
 7.28, 6.825, 7.786, 6.96, 7.792, 7.68, 8.9262, 10.6863, 9.6, 9.6, 10.812,
 2.112, 3.528, 3.824, 4.5924, 4.588, 5.2224, 5.1992, 5.6358, 5.1376, 5.082,
 5.6925, 5.9175, 5.6925, 6.384, 6.11, 5.64, 6.11, 5.875, 5.5225, 5.856,
 6.792, 5.9532, 5.2185, 6.275, 7.293, 6.375, 6.7334, 6.4395, 6.561, 7.168,
 8.323, 7.1672, 7.0516, 7.2828, 7.8204, 7.5852, 7.6156, 10.03, 10.2565,
 11.4884, 10.881, 10.6091, 10.835, 10.5717, 11.1366, 11.1366, 12.4313,
 11.9286, 11.73, 12.3808, 11.135, 12.8002, 11.9328, 12.5125, 12.604, 12.4888}
```

Zde je vhodné místo pro drobnou odbočku. Předešlý vektor nám totiž na obrazovce či na papíře zabírá zbytečně moc místa. Proč zbytečně? Inu přiznejme si, jak detailně jsme předešlý výstup zkoumali. Většinou jen vizuálně zkontrolujeme, že jde o dosti dlouhý vektor a podíváme se, že hodnoty zde uvedené se pohybují tak asi od pěti do dvaceti cm, což přibližně odpovídá našemu očekávání (vzhledem k uváděným délkám ryb a vzhledem k naší zkušenosti s tím, jak asi vypadá ryba). Naučíme se tedy, jak „šetřit místem“.

Když na konec řádku obsahujícího předešlý příkaz připíšeme středník, výpočet se provede, výsledek se uloží (pod jménem vyska), jen se nezobrazí na obrazovku:

```
In[21]:= vyska = ryby[[All, 6]] * ryby[[All, 7]] / 100;
```

Jak se ale přesvědčíme, že pod jménem vyska je opravdu uloženo to, co si myslíme? Existuje poměrně jednoduchý způsob – pomocí příkazu **`Short[]`**:

```
In[22]:= Short[vyska]
```

```
Out[22]//Short= {11.52, 12.48, 12.3778, <<153>>, 12.5125, 12.604, 12.4888}
```

Porovnejte tento výstup s výše uvedeným úplným výstupem. Zde se zobrazily jen první a poslední dva členy vektoru. Mezi nimi je symbolicky (≪155≫) zapsáno, že není zobrazeno 155 prostředních členů. Snadno se tak přesvědčíme, že celková délka vektoru vyska je 2 + 155 + 2 = 159. To odpovídá počtu ryb (řádků) v našich datech. Poznamenejme ještě, že použitím příkazu **`Short[]`** se vektor vyska nijak nemění (nekrátí), jen se nezobrazí celý.

Vraťme se ale nyní k práci s vektory. Pokud nám zatím není moc jasné, jak se s vektory pracuje, snad nám to pomohou lépe pochopit jednoduché příklady:

Sečtěme vektory v1 a v3 definované v kapitole 1.2 Jak zadat vektor.

```
In[23]:= v1 + v3
```

```
Out[23]= {12, 13, 11.2, 14}
```

Jak se počítalo? Vektor v1 má na první pozici 7, vektor v3 má na první pozici 5, proto vektor v1+v3

bude mít na první pozici hodnotu 7+5=12. Podobně počítáme pro druhou, třetí a čtvrtou pozici.

Protože vektory v1 a v3 byly zadány již před nějakou dobou, chtěli bychom si je zde možná připomenout. Vypišme si je tedy spolu s vektorem jejich součtů:

```
In[24]:= {v1, v3, v1 + v3} // TableForm
```

```
Out[24]//TableForm=
7     6     2.2     3
5     7     9       11
12    13    11.2    14
```

Možná vás napadne, proč jsme nesčítali vektor v1 s vektorem v2. Odpověď je prostá – jde o vektory různých délek. Takové vektory spolu sčítat neumíme.

Z cvičných důvodů zkusme ještě další „vektorové“ operace:

```
In[25]:= v1 - v3
```

```
Out[25]= {2, -1, -6.8, -8}
```

```
In[26]:= v1 * v3
```

```
Out[26]= {35, 42, 19.8, 33}
```

```
In[27]:= v1 / v3
```

$$\text{Out[27]= } \left\{\frac{7}{5}, \frac{6}{7}, 0.244444, \frac{3}{11}\right\}$$

```
In[28]:= v1^v3
```

```
Out[28]= {16 807, 279 936, 1207.27, 177 147}
```

Platnost pravidla vykonání operace „člen po členu“ se neomezuje pouze na situaci, kdy pracujeme se dvěma vektory, ale platí také v případě, že jedním operandem je vektor a druhým číslo (skalár).

```
In[29]:= v1
```

```
Out[29]= {7, 6, 2.2, 3}
```

```
In[30]:= v1 + 7
```

```
Out[30]= {14, 13, 9.2, 10}
```

```
In[31]:= v1 * 5
```

```
Out[31]= {35, 30, 11., 15}
```

```
In[32]:= 1 / v1
```

$$\text{Out[32]= } \left\{\frac{1}{7}, \frac{1}{6}, 0.454545, \frac{1}{3}\right\}$$

```
In[33]:= v1^2
```

```
Out[33]= {49, 36, 4.84, 9}
```

```
In[34]:= Log[v1]
```

```
Out[34]= {Log[7], Log[6], 0.788457, Log[3]}
```

Všimněme si způsobu zobrazení výsledků. Zlomek `1/7` nelze vyjádřit v podobě desetinného čísla s konečným počtem desetinných míst; *Mathematica* nám tedy „nevnucuje“ desetinné číslo a nechává výsledek v podobě zlomku. Jakmile však v zápise čísla použijeme desetinnou tečku, bude výsledek operací s tímto číslem zobrazen v podobě desetinného čísla, například `1/2.2 = 0.454545`. Ze

stejného důvodu je například na poslední řádce ponechána hodnota `Log[6]`, zatímco hodnota `Log[2.2]` je vyčíslena (`0.788457`). Pokud bychom chtěli hodnotu `Log[6]` vyčíslit v podobě desetinného čísla, napsali bychom v zápise čísla 6 desetinnou tečku, tedy 6.0. Můžeme však také použít příkaz **`N[]`** pro numerické vyčíslení.

```
In[35]:= Log[6]
Out[35]= Log[6]

In[36]:= Log[6.0]
Out[36]= 1.79176

In[37]:= N[Log[6]]
Out[37]= 1.79176
```

1.3 Práce s maticemi

Matice není v podstatě nic jiného než vektor, jehož prvky jsou (stejně dlouhé) vektory. To, co jsme se naučili o vektorech, tedy budeme moci použít i pro matice. Přesto bude užitečné podívat se na jistá specifika, které s sebou práce s maticemi nese.

Jak zadat matici

Matici zadáme jako vektor vektorů:

```
In[38]:= M1 = {{6, 1, 4, 3}, {8, 3, 2, 2}, {0, 5, 9, 1}}
Out[38]= {{6, 1, 4, 3}, {8, 3, 2, 2}, {0, 5, 9, 1}}
```

Námi zadaná matice M1 má tři řádky a čtyři sloupce. Abychom ji viděli v podobě, v jaké jsme zvyklí, připíšeme za název matice příkaz TableForm:

```
In[39]:= M1 // TableForm
Out[39]//TableForm=
6   1   4   3
8   3   2   2
0   5   9   1
```

Výběr prvků z matice (podle pozice)

Pozici prvku (či vícero prvků) matice, které chceme vybrat, popíšeme pomocí dvojice čísel, z nichž první udává číslo řádku a druhé číslo sloupce datové matice. Místo čísla může být pochopitelně podmínka, včetně často používané podmínky All pro výběr všech prvků (viz níže uvedené příklady výběru celého řádku či sloupce matice).

Výběr jediného prvku: vyberme např. prvek v 1. řádku, 3. sloupci:

```
In[40]:= M1[[1, 3]]
Out[40]= 4
```

Výběr jednoho řádku: vyberme všechny prvky v 3. řádku:

```
In[41]:= M1[[3, All]]
Out[41]= {0, 5, 9, 1}
```

Výběr jednoho sloupce: vyberme všechny prvky v 3. sloupci:

```
In[42]:= M1[[All, 3]]
Out[42]= {4, 2, 9}
```

Výběr vícero řádků či vícero sloupců:

```
In[43]:= M1[[{1, 3}, All]] // TableForm
Out[43]//TableForm=
6   1   4   3
0   5   9   1
```

```
In[44]:= M1[[All, {1, 3}]] // TableForm
Out[44]//TableForm=
6   4
8   2
0   9
```

Někdy chceme z (velké) matice vybrat její část tak, že vybereme jen řádky v určitém rozmezí (např. 3 až 10), případně sloupce v určitém rozmezí. Mohli bychom k tomu použít příkaz **Range[]**, avšak existuje i jednodušší způsob, jak ukáže následující příklad:

Vyberme z matice M1 řádky 2 až 3 a sloupce 2 až 4:

```
In[45]:= M1[[Range[2, 3], Range[2, 4]]] // TableForm
Out[45]//TableForm=
3   2   2
5   9   1
```

```
In[46]:= M1[[2 ;; 3, 2 ;; 4]] // TableForm
Out[46]//TableForm=
3   2   2
5   9   1
```

Praktický příklad, v němž se nám tato konstrukce bude hodit, najdeme v kapitole 1.5 v části věnované taxonomii.

Výběr prvků z matice (podle podmínky)

Umět z dat vybrat jen takové řádky, které splňují určitou podmínku, patří k nezbytným dovednostem člověka, který provádí analýzu dat. Vraťme se na moment k datům o rybách. Budeme chtít například ze všech dat vybrat jen taková, která odpovídají druhu jelec jesen (druh číslo 2). Opět použijeme příkazu **Select[]**, jako při práci s vektory – viz kapitola 1.2 (neboť, jak jsme již uvedli, datová matice není v softwaru *Mathematica* vlastně ničím jiným než vektorem, jehož prvky jsou vektory).

```
In[47]:= jelci = Select[ryby, #[[2]] == 2 &];
```

```
In[48]:= jelci // TableForm
Out[48]//TableForm=
36   2   270.   23.6   26.   28.7   29.2   14.8   NA
37   2   270.   24.1   26.5  29.3   27.8   14.5   NA
38   2   306.   25.6   28.   30.8   28.5   15.2   NA
39   2   540.   28.5   31.   34.    31.6   19.3   NA
40   2   800.   33.7   36.4  39.6   29.7   16.6   0
41   2   1000.  37.3   40.   43.5   28.4   15.    NA
```

Nebo můžeme vybrat jen ty ryby, jejichž hmotnost přesáhla 1 kilogram.

```
In[49]:= velkeryby = Select[ryby, #[[3]] > 1000 &];
```

In[50]:=
```
velkeryby // TableForm
```

Out[50]//TableForm=

100	6	1250.	52.	56.	59.7	17.9	11.7	NA
101	6	1600.	56.	60.	64.	15.	9.6	NA
102	6	1550.	56.	60.	64.	15.	9.6	0
103	6	1650.	59.	63.4	68.	15.9	11.	0
153	7	1015.	37.	40.	42.4	29.2	17.6	0
155	7	1100.	39.	42.	44.6	28.7	15.4	0
157	7	1100.	40.1	43.	45.5	27.5	16.3	0

Vidíme, že jen sedm ze 159 ryb přesáhlo hmotností 1 kg. V druhém sloupci, který udává číselný kód druhu ryby, je přitom u čtyřech z nich uvedeno číslo 6, u tří číslo 7. Tato čísla odpovídají druhům štika obecná (6) a okoun říční (7).

Povšimněme si nyní podoby podmínky použité pro výběr (v příkladu výběru podle druhu): # [[2]] == 2 &. Jde opět o využití pure function – ryzí funkce (viz kapitola 1.2), kde symbol # zastupuje proměnnou – prvek vektoru „ryby“. Jak však víme, prvky vektoru ryby jsou vektory, z nichž každý odpovídá záznamům o jedné rybě. Naše podmínka se týká jen druhu ryby, tedy druhého čísla ve vektoru záznamů o rybě. Proto píšeme #[[2]]. Když se podmínka týká váhy, která je třetím číslem v záznamu o rybě, budeme psát #[[3]]. Připomeňme, že symbol & je pouze signalizací toho, že jsme použili pure function.

Jelikož vybrat data týkající se pouze jednoho druhu budeme potřebovat dosti často, oplatí se nám datovou matici rozdělit do sedmi částí, z nichž každá bude obsahovat údaje pouze o jednom druhu. To uděláme pomocí příkazu **GatherBy[]**:

In[51]:=
```
Ryby = GatherBy[ryby, #[[2]] &];
```

Vznikla tak vlastně „trojrozměrná“ datová tabulka. Až dosud jsme mluvili o maticích – jejich prvky byly charakterizovány číslem řádku a číslem sloupce. Prvky proměnné Ryby jsou charakterizovány trojicí čísel, z nichž první je číslo druhu a teprve poté následuje číslo řádku a číslo sloupce. Přitom číslem řádku je nyní myšleno číslo pozorování ryby daného druhu. Údaje o všech jelcích (druh č. 2) můžeme nyní snadno získat následujícím způsobem:

In[52]:=
```
Ryby[[2, All, All]] // TableForm
```

Out[52]//TableForm=

36	2	270.	23.6	26.	28.7	29.2	14.8	NA
37	2	270.	24.1	26.5	29.3	27.8	14.5	NA
38	2	306.	25.6	28.	30.8	28.5	15.2	NA
39	2	540.	28.5	31.	34.	31.6	19.3	NA
40	2	800.	33.7	36.4	39.6	29.7	16.6	0
41	2	1000.	37.3	40.	43.5	28.4	15.	NA

Údaje z prvního řádku této datové tabulky bychom získali následovně:

In[53]:=
```
Ryby[[2, 1, All]]
```

Out[53]=
```
{36, 2, 270., 23.6, 26., 28.7, 29.2, 14.8, NA}
```

Úpravy matic

V části Počítání s vektory kapitoly 1.2 jsme řešili problém, jak hodnoty v sloupci týkajícím se „výšky“ ryb přepočítat z procent celkové délky těla ryby na standardní délkové údaje vyjádřené v centimetrech. Připomeňme zde příkaz

In[54]:=
```
vyska = ryby[[All, 6]] * ryby[[All, 7]] / 100;
```

Výsledkem je nový (samostatný) vektor nazvaný vyska. Nyní bychom chtěli, aby se tento vektor stal součástí datové matice. Naším úkolem je tedy přidat k datové matici nový sloupec vyjadřující

„výšku“ ryby v centimetrech.

Způsobů, jak se s tímto úkolem vypořádat, je vícero. My si ukážeme postup pomocí příkazu **Join[]**. Anglické sloveso *to join* znamená *spojit*. Nepřekvapí nás tedy, že příkaz **Join[]** slouží ke spojení dvou vektorů. Prvním ze spojovaných vektorů je vektor ryby. Připomeňme, že prvkem tohoto vektoru jsou vektory (délky devět). Druhý ze spojovaných vektorů by měl tudíž být rovněž vektor, jehož prvky jsou vektory, byť půjde o vektory délky jedna (k stávajícím devíti sloupcům přidáváme jeden nový). Vektor vyska, který chceme připojit, ale není vektorem vektorů, ale vektorem číselných hodnot. Pomůžeme si následujícím trikem:

In[55]:=
```
novy = Transpose[{vyska}];
```

Výsledek nechme zobrazit ve zkrácené formě:

In[56]:=
```
novy // Short
```

Out[56]//Short=
```
{{11.52}, {12.48}, {12.3778}, <<154>>, {12.604}, {12.4888}}
```

Vznikl tedy vektor (nazvaný novy), jehož prvky jsou vektory délky jedna. Ten již můžeme spojit s vektorem ryby.

In[57]:=
```
ryby = Join[ryby, novy, 2];
```

Klíčovou roli v předešlém zápisu tvoří dvojka, jakožto třetí argument příkazu **Join[]**. Tato dvojka totiž určuje, jak – na jaké úrovni – mají být vektory vektorů (ryby a novy) spojeny. Kdybychom místo dvojky uvedli hodnotu 1 (což je stejné, jako kdybychom neuvedli žádnou hodnotu), vektory by se seřadili za sebe – vznikl by vektor o 2-krát 159 prvcích. Prvních 159 prvků by byly vektory obsažené ve vektoru ryby, zbylých 159 vektorů by byly vektory obsažené ve vektoru novy. Když však uvedeme, že se mají vektory spojit na úrovni 2, říkáme, že se mají spojit prvek po prvku. Vznikne vektor délky 159, jehož první člen je spojením prvního členu vektoru ryby a prvního členu vektoru novy atd.

Výsledek jsme uložili do datové matice ryby. Původní datová matice tak byla nahrazena novou, která obsahuje o jeden sloupec víc. Je samozřejmě možné místo přepsání původní datové matice vytvořit datovou matici novou (nazvanou třeba RybyNew). Pokud bychom však při vytvoření každého nového sloupce vytvářeli také novou datovou matici, mohlo by brzy dojít k značně nepřehledné situaci.

O tom, že jsme při přidávání nového sloupce do datové matice postupovali správně, se můžeme přesvědčit vypsáním nově vzniklé datové matice:

```
ryby // TableForm
```

Zde jsme výstup z důvodu úspory místa netiskli.

Jako cvičení můžeme datovou matici ryby rozšířit o další sloupec – šířku ryby v cm:

In[58]:=
```
sirka = ryby[[All, 6]] * ryby[[All, 8]] / 100;
```

In[59]:=
```
ryby = Join[ryby, Transpose[{sirka}], 2];
```

```
ryby // TableForm
```

V druhém sloupci naší datové matice je v podobě celého čísla z množiny {1,2,3,4,5,6,7} uložena informace o druhu ryb. Abychom nemuseli neusále hovořit o druhu č. 1, druhu č. 2, ... bylo by dobré vytvořit nový sloupec, ve kterém by byla uložena slovní informace o druhu ryb.

In[60]:=
```
cesky = {"cejn", "jelec", "plotice", "cejnek", "koruška", "štika", "okoun"};
```

Původní vektor:

In[61]:=
```
druh = ryby[[All, 2]];
```

```
In[62]:= druh // Short
Out[62]//Short= {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      <<115>>, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7}
```

Nový vektor:

```
In[63]:= DruhSlovne = druh /. x_Integer :> cesky[[x]];

In[64]:= DruhSlovne // Short
Out[64]//Short= {cejn, cejn, cejn, cejn, cejn, <<149>>, okoun, okoun, okoun, okoun, okoun}
```

Při definici vektoru DruhSlovne jsme použili „patterns“. Více o této programovací technice se lze dočíst v nápovědě pod heslem „tutorial/Introduction-Patterns“. Příkaz druh/.x_Integer:>cesky[[x]] je příkazem nahrazení každého celého čísla x ve vektoru druh x-tým prvkem vektoru cesky. Tak například na všech místech, kde je ve vektoru druh napsaná 1, bude ve vektoru DruhSlovne napsáno „cejn“, neboť to je 1. prvek vektoru cesky.

Nyní nově vytvořený vektor „připojíme“ k datové tabulce:

```
In[65]:= ryby = Join[ryby, Transpose[{DruhSlovne}], 2];

ryby // TableForm
```

1.4 Řekněte si Mathematice o data! Třeba o rybách

...a byly řeky po setmění
kde jsi moh čekat na sumce...

Pavel Šrut

V úvodu kapitoly 1.1 o tom, jak vypadají data, jsme zdůraznili značnou šíři oborů, kterých se mohou týkat data, která má statistik analyzovat – od medicíny, přes fyziku až k sociologii. Statistik většinou není odborníkem v těchto oblastech. Pokud spolupráce statistika s odborníkem nepředchází již samotnému sběru dat (což je dosti častý, nikoliv však „ideální“, případ), je první etapou při analýze dat schůzka statistika s odborníkem v některé z výše uvedených oblastí lidského vědění – například s lékařem, fyzikem či sociologem. Tito odborníci většinou statistikovi „přinesou data“ a, pokud jde o opravdové odborníky, vysloví svou doměnku (hypotézu), kterou chtějí potvrdit či vyvrátit.

Josef Tvrdík v článku *Medici, lékaři a statistika* [1] shrnul své zkušenosti a popsal, co potřebuje a očekává lékař přicházející se svými daty: „...Především potřebuje dobře porozumět datům, se kterými přichází. To ostatně nutně potřebujeme i my, pokud je máme statisticky zpracovávat. A není to vůbec samozřejmá věc ani pro jednu zúčastněnou stranu ...“. Porozumění datům hraje vskutku zásadní roli při jejich analýze.

V této kapitole analyzujeme rybářská data. Nemáme, bohužel, možnost setkat se s odborníky, kteří tato data před téměř sto lety pořídili. Nemáme tak možnost položit jim otázky, které nás v souvislosti s daty napadnou, např. proč je tolik chybějících hodnot v sloupci pohlaví ryby (zda pohlaví nebylo určováno, nebylo možno určit nebo vůbec u některých ryb nemá smysl určovat). Může nás také napadnout, že veličiny jako je délka či hmotnost ryby úzce souvisí s jejich věkem. Každá „mladá ryba“ je „malá“ a teprve později vyroste. Potřebovali bychom se tedy dozvědět, zda do námi analyzovaného datového souboru byli zařazeni např. jen dospělí jedinci, nebo zda šlo o všechny chycené ryby bez ohledu na věk. V druhém případě by totiž bez informace o věku ryb (ta v datech není) mohly být závěry týkající se velikosti ryb zavádějící. Představme si například, že jsme ulovili jen samé malé štiky a staré cejnky. Pak bychom mohli usoudit, že cejnci jsou větší než štiky, i když při porovnání pouze dospělých jedinců obou druhů bychom došli k opačnému závěru. V této práci budeme předpokládat, že námi analyzované údaje se týkají pouze dospělých jedinců a jsou tudíž

porovnatelné. Tento předpoklad by nám v praxi měl potvrdit (či vyvrátit) člověk, který data sbíral. Ačkoliv nemáme možnost s tímto člověkem hovořit, máme možnost se o datech dozvědět většinu důležitých informací – jsou obsaženy v souboru fishcatch.txt.

Pojďme si nyní utvořit ještě o něco lepší představu o datech, která máme analyzovat. Společnost Wolfram, vyvíjející software *Mathematica*, vyvinula mj. volně přístupný webový vyhledávač WolframAlpha. Jde o osobitou alternativu k vyhledávačům jako je google nebo v České republice používaný seznam.cz. Pomocí webové stránky http://www.wolframalpha.com/ si tak můžeme udělat představu o jednotlivých druzích ryb, jichž se námi analyzovaná data týkají.

Stejnou informaci však můžeme získat také přímo pomocí softwaru *Mathematica*. Stačí nový řádek začít dvojicí symbolů == a za ně napsat stejný „dotaz“, jaký bychom položili webovému vyhledávači, např. WolframAlpha. Řekněme si tedy Mathematice o data o živočišném druhu *Abramis brama* (latinského názvu použijeme pro jeho jednoznačnost):

In[66]:= **Abramis brama**

Input interpretation:

bream (animal)

Alternate common names: More

aral bream | bowfin | bronze bream | carp bream | common bream | …

Taxonomy:

kingdom	Animalia (animals)
phylum	Chordata (chordates)
class	Actinopterygii (ray–finned fishes)
order	Cypriniformes (carps, suckers, loaches...)
family	Cyprinidae
genus	Abramis
species	*Abramis brama* (bream)

Image:

Výstup, který zde přetiskujeme, je krácen – skutečný výstup je o něco rozsáhlejší. Tato zkrácená verze nicméně stačí k tomu, abychom si o podobě výstupu udělali představu.

Výstup, který máme k dispozici (a se kterým budeme dále pracovat), má několik částí:

- První položkou je vždy „Input interpretation“ obsahující heslo, k němuž se vztahují všechny následující záznamy. Pokud jsme vyhledávali záznamy o rybě a zadali jsme její latinský název, je tento název jednoznačně zadaným dotazem (existuje k němu jen jeden záznam uložený pod

anglickým termínem „bream (animal)“. Některé jiné dotazy však jednoznačné nejsou. Co se například stane, zadáme-li dotaz „Michael Jackson“? Při vyslovení toho jména se většině z nás vybaví osobnost „krále popu“, amerického zpěváka Michaela Jacksona (1958–2009). Avšak historie zná i jiné Michaely Jacksony – hráče amerického fotbalu narozeného v roce 1957, jiného hráče amerického fotbalu (stejného jména) narozeného v roce 1969, hráče basketbalu narozeného v roce 1964 a konečně Michaela Jacksona – politika. Na podobné nejednoznačnosti, jimž se vyhneme při detailnějším zadání dotazu, jsme zvyklí například z wikipedie:

```
http://en.wikipedia.org/wiki/Michael_Jackson_(disambiguation)
```

Mimochodem, výstupem k heslu Michael Jackson je také časová řada počtu zobrazení (anglické verze) stránky tohoto zpěváka na wikipedii. Podobné časové řady jsou k dispozici i pro ostatní Michaely Jacksony. Také u nich lze vypozorovat obrovský "nárůst zájmu" na konci června 2009, kdy zemřel jejich jmenovec – král popu.

- Druhá položka obsahuje další možné anglické termíny označující rybu „bream“: aral bream, bowfin, bronze bream, carp bream, common bream. Protože je záznamů mnoho, nejsou zobrazeny všechny (jen výše uvedených pět). Kliknutím na tlačítko More (více) v pravém horním rohu příslušného políčka získáme další názvy: danube bream, eastern bream, freshwater bream.
- Další z položek je označena termínem Taxonomy. Pokud se vám při vyslovení termínu taxonomie nic moc nevybaví, pak věřte, že podle wikipedie je taxonomie „teorie a praxe klasifikace organismů podle určitých pravidel do jednotlivých hierarchicky uspořádaných taxonomických kategorií“. Možná se vám pak vybaví, že jste se na základní či střední škole učili, že biologové zařazují živé organismy podle jejich příslušnosti k říši, kmenu, oddělení, třídě, řádu, čeledi, rodu a konečně k druhu. Anglické ekvivalenty těchto českých termínů, pro přehlednost zde uvedené v tabulce, můžeme najít například na stránkách wikipedie:

```
http://cs.wikipedia.org/wiki/Kategorie_(biologie)
```

česky	anglicky
říše	kingdom
kmen	phylum
oddělení	division
třída	class
řád	order
čeleď	family
rod	genus
druh	species

Zajímá-li vás, jak "vytvořit" pomocí softwaru *Mathematica* výše uvedenou tabulku přímo z tabulky na uvedené stránce wikipedie, přečtěte si to v kapitole 1.5.

- Posledním zobrazeným výstupem je fotografie, pomocí které si my „nerybáři“ uděláme aspoň přibližnou představu o tom, jak vlastně takový cejn (bream) vypadá.

Udělali jsme si představu o cejnech. Zbývá však dalších šest druhů. Samozřejmě bychom mohli postupně zadávat všechny latinské názvy a dívat se na „profily“ jednotlivých druhů. Můžeme ale udělat něco chytřejšího – pracovat s údaji o všech sedmi druzích najednou a vzájemně druhy porovnávat.

Klíčovou skutečností je, že výše uvedené informace můžeme získat přímo pomocí příkazu **WolframAlpha[]**. Nemusíme si přitom dělat těžkou hlavu s tím, jak by měl příkaz vypadat – můžeme ho získat přímo. V pravém horním rohu jednotlivých obdélníčků, na něž je výstup rozdělen, se zobrazuje znaménko + (v tištěné verzi toto znaménko není zobrazeno, je třeba si vyzkoušet příkaz ==“Abramis brama“). Kliknutím na toto znaménko získáme možnost extrahovat příslušná data, např. zvolením možnosti „Computable data“ (v případě práce s obrázky volíme možnost „Subpod content“).

Názvy ryb anglicky

Naším prvním úkolem bude získat k latinským názvům, které jsou uvedeny v souboru fishcatch.txt jejich anglické ekvivalenty. Porovnáme pak své výsledky s tím, co je uvedeno v souboru fishcatch.txt.

Budeme vycházet z položky „Input interpretation“. Klikněme na znaménko + v pravém horním rohu příslušného obdélníku ve výstupu, který jsme získali při zadání dotazu „Abramis brama“ a zvolme možnost „Computable data“. Automaticky se vygeneruje a provede následující příkaz:

```
In[67]:= WolframAlpha["Abramis brama", {{"Input", 1}, "ComputableData"}]
Out[67]= bream
```

Nyní bychom totéž chtěli udělat pro druh číslo 2, tedy „Leuciscus idus“. Můžeme to udělat tak, že ve výše uvedeném příkazu zaměníme výraz „Abramis brama“ za výraz „Leuciscus idus“:

```
In[68]:= WolframAlpha["Leuciscus idus", {{"Input", 1}, "ComputableData"}]
Out[68]= ide
```

Při získávání anglických názvů tak postupně měníme latinský název ryby, který uvádíme v předešlém příkaze. Využijeme příkazu **`Table[]`** k zjištění všech anglických jmen najednou.

```
In[69]:= NazvyLatinsky =
           {"Abramis brama", "Leuciscus idus", "Leuciscus rutilus", "Abramis bjoerkna",
            "Osmerus eperlanus", "Esox lucius", "Perca fluviatilis"};

In[70]:= NazvyAnglicky = Table[WolframAlpha[latinsky, {{"Input", 1}, "ComputableData"}],
           {latinsky, NazvyLatinsky}]
Out[70]= {bream, ide, roach, silver bream, european smelt, jack, perch}
```

Základem výše uvedeného příkazu je nám již dobře známý příkaz

`WolframAlpha[latinsky,{{„Input“,1},“ComputableData“}]`,

v němž slovo *latinsky* nahrazuje latinský název ryby, jejíž anglické jméno chceme získat. Jaké hodnoty se za výraz *latinsky* mají dosadit je určeno hned poté:

`{latinsky,NazvyLatinsky}`.

Tento zápis říká, že za výraz *latinsky* se mají postupně dosazovat všechny prvky obsažené ve vektoru *NazvyLatinsky*.

Výsledek je pak uložen jako vektor nazvaný *NazvyAnglicky*.

Porovnejme nyní vektor NazvyAnglicky s tím, co je napsáno v souboru fishcath.txt. Místo ide je zde whitefish, druh číslo 4 (silver beam) není vůbec vyplněn, místo european smelt je zde jen smelt a místo jack je zde pike. Poznamenejme, že zatímco název *pike* bychom u druhu „Esox lucius“ našli v kolonce „Alternate common names“ udávající jiná běžná pojmenování, u druhu „Leuciscus idus“ bychom označení *whitefish* hledali marně. Místo termínu ide bychom mohli použít termín orfe, použití termínu whitefish je však nesprávné. Naše pozornost věnovaná názvům jednotlivých ryb tak vedla k doplnění jedoho chybějícího údaje (anglický název druhu číslo 4) a k zjištění jednoho chybného údaje (anglického názvu druhu číslo 2).

Doplňme ještě české názvy:

In[71]:=
```
NazvyCesky = {"Cejn velký", "Jelec jesen", "Plotice obecná",
    "Cejnek malý", "Koruška evropská", "Štika obecná", "Okoun říční"};
```

Rybí portréty

Informace, kterou můžeme získat pomocí WolframAlpha nemusí mít jen podobu číselnou či slovní, ale může jít také o informaci v podobě obrazu. V obdélníčku nazvaném Image je u všech sedmi v datech obsažených druhů ryb fotografie jejich zástupce. Nahlédněme tedy nyní do němých tváří těch, jichž se naše data týkají.

Příkaz pro získání „portrétu“ druhu „Abramis brama“ vygenerujeme opět kliknutím na znaménko + v pravém horním rohu příslušného obdélníčku. Volba „Computable data“, kterou jsme použili při získávání anglických názvů ryb, nyní není k dispozici. Vyberme tedy možnost „Subpod content“; tím získáme následující příkaz:

In[72]:=
```
WolframAlpha["Abramis brama", {{"Image:SpeciesData", 1}, "Content"}]
```

Out[72]=

Opět bychom mohli využít příkazu **Table[]** k tomu, abychom získali „portréty“ zástupců všech sedmi druhů. Pro pořádek zde uvedeme příkaz, ale výstup nebudeme zatím zobrazovat, jen ho uložíme pod názvem *Portrety* (přidáme na konci příkazu středník).

In[73]:=
```
Portrety = Table[WolframAlpha[latinsky, {{"Image:SpeciesData", 1}, "Content"}],
    {latinsky, NazvyLatinsky}];
```

Dejme nyní získanou obrazovou informaci (obsaženou ve vektoru Portrety) do souvislosti s informací slovní – s českými, latinskými a anglickými názvy jednotlivých druhů. Vytvoříme datovou matici (pojmenujeme ji třeba ProfilyRyb), která bude obsahovat sedm vektorů (ke každému sledovanému druhu ryb jeden), z nichž každý bude mít čtyři prvky – portrét ryby a její český, latinský a anglický název.

In[74]:=
```
ProfilyRyb = Transpose[{Portrety, NazvyCesky, NazvyLatinsky, NazvyAnglicky}];
```

Kdybychom nepoužili příkaz **Transpose[]**, měli bychom v datové matici jen čtyři vektory, každý o délce 7. To, mimochodem, nemusí být vůbec na škodu. Můžete sami vyzkoušet, jaké výstupy bychom pak dostávali.

Než budeme zobrazovat datovou matici ProfilyRyb, přidejme na její začátek ještě jeden vektor (délky 4, jako mají ostatní vektory ve vektoru ProfilyRyb), který bude obsahovat „názvy sloupců“ datové matice. Použijeme k tomu příkaz **Prepend[]**:

In[75]:=
```
ProfilyRyb = Prepend[ProfilyRyb,
    {"Obrázek", "Název česky", "Název latinsky", "Název anglicky"}];
```

Zobrazme nyní datovou matici ProfilyRyb. Pomocí příkazu **Grid[]** můžeme výstup nejrůznějším způsobem upravovat. Spokojme se nyní s tím, že ke každému druhu ryby přidáme „rámeček“ ohraničující údaje o tomto druhu:

In[76]:=
```
Grid[ProfilyRyb, Frame → {False, All}]
```

Out[76]=

Obrázek	Název česky	Název latinsky	Název anglicky
	Cejn velký	Abramis brama	bream
	Jelec jesen	Leuciscus idus	ide
	Plotice obecná	Leuciscus rutilus	roach
	Cejnek malý	Abramis bjoerkna	silver bream
	Koruška evropská	Osmerus eperlanus	european smelt
	Štika obecná	Esox lucius	jack
	Okoun říční	Perca fluviatilis	perch

Jedna z věcí, která je na statistice krásná, je to, že se statistik při své práci dozví mnohé z oborů, v nichž sám není odborníkem, ať už jde o medicínu, fyziku nebo třeba rybaření.

Taxonomie

Údaje, které jsme získali prostřednictvím WolframAlpha mohou být pro nás zajímavým zdojem hypotéz. Pomocí taxonomických dat můžeme mezi sedmi druhy, jichž se data týkají, najít skupiny, jejichž členové mají podobné vlastnosti. Již podobnost názvů cejn velký a cejnek malý (bream, silver bream) napovídá, že některé námi sledované druhy jsou si „blízké“. Cejnek malý bude zřejmě podobnější cejnovi velkému (byť se zřejmě bude lišit velikostí) než například štice. Které druhy tedy

tvoří skupinu? To zjistíme na základě porovnání (taxonomické) klasifikace námi sledovaných druhů.

Začněme opět tím, že „klikacím způsobem“ (jako jsme to provedli u názvů a portrétů ryb) zjistíme u jednoho druhu – „Abramis brama“ – jak má vypadat příkaz pro získání taxonomických údajů:

```
In[77]:= WolframAlpha["Abramis brama", {{"Taxonomy:SpeciesData", 1}, "ComputableData"}]

Out[77]= {{kingdom, animals}, {phylum, chordates},
          {class, ray-finned fishes}, {order, carps, suckers, loaches...},
          {family, Cyprinidae}, {genus, Abramis}, {species, bream}}
```

Vidíme, že výstupem je vektor, jehož prvky jsou dvoučlenné vektory. Ty mají na prvním místě vždy uvedeno, o jakou toxonomickou kategorii jde (kingdom, phylum, ...), na druhém místě je pak vlastní informace o rybě – její klasifikace. Nás tedy zajímají jen druhé členy. Ty můžeme získat například takto (viz kapitola 1.3 – výběr prvků z matice podle pozice) :

```
In[78]:= WolframAlpha["Abramis brama",
           {{"Taxonomy:SpeciesData", 1}, "ComputableData"}][[All, 2]]

Out[78]= {animals, chordates, ray-finned fishes,
          carps, suckers, loaches..., Cyprinidae, Abramis, bream}
```

```
In[79]:= Taxonomie = Table[
           WolframAlpha[ryba, {{"Taxonomy:SpeciesData", 1}, "ComputableData"}][[All, 2]],
           {ryba, NazvyLatinsky}]; Taxonomie // TableForm
```

Out[79]//TableForm=

animals	chordates	ray-finned fishes	carps, suckers, loaches...
animals	chordates	ray-finned fishes	carps, suckers, loaches...
animals	chordates	ray-finned fishes	carps, suckers, loaches...
animals	chordates	ray-finned fishes	carps, suckers, loaches...
animals	chordates	ray-finned fishes	smelts, noodlefishes...
animals	chordates	ray-finned fishes	pikes
animals	chordates	ray-finned fishes	perches, butterflyfishes, cichlids,

Výsledná tabulka je dosti veliká. Dokonce tak veliká, že při tisku „přeteče“ okraj stránky a vytiskne se jen její část. To nám nevadí, to podstatné je stejně ukryto v prvních sloupcích této tabulky. První tři sloupce – odpovídající říši, kmenu a třídě (kingdom, phylum, class) – jsou u všech sedmi druhů stejné. Zabýváme se tedy živočichy patřícími do kmene strunatců (chordata), do třídy paprsko-ploutvých (Actinopterygii, anglicky ray-finned fishes). Ve čtvrtém sloupci odpovídajícím řádu (order) se už ale zařazení jednotlivých druhů začíná lišit. Zatímco druhy 1 až 4 (cejn, jelec, plotice a cejnek) patří ke stejnému řádu máloostních (Cypriniformes), tři zbývající druhy (koruška, štika a okoun) patří každý k jinému řádu.

Výše uvedená informace o zařazení jednotlivých druhů do řádů může sloužit k formulaci hypotéz týkajících se například rozdílnosti (či stejnosti) sledovaných druhů v rámci řádu máloostních. Je totiž možné (ba dokonce pravděpodobné), že některé charakteristiky budou v rámci jednoho řádu podobné (ne-li totožné), zatímco stejná charakteristika u jiného řádu bude mít odlišnou hodnotu.

Jak již bylo řečeno, tabulka, která obsahuje taxonomické údaje o jenotlivých druzích, je pro tisk příliš velká. Nás by mohla zajímat informace o dalším zařazení druhů řádu máloostních, tedy o zařazení druhů cejn, jelec, plotice a cejnek do čeledi (family), případně rodu (genus). Zobrazme proto příslušnou část tabulky Taxonomie (řádky 1 až 4, sloupce 5 až 7):

```
In[80]:= Taxonomie[[1 ;; 4, 5 ;; 7]] // TableForm
```

Out[80]//TableForm=

Cyprinidae	Abramis	bream
Cyprinidae	Leuciscus	ide
Cyprinidae	Rutilus	roach
Cyprinidae	Blicca	silver bream

Vidíme, že všechny čtyři sledované druhy řádu máloostních patří do čeledi Cyprinidae (kaprovití) (viz první sloupec této tabulky), nicméně každý k jinému rodu (viz druhý sloupec tabulky). To mimo jiné znamená, že „cejnek není o nic více příbuzný cejnovi než plotice nebo jelec“ a to i přes zjevnou jazykovou podobu názvů těchto dvou druhů.

1.5 Řekněte si o data z internetu! Třeba o taxonomii

V předchozí kapitolce jsme pracovali s tabulkou obsahující anglické ekvivalenty českých termínů používaných v taxonomii (říše, kmen, oddělení, třída, řád, čeleď, rod a druh). Tato tabulka byla vytvořena na základě tabulky na stránce wikipedie:

`http://cs.wikipedia.org/wiki/Kategorie_(biologie)`

Nyní si ukážeme, jak jsme postupovali při zpracování této tabulky.

Prvním krokem je import dat z internetu:

In[81]:=
```
vsechno = Import["http://cs.wikipedia.org/wiki/Kategorie_(biologie)", "Data"];
```

Kdybychom nechali vypsat proměnnou vsechno na obrazovku, objevila by se nepřehledná změť slov, které jsou obsaženy v (přehledných) tabulkách na výše uvedené stránce wikipedie. My budeme chtít pracovat s tabulkou nazvanou „Základní taxonomické kategorie vícejazyčně (zoologie a botanika)“. V záhlaví jejích sloupců jsou postupně slova „česky“, „latinsky“, „anglicky“,... Najděme tedy v importované změti (proměnná vsechno) slova „česky“ a „latinsky“:

In[82]:=
```
Position[vsechno, "česky"]
```

Out[82]=
```
{{1, 1, 3, 2, 1}, {1, 1, 8, 2, 1}}
```

In[83]:=
```
Position[vsechno, "latinsky"]
```

Out[83]=
```
{{1, 1, 3, 2, 2}, {1, 1, 8, 2, 2}}
```

Nyní už máme představu, kde v naší změti hledat. Řádek obsahující celé záhlaví námi zpracovávané tabulky zobrazíme pomocí příkazu:

In[84]:=
```
vsechno[[1, 1, 3, 2, All]]
```

Out[84]=
```
{česky, latinsky, anglicky, německy, španělsky, francouzsky, japonsky}
```

Celou tabulku pak získáme, když i předposlední hodnotu (2) zaměníme za hodnotu All:

In[85]:=
```
tabulka = vsechno[[1, 1, 3, All, All]];
```

In[86]:=
```
tabulka // TableForm
```

Out[86]//TableForm=

String[]

česky	latinsky	anglicky	německy	španělsky	francouzsky	ja
říše	regnum	kingdom	Reich	reino	règne	界
kmen 1	phylum	phylum	Stamm	filo	phylum	門
oddělení 2	divisio	division	Abteilung	división	embranchement	門
třída	classis	class	Klasse	clase	classe	綱
řád	ordo	order	Ordnung	orden	ordre	目
čeleď	familia	family	Familie	familia	famille	科
rod	genus	genus	Gattung	género	genre	属
druh	species	species	Art	especie	espèce	種

Vynechme nyní první prvek tabulky (`String[]`)

In[87]:=
```
tabulka = Drop[tabulka, 1];
```

Ještě nás „zlobí“ číselné hodnoty 1 a 2 v prvním sloupci tabulky. V původní tabulce na wikipedii byly u termínů „kmen“ a „oddělení“ totiž ještě vysvětlivky (číslované 1 a 2). Těchto čísel se zbavíme

pomocí příkazu **StringReplace[]**:

In[88]:=
```
tabulka[[All, 1]] = StringReplace[tabulka[[All, 1]], DigitCharacter .. → ""];
```

Nyní z tabulky vybereme sloupce 1 a 3 obsahující české a anglické taxonomické termíny:

In[89]:=
```
CZEN = tabulka[[All, {1, 3}]];
```

In[90]:=
```
CZEN // TableForm
```

Out[90]//TableForm=

česky	anglicky
říše	kingdom
kmen	phylum
oddělení	division
třída	class
řád	order
čeleď	family
rod	genus
druh	species

Tabulku pak můžeme pomocí příkazu **Grid[]** připravit do podoby vhodné k prezentaci:

In[91]:=
```
Grid[CZEN, Dividers → {2 → True, 2 → True}, Frame → {All, False}]
```

Out[91]=

česky	anglicky
říše	kingdom
kmen	phylum
oddělení	division
třída	class
řád	order
čeleď	family
rod	genus
druh	species

Čtenáři, který se softwarem *Mathematica* nemá mnoho zkušeností, pomůže k pochopení jednotlivých voleb výše uvedeného příkazu, když vyzkouší postupně příkazy

```
Grid[CZEN, Frame -> {All, False}]
Grid[CZEN, Dividers -> {2 -> True, 2 -> True}]
```

2 Popis rozdělení jednoho znaku

Pro čtenáře, který vynechal předešlou kapitolu, zopakujme nejdůležitější kroky načtení dat z kapitoly 1:

In[92]:=
```
ryby =
  Import["http://www.amstat.org/publications/jse/datasets/fishcatch.dat.txt",
   "Data"];
vyska = ryby[[All, 6]] * ryby[[All, 7]] / 100;
cesky = {"cejn", "jelec", "plotice", "cejnek", "koruška", "štika", "okoun"};
druh = ryby[[All, 2]] /. x_Integer :> cesky[[x]];
ryby = Join[ryby, Transpose[{vyska, druh}], 2];
Ryby = GatherBy[ryby, #[[2]] &];
n = Dimensions[ryby][[1]];
```

2.1 Kvalitativní znak

Datový soubor, který v tomto textu zkoumáme, obsahuje údaje o 159 rybách. U každé ryby přitom máme k dispozici záznamy o 8 různých znacích (vlastnostech). „Druh ryby“ je typickým *kvalitativním* znakem. V „kolonce druh ryby“ je (u ryb z našeho datového souboru) uvedena vždy jedna z následujících možností: cejn, jelec, plotice, cejnek, koruška, štika, okoun. „Hodnoty“ znaku „druh ryby“ jsou tedy popsány slovně. Slovní popis hodnot je typickou vlastností kvalitativních znaků.

Kromě toho, že „druh ryby“ je znakem kvalitativním, můžeme říci, že jde o znak *nominální* (případně, že je tento znak měřen na nominální škále). Říkáme tím, že hodnoty tohoto znaku nejsou uspořádané. Jestliže jsme uvedli sedm druhů v pořadí od cejna po okouna, je toto pořadí stejně dobré, jako kdybychom možné hodnoty uvedli v libovolném jiném pořadí (např. seřazeny podle abecedy). Jinak by tomu bylo, kdybychom uváděli třeba znak „šupinatost ryb“ s možnostmi „bez šupin“, „málo šupinatá“, „hodně šupinatá“. Zde jde o kategorie, jejichž pořadí má smysl, ať už je seřadíme „vzestupně“, tedy tak, jak je výše uvedeno, nebo sestupně, tedy v opačném pořadí. Znak „šupinatost“ by byl rovněž kvalitativní (hodnoty jsou popsány slovně), ale už ne nominální, nýbrž *ordinální* (neboť kategorie jsou uspořádané). Učebnicovým příkladem ordinálního znaku je například dosažené vzdělání (základní, středoškolské bez maturity, středoškolské s maturitou, vysokoškolské) nebo třeba hodnocení nějakého výrobku (nelíbí se vůbec, spíše se nelíbí, je vnímán neutrálně, spíše se líbí, líbí se hodně).

Tabulka (absolutních) četností

Základní představu o kvalitativní veličině si uděláme na základě tabulky četností, tedy tabulky, v níž jsou k jednotlivým variantám znaku (v našem případě k jednotlivým druhům ryb) uvedeny jejich počty v datovém souboru. Software *Mathematica* nabízí pro vytvoření četnostní tabulky příkaz **Tally[]**:

In[99]:=
```
CetnostDruhu = Tally[druh]
```

Out[99]=
```
{{cejn, 35}, {jelec, 6}, {plotice, 20},
 {cejnek, 11}, {koruška, 14}, {štika, 17}, {okoun, 56}}
```

In[100]:=
```
CetnostDruhu // TableForm
```

Out[100]//TableForm=

cejn	35
jelec	6
plotice	20
cejnek	11
koruška	14
štika	17
okoun	56

Seřaďme nyní druhy podle četnosti od nejpočetnějšího po nejméně zastoupený. Budeme tedy řadit vektor CetnostDruhu. Jeho prvky jsou dvouprvkové vektory, kde na první pozici je název druhu, na druhé (poslední) je pak počet jeho zástupců v datech (tzv. absolutní četnost). Právě podle těchto počtů chceme řadit. Použijeme k tomu příkaz **SortBy[]** (česky „řadit podle“). V něm nejprve uvedeme, že chceme řadit vektor CetnostDruhu, a doplníme, že řadíme podle hodnot posledních (Last) prvků vektorů obsažených v tomto vektoru.

In[101]:=
```
SortBy[CetnostDruhu, Last]
```

Out[101]=
```
{{jelec, 6}, {cejnek, 11}, {koruška, 14},
 {štika, 17}, {plotice, 20}, {cejn, 35}, {okoun, 56}}
```

Protože jsou nyní druhy seřazeny od nejméně zastoupeného a my jsme chtěli opačné pořadí, použijeme příkaz **Reverse[]**:

In[102]:=
```
DruhyDleCetnosti = Reverse[SortBy[CetnostDruhu, Last]]
```

Out[102]=
```
{{okoun, 56}, {cejn, 35}, {plotice, 20},
 {štika, 17}, {koruška, 14}, {cejnek, 11}, {jelec, 6}}
```

Nyní připravíme tabulku do podoby, v níž bude možno ji prezentovat. Poznamenejme, že pokud tabulka slouží jen pro naši orientaci, nemusíme se s její podobou příliš namáhat a stačí nám zobrazení pomocí příkazu **//TableForm** (viz výše).

In[103]:=
```
tab1 = Prepend[DruhyDleCetnosti, {"druh", "počet"}]
```

Out[103]=
```
{{druh, počet}, {okoun, 56}, {cejn, 35}, {plotice, 20},
 {štika, 17}, {koruška, 14}, {cejnek, 11}, {jelec, 6}}
```

In[104]:=
```
Tab1 = Grid[tab1,
  Frame → True, Dividers → {2 → True, 2 → True},
  Alignment → {{Left, Right}},
  Background → {None, {LightBlue}}]
```

Out[104]=

druh	počet
okoun	56
cejn	35
plotice	20
štika	17
koruška	14
cejnek	11
jelec	6

Z tabulky je ihned patrné, že nejvíce zástupců má v našich datech okoun, a to 56, druhý druh v pořadí podle četnosti je cejn s 35 zástupci. Naopak ze sedmi sledovaných druhů je v datech nejméně zastoupen jelec. Ten má jen 6 zástupců.

Všimněme si, jakých voleb příkazu **Grid[]** jsme využili. Nastavením hodnot *Frame* a *Dividers* určíme pozici svislých a vodorovných čar; zarovnání ve sloupcích (případně i řádcích) určíme pomocí nastavení hodnoty *Alignment*; pomocí *Background* jsme nastavili světle modré zvýraznění

záhlaví tabulky.

Tabulky a grafy v textu

Udělejme nyní krátkou odbočku a povězme si něco o „zvyklostech“ při psaní textu, jehož součástí jsou nejrůznější tabulky a grafy. Příkladem takového textu může být bakalářská či diplomová práce nebo článek v odborném časopise.

Nejdůležitějším pravidlem je, že každá tabulka a každý graf musí mít své číslo a musí být popsány. V praxi to může vypadat takto:

In[105]:=
```
Labeled[Tab1, "Tabulka 1: četnosti ryb dle druhů."]
```

Out[105]=

druh	počet
okoun	56
cejn	35
plotice	20
štika	17
koruška	14
cejnek	11
jelec	6

Tabulka 1: četnosti ryb dle druhů.

Jedná-li se o tabulku (anglicky *table*), uvedeme nejprve „Tabulka 1:“, „Tabulka 2:“, ... Případně můžeme použít i jinou podobu: „Tabulka č. 1“, „Tab. 1“, „Tab. č. 1“ a podobně. Jakmile si však vybereme jednu z těchto podob, musíme ji dodržovat v celém textu. Nemělo by se tedy stát, že se v textu vyskytnou „Tabulka 1“, „Tab. č. 2“ a „Tabulka č. 3“.

Jedná-li se o graf či jiný grafický výstup, uvedeme nejprve „Obrázek 1:“, „Obrázek 2:“, ... (v anglickém textu se používá slovo *figure*). Opět můžeme použít i zkrácenou podobu (Obr.), případně přidat zkratku „č.“ (číslo).

Tabulky a obrázky se číslují zvlášť. První tabulka má číslo 1 (i kdyby jí předcházel jeden či více obrázků); první obrázek má rovněž číslo 1 (bez ohledu na počet tabulek, které mu předcházejí). V textu tak například může být postupně „Tabulka 1“, „Tabulka 2“, „Obrázek 1“, „Tabulka 3“.

Popis (legenda) tabulky či grafu by měl být sice stručný, zároveň však je nezbytné, aby byl natolik podrobný, že čtenář bude schopen mu porozumět bez důkladného studování textu diplomové práce (případně článku).

Tabulka či graf jsou účelné jen tehdy, když jsou komentovány v textu. Uveďme tedy další pravidlo o tabulkách a grafech: „Na každou tabulku a graf musí být v textu odkaz pomocí jejich čísla. Na příslušném místě textu pak shrneme nejdůležitější závěry, které lze z tabulky či grafu učinit.“ Je dobré mít na paměti, že text by měl být čitelný a srozumitelný i bez prohlížení tabulek a grafů.

Umístění popisu tabulky (či grafu) bývá nejčastěji pod tabulkou (či pod grafem). Můžeme se však setkat i s umístěním nad tabulkou (nad grafem).

Píšeme-li bakalářskou či diplomovou práci, je dobré zjistit, jaké jsou „fakultní zvyklosti“ týkající se popisu tabulek a obrázků a jejich umístění. Většina kateder má dnes své webové stránky a na nich (obsahová, formální a procesní) pravidla pro bakalářské a diplomové práce. Zde se dozvíme, jak by měla vypadat formální stránka práce. Nezřídka je součástí takového dokumentu také vzor bakalářské či diplomové práce. Není-li si student jist výkladem některého z požadovaných pravidel, může toto pravidlo probrat při konzultaci s vedoucím své práce.

Poznamenejme, že podobné požadavky na formální stránku textu mají také odborné časopisy. Stejně jako se požadavky na podobu bakalářské či diplomové práce liší doslova katedru od kat-

edry, tak i požadavky na podobu článků se liší časopis od časopisu. Opět platí, že časopis své požadavky na formální podobu článků zveřejňuje na svých webových stránkách.

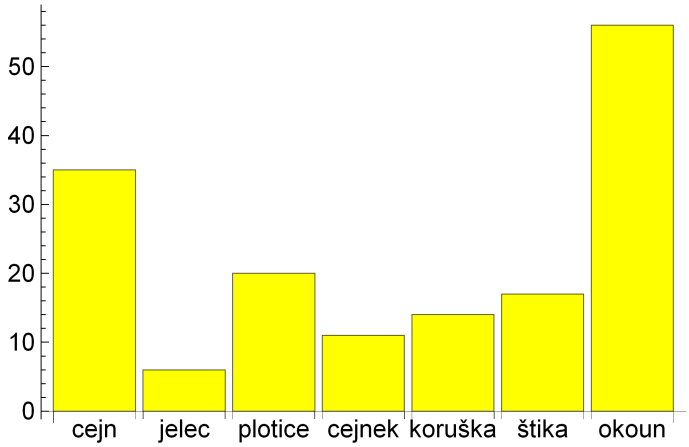
Výjimka potvrzuje pravidlo a tak autor této práce zcela vědomě u většiny tabulek a grafů, které čtenáři nabízí, neuvádí jejich čísla ani popisy. Je to proto, že cílem autora je ukázat, jak požadované tabulky či grafy sestrojit pomocí softwaru *Mathematica* a to tak, aby kód byl co nejjednodušší. Čtenář si snadno doplní příkaz **`Labeled[]`**, pomocí kterého by se k tabulkám či grafům přidával jejich popis.

Sloupkový diagram

Někdy je vhodné informaci obsaženou v četnostní tabulce prezentovat graficky. V takovém případě použijeme tzv. sloupkový diagram (používá se též výraz sloupcový diagram; anglicky bar chart):

In[106]:=
```
BarChart[CetnostDruhu[[All, 2]], ChartLabels -> CetnostDruhu[[All, 1]],
 BaseStyle → {FontSize → 13}, ChartStyle → Yellow]
```

Out[106]=



Z grafu lze vpodstatě vyčíst podobnou informaci o počtu zástupců jako z četnostní tabulky – vidíme, že nejvíce zástupců má v našich datech okoun, druhý druh v pořadí podle četnosti je cejn, nejméně zástupců má jelec. Přesné počty zástupců bychom z grafu (v jeho tištěné podobě) četli těžko, nicméně porovnáním výšky sloupce s hodnotami na y-ové ose vidíme, že okounů je více než 50, cejnů mezi 30 a 40, jelců méně než 10... Pokud pracujeme se softwarem *Mathematica,* tedy ne jen s tištěnou verzí, můžeme přesný počet zástupců určitého druhu zjistit umístěním kurzoru na odpovídající sloupec v grafu (např. pomocí myši).

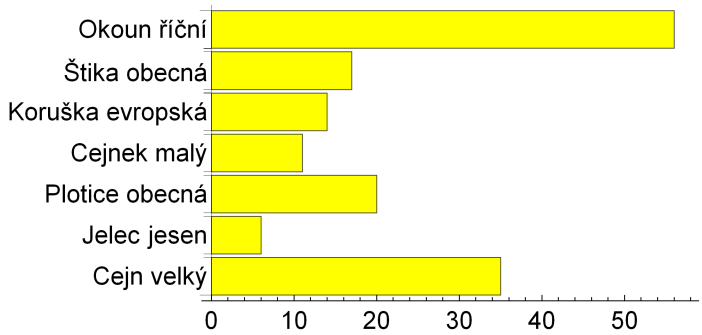
Vraťme se ještě k výše uvedenému příkazu **`BarChart[]`** a okomentujme postupně volby, které jsme provedli. Nejprve jsme zadali, jaké hodnoty se mají vykreslovat. Tyto hodnoty jsou uloženy ve vektoru *CetnostDruhu[[All,2]]*. Poté jsme pomocí *ChartLabels→* zadali názvy jednotlivých sloupců. Ty jsou uloženy ve vektoru *CetnostDruhu[[All,1]]*. Pomocí *BaseStyle→* jsme upravili velikost písma a pomocí *ChartStyle→* barvu sloupců. Poznamenejme, že pokud bychom velikost písma nezadali, byly by popisky značně menší. Tištěná verze textu by pak mohla být hůře čitelná.

Dodejme, že někdy můžeme tentýž graf zobrazit „naležato“. Výhodné je to zejména tehdy, když jsou popisky jednotlivých sloupců příliš dlouhé:

In[107]:=
```
NazvyCesky = {"Cejn velký", "Jelec jesen", "Plotice obecná",
   "Cejnek malý", "Koruška evropská", "Štika obecná", "Okoun říční"};
```

In[108]:=
```
BarChart[CetnostDruhu[[All, 2]], ChartLabels → NazvyCesky,
 BaseStyle → {FontSize → 13}, ChartStyle → Yellow, BarOrigin → Left]
```

Out[108]=



Volbu zobrazení grafu „naležato“ jsme zadali volbou *BarOrigin→Left*.

Tabulka relativních četností

Kromě absolutních četností, tedy počtů zástupců jednotlivých druhů v našich datech, může být pro nás zajímavá i tzv. relativní četnost, tedy údaj o tom, jakou část z celku (všech sledovaných ryb) tvoří zástupci jednotlivých druhů. Naším úkolem bude upravit výše zkonstruovanou tabulku absolutních četností tak, že k ní přidáme ještě třetí sloupec, v němž bude uvedeno procentuální zastoupení jednotlivých druhů v studovaném souboru. Výpočet relativních četností je snadný:

In[109]:=
```
procent = 100 * DruhyDleCetnosti[[All, 2]] / n
```

Out[109]=
$$\left\{\frac{5600}{159}, \frac{3500}{159}, \frac{2000}{159}, \frac{1700}{159}, \frac{1400}{159}, \frac{1100}{159}, \frac{200}{53}\right\}$$

Menší problém může nastat, když chceme všechny hodnoty zaokrouhlit na jedno desetinné místo. Uvedeme nejprve dva způsoby, které někdy mohou být užitečné, ale v našem případě „nefungují“:

In[110]:=
```
N[procent, 3]
```

Out[110]=
```
{35.2, 22.0, 12.6, 10.7, 8.81, 6.92, 3.77}
```

In[111]:=
```
Round[procent, 0.1]
```

Out[111]=
```
{35.2, 22., 12.6, 10.7, 8.8, 6.9, 3.8}
```

Zatímco funkce **N[]** pracuje s platnými číslicemi a počet desetinných míst se tak liší podle toho, jestli celá část uvedeného čísla je menší než 10, funkce **Round[]** sice zaokrouhluje na jedno desetinné místo (nejbližší celočíselný násobek čísla 0.1), nicméně druhou hodnotu 22,0 zobrazí ve formě 22. – nulu za desetinnou čárkou resp. tečkou nezobrazuje. To je problém, pokud chceme čísla použít v tabulce, která má „pěkně vypadat“. Tento problém můžeme vyřešit pomocí příkazu **NumberForm[]** upravujícího podobu zobrazení čísel. Více k formě zobrazování čísel se lze dočíst v nápovědě pod heslem guide/DisplayOfNumbers.

In[112]:=
```
NumberForm[N[procent], {Infinity, 1}]
```

Out[112]//NumberForm=
```
{35.2, 22.0, 12.6, 10.7, 8.8, 6.9, 3.8}
```

Upravme tedy vektor DruhyDleCetnosti tak, že ke všem vektorům, které jej tvoří (mají dva prvky), přidáme ještě jeden prvek – relativní četnost odpovídající absolutní četnosti v tomto vektoru uvedené:

In[113]:=
```
DruhyCetnostVic =
 DruhyDleCetnosti /. {a_, b_} → {a, b, NumberForm[N[100 * b / n], {Infinity, 1}] }
```

Out[113]=
```
{{okoun, 56, 35.2}, {cejn, 35, 22.0}, {plotice, 20, 12.6},
 {štika, 17, 10.7}, {koruška, 14, 8.8}, {cejnek, 11, 6.9}, {jelec, 6, 3.8}}
```

Tabulku ještě upravme – přidejme řádek sloupcových součtů, do třetího sloupce vepišme symboly „%“ a přidejme řádek obsahující hlavičku tabulky:

In[114]:=
```
Tab2 = Append[DruhyCetnostVic, {"celkem", n, NumberForm[N[100], {Infinity, 1}]}];
Tab2 = Tab2 /. {a_, b_, c_} :> {a, b, ToString[c] <> " %"};
Tab2 = Prepend[Tab2, {"druh", "počet", "procent"}];
```

Tabulku zobrazíme opět pomocí příkazu **Grid[]**:

In[116]:=
```
Grid[Tab2,
 Frame → True, Dividers → {2 → True, {2 → True, -2 → True}},
 Alignment → {{Left, Right, Right}},
 Background → {None, {LightBlue}},
 Spacings → {{1, 1, 3}, Automatic}]
```

Out[116]=

druh	počet	procent
okoun	56	35.2 %
cejn	35	22.0 %
plotice	20	12.6 %
štika	17	10.7 %
koruška	14	8.8 %
cejnek	11	6.9 %
jelec	6	3.8 %
celkem	159	100.0 %

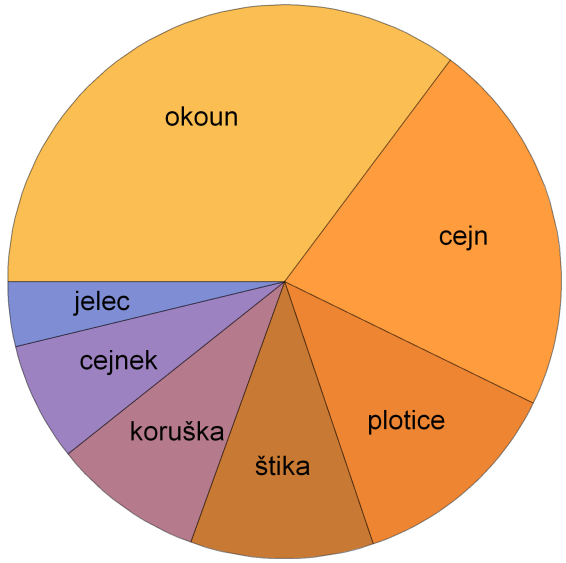
S takto upravenou tabulkou již můžeme být spokojeni.

Koláčový graf

Procentuální zastoupení jednotlivých druhů ryb, tedy jejich reativní četnosti, můžeme graficky znázornit pomocí tzv. koláčového grafu (anglicky pie chart):

In[117]:=
```
PieChart[DruhyCetnostVic[[All, 2]], ChartLabels → DruhyCetnostVic[[All, 1]],
 BaseStyle → {FontSize → 14}]
```

Out[117]=



Z tohoto grafu je například ihned patrné, že zástupci dvou nejvíce zastoupených druhů – okouna a cejna – představují dohromady více než polovinu ryb v datovém souboru.

Koláčový graf vytvořený softwarem *Mathematica* je (stejně jako sloupkový diagram) „interaktivní“. Umístěním kurzoru na příslušnou výseč můžeme zobrazit hodnotu, kterou tato výseč reprezentuje. Kliknutím na některou z výsečí tuto výseč „odsadíme“ – zvýrazníme.

Na závěr ještě poznamenejme, že sloupkový diagram (BarChart) i koláčový graf (PieChart) mohou být zobrazeny i „prostorově“ pomocí příkazů **`BarChart3D[]`** a **`PieChart3D[]`**. Více informací lze nalézt v Helpu pod heslem guide/ChartingAndInformationVisualization. Autorova zkušenost říká, že při vizualizaci informace o jediném statistickém znaku jsou „efektní“ 3D grafy používány spíše ve sféře „komerční“, zatímco sféra akademická vnímá podobné 3D-efekty jako nadbytečné a ocení spíše jednoduchost „obyčejných“ grafů. 3D-grafika se nicméně může velmi dobře uplatnit při vizualizaci informace o více znacích najednou.

Shrnutí

Kvalitativní znaky (jako například druh ryby či pohlaví ryby) popisujeme:

- číselně pomocí tabulky absolutních (případně i relativních) četností,
- graficky pomocí sloupkového diagramu či koláčového grafu.

Užitečné příkazy softwaru *Mathematica*:

`Tally[], BarChart[], PieChart[]`

2.2 Kvantitativní znak

Připomeňme, že datový soubor, který v tomto textu zkoumáme, obsahuje údaje o 159 rybách. U každé ryby přitom máme k dispozici záznamy o 8 různých znacích (vlastnostech). V předešlé kapitolce byla řeč o kvalitativních znacích, tedy znacích, jejichž hodnota vyjadřuje příslušnost k určité skupině (kategorii) objektů. Kvalitativními znaky v našem souboru jsou „druh ryby“ a „pohlaví

ryby“. Všechny ostatní sledované znaky – znaky týkající se hmotnosti, délky, výšky a šířky ryb – jsou kvantitativní. Typickou vlastností kvantitativních znaků je jejich číselná povaha.

V této kapitolce si vyzkoušíme popis kvantitativního znaku. Uvedeme zde dva příklady. Nejprve budeme zkoumat znak, jehož rozdělení může být považováno za normální (Gaussovo). Poté se budeme zabývat popisem kvantitativního znaku, jehož rozdělení normální není.

Prvním příkladem je veličina „celková délka cejnů“. Jako příklad by nám mohla posloužit veličina „celková délka ryby“ udávaná v centimetrech. Protože je však náš soubor nehomogenní – obsahuje údaje o sedmi různých druzích ryb – bude lépe, když nebudeme míchat „hrušky s jabkama“ či „cejny se štikami“ a vybereme jen údaje o celkové délce ryb jednoho druhu, například cejna:

In[118]:=
```
CejniDelka = Ryby[[1, All, 6]]
```

Out[118]=
```
{30., 31.2, 31.1, 33.5, 34., 34.7, 34.5, 35., 35.1, 36.2, 36.2,
 36.2, 36.4, 37.3, 37.2, 37.2, 38.3, 38.5, 38.6, 38.7, 39.5, 39.2, 39.7,
 40.6, 40.5, 40.9, 40.6, 41.5, 41.6, 42.6, 44.1, 44., 45.3, 45.9, 46.5}
```

K příkazu CejniDelka = Ryby[[1,All,6]] poznamenejme, že číslo 1 znamená, že pracujeme s druhem číslo 1 (cejn), All znamená, že chceme všechny řádky (všechny ryby) odpovídající druhu číslo 1, a 6 znamená, že pracujeme se šestým sloupcem, v němž jsou právě údaje o celkové délce ryb.

In[119]:=
```
n1 = Length[CejniDelka]
```

Out[119]=
```
35
```

Všimněme si, že hodnoty v datovém souboru jsou jaksi uspořádány. Při bližším zkoumání však zjistíme, že uspořádání není „dokonalé“, např. hodnota 34.7 předchází hodnotě 34.5. Ryby tedy zřejmě někdo seřadil, ale kritériem řazení nebyla celková délka (možná, že jde o řazení podle váhy, která je prvním číselným údajem o každé rybě). Zde tedy už na první pohled vidíme, že se hodnoty pohybují v rozmezí 30 až 46,5 cm. U většiny datových souborů však takové štěstí, že by hodnoty byly seřazeny, nemáme a vyznat se ve změti neuspořádaných hodnot bývá obtížnější. Posuďte sami:

In[120]:=
```
SeedRandom[1]
CejniDelka[[RandomSample[Range[n1]]]]
```

Out[121]=
```
{39.5, 31.2, 31.1, 33.5, 30., 39.2, 44.1, 38.3, 40.6, 37.2, 44.,
 35.1, 38.7, 34.7, 38.6, 41.5, 36.4, 41.6, 34., 36.2, 42.6, 39.7, 40.5,
 38.5, 45.9, 46.5, 40.9, 36.2, 37.3, 37.2, 36.2, 40.6, 45.3, 35., 34.5}
```

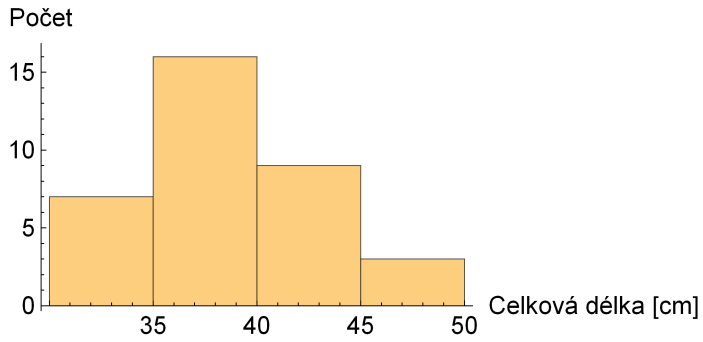
Tento příkaz vytvořil (pseudo)náhodnou permutaci čísel uvedených ve vektoru CejniDelka. Rozebírat jej zde nebudeme, protože nás zajímá spíše výsledek, než příkaz samotný.

Histogram

Představu o rozdělení hodnot znaku „celková délka“ cejnů si uděláme pomocí histogramu:

In[122]:=
```
Histogram[CejniDelka, BaseStyle → {FontSize → 13},
 AxesLabel → {"Celková délka [cm]", "Počet"}]
```

Out[122]=



Histogram je „interaktivní“ – pohybem kurzoru např. pomocí myši můžeme zobrazit vykreslené počty.

Z histogramu vidíme, že nejvíce (16) cejnů má celkovou délku mezi 35 a 40 cm. V kategorii „nejmenších“, tj. s celkovou délkou mezi 30 a 35 cm je 7 ryb, o dva zástupce více (9) má kategorie 40 až 45 cm. V kategorii „největších“, tj. 45 až 50 cm, jsou 3 zástupci.

Možná nás napadne otázka, do které kategorie patří ryby, které mají přesně 35 (resp. 40, 45) cm? Můžeme samozřejmě pátrat v nápovědě k příkazu **`Histogram[]`**, ale odpoveď můžeme najít i pouhým prozkoumáním vektoru CejniDelka. Především zjistíme, že v datech není žádná ryba, jejíž celková délka by byla přesně 40 nebo 45 cm a jen jedna ryba s celkovou délkou 35 cm. Tato ryba je osmá nejmenší. Jelikož kategorie nejmenších (30 až 35 cm) obsahuje jen 7 ryb, je ryba velikosti 35 cm již zřejmě zařazena do kategorie 35 až 40 cm.

Rozdělení můžeme hodnotit z hlediska symetrie (zde žádná vážnější asymetrie vidět není), odlehlých pozorování (v našem případě zřejmě nejsou přítomna) či unimodality (zdá se, že je zde jen jediný „vrcholek – modus“). Zřejmě nejpoužívanějším symetrickým unimodálním spojitým rozdělením je normální (Gaussovo) rozdělení, viz

In[123]:=
```
Normal distribution
```

(výstup není z důvodu úspory místa zobrazen)

Pokud histogram nepopírá symetrii a unimodalitu rozdělení sledované veličiny, můžeme vyšetřovat, zda námi analyzované hodnoty můžeme považovat za realizace normálně rozdělené náhodné veličiny. V následujícím textu si řekneme, jak posuzovat normalitu na základě kvantilového grafu a testů normality. Otázku „Proč nás normalita rozdělení hodnot nějaké veličiny vůbec zajímá?“ se pokusíme zodpovědět hned poté.

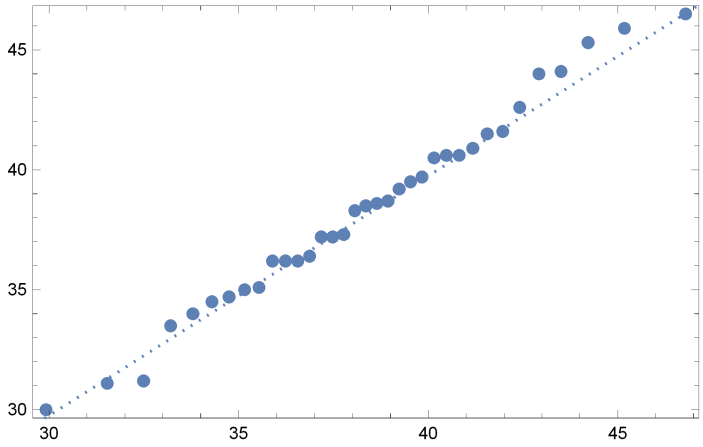
Jen pro pořádek doplňme, že vykazuje-li rozdělení dat asymetrii nebo multimodalitu, nemá smysl o normálním rozdělení uvažovat. Příkladem nesymetrického rozdělení je lognormální či exponenciální rozdělení; multimodální rozdělení může být směsí vícero unimodálních rozdělení.

Posouzení normality

Jak již bylo uvedeno, je prvním krokem při posuzování normality rozdělení určité veličiny hodnocení histogramu z hlediska symetrie a unimodality. V našem případě můžeme říci, že celková délka cejnů může být pokládána za symetrickou unimodální veličinu. K detailnějšímu posouzení normality můžeme použít tzv. kvantilový graf:

In[124]:=
```
QuantilePlot[CejniDelka]
```

Out[124]=



y-ové souřadnice bodů v tomto grafu jsou kvantily rozdělení datového vektoru CejniDelka, tedy takzvané empirické kvantily; x-ové souřadnice jsou odpovídající kvantily normálního rozdělení. Pokud je předpoklad normality splněn, měly by si být hodnoty na y-ové a na x-ové ose blízké, tedy body by měly ležet blízko diagonální přímky.

V našem případě jsou body v grafu blízko diagonální přímky, empiricky zjištěné kvantily tak odpovídají kvantilům normálního rozdělení. Tento obrázek tak hypotézu normality rozdělení veličiny „celková délka cejna“ podporuje. Jak může vypadat kvantilový graf při nesplnění podmínky normality rozdělení ukážeme v části „Popis kvantitativního znaku v případě nenormality“.

O tom, zda jsme při posuzování normality „něco nepřehlédli“ se můžeme přesvědčit některým z testů normality, například testem známým pod názvem „Shapirův-Wilkův test“:

In[125]:=
```
ShapiroWilkTest[CejniDelka]
```

Out[125]=
```
0.876321
```

Hodnota, kterou nám *Mathematica* ukázala, je p-hodnota testu normality. Jelikož je větší než 0,05, nezamítáme nulovou hypotézu (hypotézu normality rozdělení). Rozdělení délky cejnů tedy budeme pokládat za normální. Pro čtenáře (či spíše uživatele), který si ještě neosvojil základy testování hypotéz, nabízí *Mathematica* výstup v podobě slovní informace:

In[126]:=
```
ShapiroWilkTest[CejniDelka, "TestConclusion"]
```

Out[126]=
```
The null hypothesis that
 the data is distributed according to the NormalDistribution[x̄, ȳ]
 is not rejected at the 5 percent level based on the Shapiro-Wilk test.
```

Tento výstup můžeme doslovně (tedy poněkud nehezky) přeložit takto: „Nulová hypotéza, že data mají normální rozdělení, není zamítnuta na 5 procentní hladině významnosti na základě Shapirova-Wilkova testu.“ Česky bychom tedy řekli, že při použití Shapirova-Wilkova testu nezamítáme na 5 procentní hladině významnosti hypotézu normality rozdělení.

Shapirův-Wilkův test není jediným možným testem k testování normality rozdělení. *Mathematica* nabízí řadu dalších možností: „AndersonDarling“, „CramerVonMises“, „JarqueBeraALM“, „KolmogorovSmirnov“, „Kuiper“, „PearsonChiSquare“, „WatsonUSquare“. Detailnější informace lze najít v nápovědě k příkazu **`DistributionFitTest[]`**.

Debata o normálním rozdělení – pro a proti

Mám rád normální rozdělení,
má klidně tři sta momentů,
nechystá žádný překvapení

v ňákým datovým segmentu.
Je normální i pro Japonce,
nad Prahou i nad Tokiem...

Zdeněk Fabián

Pokusme se nyní zodpovědět klíčovou otázku: *Proč nás normalita rozdělení hodnot nějaké veličiny vůbec zajímá?* Odpověď je poněkud obšírnější.

Pravdou je, že normální rozdělení má zvláštní, řekli bychom privilegované, postavení mezi všemi rozděleními. Tato privilegovanost zřejmě vychází ze skutečnosti, kterou bychom mírně nepřesně mohli vyjádřit tvrzením, že „průměrujeme-li hodně hodnot, je rozdělení artmetického průměru přibližně normální (a to bez ohledu na rozdělení průměrovaných hodnot)". Toto tvrzení je známé jako *centrální limitní věta*. (Přesné znění včetně předpokladů, které však pro většinu praktických aplikací nejsou nikterak omezující, zde uvádět nebudeme. Lze jej nalézt ve většině učebnic základních kurzů pravděpodobnosti, viz například Věta 4.6 v učebnici [3].) Normálnímu rozdělení, jakožto limitnímu rozdělení aritmetického průměru, byla dlouhodobě věnována značná pozornost. Mnoho statistických metod (jako je například t-test) se opírá o předpoklad normality zkoumané veličiny. Poznamenejme, že v některých oblastech lidského zkoumání je předpoklad normality celkem oprávněný (například chyby některých měření fyzikálních veličin mohou být pokládány za normálně rozdělené), v jiných oblastech se však s normálně rozdělenými znaky setkáme jen zcela výjimečně.

Odpověď na otázku proč nás zajímá normalita rozdělení sledované veličiny, však můžeme s trochou nadhledu nalézt i v písňovém textu Zdeňka Fabiána. Víme-li, že rozdělení je normální, stačí k úplnému popisu tohoto rozdělení určit hodnotu jeho dvou parametrů
- střední hodnoty (prvního obecného momentu), tradičně značené symbolem μ [mí]
- a rozptylu (druhého centrálního momentu), tradičně značeného symbolem σ^2 [sigma].

A pak už nás „nečeká žádný překvapení". Atraktivita normálního rozdělení tedy tkví mimo jiné v tom, že se popis rozdělení zjednoduší na udání dvou číselných charakteristik tohoto rozdělení – střední hodnoty a rozptylu. Je tedy velmi snadný.

Říká se, že v jednoduchosti je síla, avšak někdy je přílišné zjednodušování na škodu. O tom, jak nemilá překvapení nás mohou čekat, když se tohoto zjednodušení dopustíme, pojednává kniha Nassima Taleba *Černá Labuť* [2]. Taleb hovoří především o nevhodném použití předpokladu normality při analýze dat z oblasti finančních trhů. Budou-li ceny akcií po nějaký čas „normálně" oscilovat kolem určité hodnoty, neznamená to, že hned další den nemůže nastat dramatický propad cen těchto akcií, tedy situace, kterou bychom při analýze „historických dat" za předpokladu jejich normality považovali za téměř vyloučenou.

V kapitole nazvané *Velký intelektuální podvod jménem Gaussova křivka* Taleb zmiňuje, že portrét Carla Friedricha Gausse a vyobrazení jeho slavné křivky bylo možno až do roku 2001, kdy německou marku vystřídalo euro, nalézt na desetimarkové bankovce. Tuto skutečnost komentuje slovy: „*Ironií osudu je, že Gaussova křivka je tou poslední věcí, kterou si lze s německou markou spojovat: kurs říšské marky (jak se tehdy nazývala) se ve dvacátých letech během několika let vyhoupl ze čtyř až na na čtyři bilióny marek za dolar, což nám bezpochyby napoví, že používat zvonovou křivku k popisu nahodilostí ve fluktuacích měnových kurzů postrádá smysl.[...] Ještě neuvěřitelnější je, že Gaussovu křivku používají centrální bankéři a regulátoři, muži pověstní tmavými obleky a nudnými výroky o měnách, jako nástroj řízení rizik*."

Má tedy vůbec smysl zabývat se normalitou rozdělení určité veličiny? Dosti rozšířený je názor, že ačkoliv většina veličin, které sledujeme, nemá přesně normální rozdělení, je hodně případů, kdy je skutečné rozdělení sledované veličiny dosti podobné normálnímu rozdělení, aby bylo zjednodušení v podobě předpokladu normality ospravedlnitelné. Je dobré mít na paměti, že statistika pracuje s modely, tedy zjednodušeními často komplikované reality. Statistik se při popisu reality musí vyrovnat s úkoly, které „jdou proti sobě"; má totiž za úkol popsat realitu jednoduše (pokud možno co

nejjednodušeji), ale přitom věrně, tedy ne příliš zjednodušeně. Univerzální návod, který by říkal, co je příliš zjednodušující a co je naopak zbytěčně složité a dalo by se zjednodušit, přitom neexistuje. Taleb velmi trefně hovoří v tom smyslu, že bychom si přílišné zjednodušení neměli dovolit, pokud by takovéto zjednodušení mohlo mít „nedozírné následky“.

Základní číselné charakteristiky rozdělení

Rozdělení kvantitativní veličiny zjednodušeně popisujeme uvedením tzv. číselných charakteristik rozdělení. Jde o tzv. parametry polohy (např. střední hodnota, medián), parametry variability (např. rozptyl, směrodatná odchylka, mezikvartilové rozpětí, rozpětí), parametry šikmosti (koeficient šikmosti) a parametry špičatosti (koeficient špičatosti). Z dat jsme schopni tyto charakteristiky odhadnout.

1) CHARAKTERISTIKY POLOHY

Parametrem polohy je například střední hodnota. Tu obvykle značíme E*X*, případně μ. Odhadem střední hodnoty může být aritmetický průměr:

In[127]:=
```
prumer = Mean[CejniDelka]
```

Out[127]=
```
38.3543
```

Aritmetický průměr, značený často symbolem $\overline{X_n}$, je součtem všech hodnot děleným počtem hodnot, tedy:

$$\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

In[128]:=
```
Total[CejniDelka] / Length[CejniDelka]
```

Out[128]=
```
38.3543
```

Zapisujeme $\hat{\mu} \doteq 38{,}4$. Stříška nad symbolem μ značí, že jde o odhad parametru μ (střední hodnoty). V základních kurzech statistiky jsou studenti nabádáni, aby striktně rozlišovali, co je (reálný) parametr, co náhodná veličina a co realizace náhodné veličiny, učí se rozlišovat symboly μ a $\hat{\mu}$:

- μ značí střední hodnotu. Střední hodnota je nějaké reálné číslo.
- $\hat{\mu}$ značí odhad střední hodnoty. Je to náhodná veličina. Stejný symbol se však (bohužel) používá i pro realizaci této náhodné veličiny, jíž je reálné číslo, viz náš zápis $\hat{\mu} \doteq 38{,}4$.

Možná se ptáte, proč jsme provedli zaokrouhlení na jedno desetinné místo a přišli tak o „cennou“ informaci v podobě dalších tří desetinných míst. Odpověď si žádá poněkud delší výklad.

Především nezapomínejme, že námi uváděná hodnota – aritmetický průměr – je odhadem střední hodnoty, nikoliv střední hodnotou samotnou. Aritmetický průměr je náhodná veličina, neboť průměruje náhodné veličiny, a hodnota 38,4 je realizací této náhodné veličiny. Každá (nedegenerovaná) náhodná veličina má přitom určitou variabilitu. Laicky řečeno, každý odhad je spojen s určitou nejistotou. Student má někdy tendenci, ve snaze poskytnout „co nejpřesnější“ informaci, uvést i desetinná místa, která jsou vzhledem k velikosti nejistoty v odhadu naprosto irelevantní. Absurdnost takového počínání trefně vystihli autoři hry Posel z Liptákova:

... Tak zrovna doc. Vozáb řešil například otázku, kdy vlastně Cimrman do Liptákova přišel, měřením rozpadu radiokativního uhlíku v organické nečistotě na podrážkách Cimrmanových bot. Zjistil, že – pokud Cimrman v těchto botách do Liptákova přišel – se tak stalo na podzim roku 1906 plus minus 200 let. ...

Informace, že Cimrman přišel do Liptákova na podzim roku 1906 se zdá být užitečná, ale jakmile je odhalena nejistota odhadu (plus minus 200 let), stává se „přesnost“ této informace naprosto směšnou.

Než budeme vyčíslovat nejistotu v našem odhadu, všimněme si ještě jedné podstatné okolnosti. Ve výše uvedeném příkladu je uvedena velikost nejistoty „plus minus 200 let“. Toto je nejistota v rámci

jakéhosi modelu. Ten má ovšem určité předpoklady, z nichž možná ten nejdůležitější je, že byly analyzovány podrážky bot, ve kterých Cimrman opravdu do Liptákova přišel. Jak ale napovídá uvedený úryvek, i tento předpoklad je svým způsobem nejistý. Nejistot, se kterými se při analýze dat setkáváme, je tedy vícero.

Pokud stejně jako docent Vozáb zveřejníme nejistotu svého odhadu, zveřejníme tzv. intervalový odhad (střední hodnoty). Ten má pro normálně rozdělenou veličinu podobu:

$$(\hat{\mu} - \frac{\hat{\sigma}}{\sqrt{n}} t_{n-1,1-\frac{\alpha}{2}}, \hat{\mu} + \frac{\hat{\sigma}}{\sqrt{n}} t_{n-1,1-\frac{\alpha}{2}}).$$

V tomto vzorci symbol $\hat{\mu}$ značí aritmetický průměr, $\hat{\sigma}$ výběrovou směrodatnou odchylku (viz charakteristiky variability) a *n* počet pozorování. Symbol $t_{n-1,1-\frac{\alpha}{2}}$ je použit pro $(1 - \frac{\alpha}{2})$-kvantil Studentova t-rozdělení o *n-1* stupních volnosti. Pokud vám pojmy jako kvantil či Studentovo t-rozdělení mnoho neříkají, představte si pod tímto symbolem nějaké číslo, jehož hodnotu můžeme zjistit pomocí počítače. V softwaru *Mathematica* použijeme příkazu **`InverseCDF[]`**.

```
In[129]:= Kvantil = InverseCDF[StudentTDistribution[n1 - 1], 0.975]
Out[129]= 2.03224
```

Chceme získat interval, v němž se skutečná střední hodnota nachází s velkou pravděpodobností. Tuto pravděpodobnost označujeme obecně symbolem 1-α, přičemž obvykle se uvažuje pravděpodobnost 0,95, tedy α = 0,05. Proto jsme ve výše uvedeném příkazu uvedli hodnotu 1-α/2 = 0,975.

Vypočtěme ještě hodnotu $\hat{\sigma}$:

```
In[130]:= s = StandardDeviation[CejniDelka]
Out[130]= 4.15787
```

Intervalový odhad střední hodnoty má tedy následující podobu:

```
In[131]:= prumer + {-1, 1} * Kvantil * s / Sqrt[n1]
Out[131]= {36.926, 39.7826}
```

Budeme-li tedy chtít říci něco o velikosti cejnů, řekneme, že průměrná velikost cejnů je 37 až 40 centimetrů. Pokud chceme být přesnější, například v psaném textu, uvedeme, že střední hodnota velikosti cejnů byla odhadnuta na 38,4 cm s 95% konfidenčním intervalem (37,0; 39,7) cm. Vzhledem k velikosti nejistoty v odhadu nemá tedy v námi uvažovaném případě smysl uvádět více než jedno desetinné místo v odhadu střední hodnoty.

Pokud bychom způsob výpočtu intervalového odhadu zapomněli, nabízí *Mathematica* možnost použít příkazu **`MeanCI[]`** z knihovny HypothesisTesting`:

```
In[132]:= << HypothesisTesting`
In[133]:= MeanCI[CejniDelka]
Out[133]= {36.926, 39.7826}
```

Jde opravdu o stejný výsledek, jaký jsme získali již dříve.

Jiným parametrem polohy je medián. Ten odhadujeme tzv. výběrovým mediánem – hodnotou „uprostřed“, jsou-li hodnoty řazeny od nejmenší po největší. Pro symetrická rozdělení je medián roven střední hodnotě a tudíž výběrový medián je vlastně také odhadem střední hodnoty.

```
In[134]:= Median[CejniDelka]
Out[134]= 38.5
```

Všimněme si, že tato hodnota sice není přesně rovna aritmetickému průměru (38,3543), ale je této

hodotě dosti blízko. To nám znovu připomene skutečnost, že jde o dva různé odhady téhož (střední hodnoty symetrického rozdělení).

Výběrový medián je, stejně jako aritmetický průměr, náhodnou veličinou. Když jsme odhadovali střední hodnotu aritmetickým průměrem, ukazovali jsme si, jak najít rovněž příslušný intervalový odhad. Také při odhadu střední hodnoty pomocí výběrového mediánu bychom mohli najít příslušný intervalový odhad. Jeho konstrukce však není jednoduchá a v základních kurzech se většinou nevyučuje.

Nabízí se otázka, který z odhadů střední hodnoty normálního rozdělení si vybrat a používat: aritmetický průměr nebo výběrový medián? Aritmetický průměr je v jistém smyslu optimálním odhadem střední hodnoty normálního rozdělení (má nejmenší rozptyl mezi všemi nestrannými odhady střední hodnoty normálního rozdělení), je však citlivý na odlehlá pozorování – jediná „odlehlá" hodnota může hodnotu aritmetického průměru značně ovlivnit. Výběrový medián tuto špatnou vlastnost nemá – je robustní vůči odlehlým pozorováním. Oba způsoby odhadu tedy mají své výhody. Nemusíme-li se obávat odlehlých pozorování, použijme průměr, pokud však chceme vyloučit vliv případných odlehlých pozorování, použijme výběrový medián. V případě celkové délky cejnů je aritmetický průměr vhodnou volbou.

2) CHARAKTERISTIKY VARIABILITY

Střední hodnota (případně medián), to je jen jedno číslo. „Různí cejni jsou však různě velcí", tedy hodnoty sledované veličiny „kolísají kolem střední hodnoty". Jak moc tyto hodnoty kolísají popisujeme pomocí některé z charakteristik variability. Uvedeme si dvě úzce spjaté charakteristiky variability – rozptyl a směrodatnou odchylku, a ještě jednu charakteristiku – mezikvartilové rozpětí.

Rozptyl (anglicky variance) je druhým centrálním momentem: $\mathrm{Var}X = \mathrm{E}(X - \mathrm{E}X)^2$. My jej můžeme odhadnout výběrovým rozptylem.

In[135]:=
```
s2 = Variance[CejniDelka]
```

Out[135]=
```
17.2878
```

Připomeňme, že výběrový rozptyl je dán vzorcem

$$\frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X_n}\right)^2,$$

kde symbol $\overline{X_n}$ označuje aritmetický průměr. Připomeňme, že „záhadné" odečtení jedničky ve jmenovateli zlomku $\frac{1}{n-1}$ je zdůvodněno tím, že takto definovaný výběrový rozptyl je *nestranným* odhadem rozptylu. O tom, že *Mathematica* opravdu používá tuto definici výběrového rozptylu se můžete přesvědčit buďto v nápovědě příkazu **Variance[]**, nebo přímým výpočtem:

In[136]:=
```
Total[(CejniDelka - Mean[CejniDelka])^2] / (Length[CejniDelka] - 1)
```

Out[136]=
```
17.2878
```

Směrodatná odchylka (anglicky Standard Deviation) není nic jiného než odmocnina z rozptylu. Odmocněním výběrového rozptylu získáme výběrovou směrodatnou odchylku:

In[137]:=
```
StandardDeviation[CejniDelka]
```

Out[137]=
```
4.15787
```

In[138]:=
```
s2^(1 / 2)
```

Out[138]=
```
4.15787
```

Jelikož se směrodatná odchylka tradičně značí řeckým písmenem σ (sigma) (a rozptyl pak σ^2), můžeme zapsat $\hat{\sigma} = 4{,}16$. Stříška nad symbolem sigma značí, že jde o odhad, nikoliv o teoretickou

(neznámou) hodnotu směrodatné odchylky.

Vrátíme-li se k části věnované charakteristikám polohy, zjistíme, že už jsme s odhadem směrodatné odchylky pracovali. Bylo to v souvislosti s konstrukcí intervalového odhadu pro střední hodnotu. Je to pochopitelné – čím větší variabilita sledované veličiny, tím větší nejistota (při stejném počtu pozorování) v odhadu střední hodnoty. Tato skutečnost je vyjádřena známým vztahem:

Je-li $X_1, \ldots, X_n$ *náhodný výběr z rozdělení* $N(\mu,\sigma^2)$, *pak aritmetický průměr* $\overline{X_n}$ *má rozdělení* $N\left(\mu, \frac{\sigma^2}{n}\right)$.

Vraťme se ale zpět k samotné hodnotě $\hat{\sigma}$ = 4,16. Má toto číslo nějakou interpretaci? Všimněme si, že z hlediska fyzikálních jednotek má směrodatná odchylka stejné jednotky jako sledovaná veličina. V našem případě jde o centimetry. Hodnota směrodatné odchylky tak bude snáze interpretovatelná než hodnota rozptylu, jehož jednotky jsou druhou mocninou původních jednotek (rozptyl má v našem případě jednotku cm^2).

Na základě směrodatné odchylky si uděláme představu o rozsahu hodnot sledované veličiny. Víme totiž, že se v případě znaku s normálním rozdělením naprostá většina hodnot (asi 99,7 %) od střední hodnoty neliší o více než trojnásobek směrodatné odchylky, velká většina (asi 95 %) se neliší od střední hodnoty o více než dvojnásobek směrodatné odchylky. Velmi hrubý odhad nám říká: průměr je 38, směrodatná odchylka 4, tedy 95 % cejnů (dospělých) má celkovou délku mezi 30 a 46 centimetry. K těmto číslům jsme dospěli odečtením respektive přičtením dvojnásobku hodnoty 4 k průměru (hodnotě 38).

Budeme-li chtít postupovat „rigorózně“, použijeme kvantil normálního rozdělení:

In[139]:=
```
KvantilN = InverseCDF[NormalDistribution[], 0.975]
```

Out[139]=
```
1.95996
```

In[140]:=
```
prumer + {-1, 1} * KvantilN * s
```

Out[140]=
```
{30.205, 46.5036}
```

Délka 95 % cejnů (tedy všech, kromě těch největších a nejmenších) by neměla být méně než 30,2 cm a ne více než 46,5 cm. V našem souboru byla měření 35 cejnů.

In[141]:=
```
0.95 * 35
```

Out[141]=
```
33.25
```

Mimo výše uvedený interval by tak měly být asi 2 hodnoty. Je však možné, že mimo tento interval bude jen jedna hodnota, ale třeba i tři hodnoty. Ověřme si tuto skutečnost:

In[142]:=
```
HodnotyVIntervalu =
  Select[CejniDelka, (# < prumer + KvantilN * s) && (# > prumer - KvantilN * s) &];
Length[HodnotyVIntervalu]
```

Out[143]=
```
34
```

Celkem je tedy 34 hodnot v intervalu [30,2; 46,5] a jen jedna hodnota je mimo tento interval. 35 hodnot není tak moc, abychom si nemohli dovolit vypsat všechny hodnoty (seřazeny od nejmenší po největší):

In[144]:=
```
Sort[CejniDelka]
```

Out[144]=
```
{30., 31.1, 31.2, 33.5, 34., 34.5, 34.7, 35., 35.1, 36.2, 36.2,
 36.2, 36.4, 37.2, 37.2, 37.3, 38.3, 38.5, 38.6, 38.7, 39.2, 39.5, 39.7,
 40.5, 40.6, 40.6, 40.9, 41.5, 41.6, 42.6, 44., 44.1, 45.3, 45.9, 46.5}
```

Vidíme, že jedna hodnota (30,0) je menší než dolní hranice intervalu 30,2. Žádná hodnota není větší než horní hranice 46,5036. Jedna hodnota (46,5) je však této hranici velmi blízko. Naše

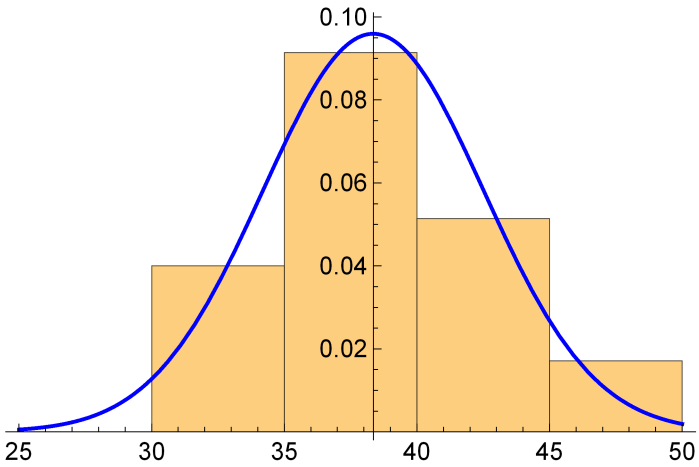
představa, že přibližně 95 % hodnot bude v intervalu [30,2; 46,5] tak velmi dobře odpovídá naměřeným hodnotám.

Připravme ještě grafické srovnání histogramu, jakožto reprezentace naměřených hodnot, a hustoty normálního rozdělení s parametry μ = 38,3543 a σ = 4,15787. Jelikož plocha pod grafem hustoty je vždy rovna 1, tedy $\int_{-\infty}^{\infty} f(x)\,dx = 1$, musíme u histogramu změnit měřítko na y-ové ose tak, aby celková plocha obdélníků histogramu byla rovna 1. To zajistíme tak, že u příkazu **Histogram[]** nastavíme volbu „PDF“. Jde o zkratku z anglického termínu pro hustotu: Probability Density Function.

In[145]:=
```
hist = Histogram[CejniDelka, Automatic, "PDF"];
hustotaN = Plot[PDF[NormalDistribution[prumer, s], x], {x, 25, 50},
   PlotStyle → Directive[Blue, Thick]];
```

In[147]:=
```
Show[{hist, hustotaN},
 PlotRange → {{25, 50}, Automatic},
  AxesOrigin → {prumer, 0}, BaseStyle → {FontSize → 13}]
```

Out[147]=



Při porovnání tohoto obrázku s histogramem, který jsme získali v části Histogram, si všimněme, že tvar histogramu zůstal stejný, ale měřítko y-ové osy se změnilo, jak bylo popsáno výše. Podrobné vysvětlení výše uvedených příkazů zde neuvádíme. Věříme, že čtenáři už by měli být v tuto chvíli schopni s pomocí nápovědy a vlastních pokusných změn kódu přijít na význam jednotlivých částí kódu.

Výše uvedený graf nás utvrzuje v tom, že popis veličiny celková délka cejna jako normálně rozdělené náhodné veličiny se střední hodnotou 38,4 a směrodatnou odchylkou 4,2 je vhodný.

Popis kvantitativního znaku v případě nenormality

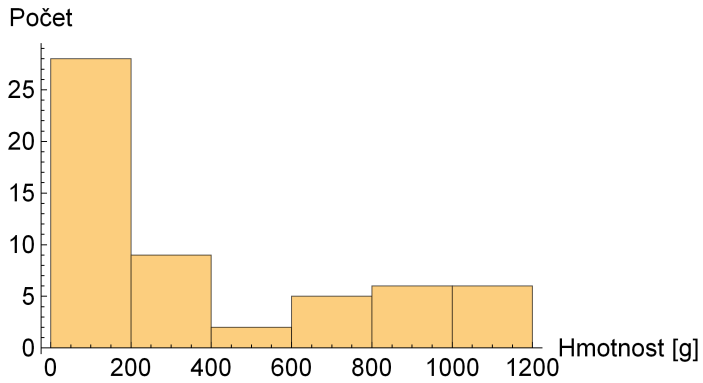
V případě celkové délky cejna jsme měli situaci zjednodušenou tím, že jsme tuto veličinu mohli považovat za normálně rozdělenou. Popis se pak omezil na odhad dvou parametrů tohoto rozdělení – střední hodnoty a rozptylu. Ne vždy je však situace tak jednoduchá. Příkladem může být popis hmotnosti okounů.

In[148]:=
```
HmotnostOkounu = Ryby[[7, All, 3]];
```

In[149]:=
```
nHO = Length[HmotnostOkounu];
```

In[150]:=
```
Histogram[HmotnostOkounu, BaseStyle → {FontSize → 13},
 AxesLabel → {"Hmotnost [g]", "Počet"}]
```

Out[150]=

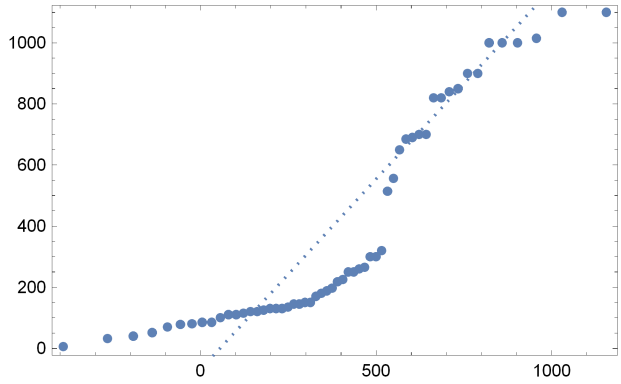


Na první pohled je zřejmá nesymetričnost rozdělení hmotnosti okounů. Hmotnost 28 z 56, tedy rovné poloviny, sledovaných okounů nepřesáhla 200 gramů, hmotnosti dalších 28 byly víceméně rovnoměrně rozprostřeny v intervalu 200 až 1200 gramů.

Ačkoliv již z histogramu vidíme, že rozdělení hmotnosti okounů není normální (protože není symetrické), předvedeme zde ze studijních důvodů vyšetření normality této veličiny tak, jak bylo popsáno v části Posouzení normality:

In[151]:=
```
QuantilePlot[HmotnostOkounu]
```

Out[151]=



In[152]:=
```
ShapiroWilkTest[HmotnostOkounu]
```

Out[152]= 7.34223×10^{-7}

Vidíme, že body vykreslované v kvantilovém grafu ani vzdáleně nekopírují „diagonálu“, p-hodnota Shapirova-Wilkova testu je velmi malá (výrazně menší než 0,05) a tedy na 5% hladině významnosti zamítáme hypotézu normality. Poznamenejme zde, že pokud (jakýkoliv) software ukazuje takto nízkou p-hodnotu, uvádí se většinou ve výstupech, které statistik prezentuje (ať už jde o odborný článek či diplomovou práci), pouze informace, že p-hodnota < 0,0001. Slangově říkáme, že nulovou hypotézu „s přehledem zamítáme“ a je už celkem nepodstatné, zda software uvádí hodnotu řádu 10^{-7} nebo třeba 10^{-20}.

Rozdělení hmotnosti okounů opět popíšeme pomocí několika číselných veličin. Začneme opět charakteristikami polohy:

1) CHARAKTERISTIKY POLOHY

In[153]:=
```
prumerHO = Mean[HmotnostOkounu]
```

Out[153]= 382.239

Při popisu celkové délky cejnů jsme kromě bodového odhadu střední hodnoty v podobě aritmetického průměru uvedli také intervalový odhad střední hodnoty. Uvedený interval byl konstruován za

předpokladu normality. V případě hmotnosti okounů předpoklad normality není splněn. Nabízí se tedy otázka, lze-li podobný intervalový odhad střední hodnoty najít i v případě, že podmínka normality rozdělení není splněna. Odpověď dává *centrální limitní věta*, tedy tvrzení, že „průměrujeme-li hodně hodnot, je rozdělení artmetického průměru přibližně normální“, viz Debata o normálním rozdělení.

Pro dosti velký počet průměrovaných hodnot (v praxi často postačí 30 a více) můžeme tedy i bez předpokladu normality rozdělení sledované veličiny určit interval, který obsahuje skutečnou hodnotu parametru μ (střední hodnotu) s předem danou pravděpodobností. Protože však rozdělení aritmetického průměru je jen „přibližně“ normální, mluvíme o přibližném intervalovém odhadu.

In[154]:=
```
KvantilN = InverseCDF[NormalDistribution[0, 1], 0.975]
```
Out[154]=
```
1.95996
```

In[155]:=
```
sHO = StandardDeviation[HmotnostOkounu]
```
Out[155]=
```
347.618
```

In[156]:=
```
prumerHO + {-1, 1} * KvantilN * sHO / Sqrt[nHO]
```
Out[156]=
```
{291.194, 473.284}
```

Vidíme, že konfidenční interval je dosti široký. Naše nejistota v odhadu střední hodnoty je tedy značná. Odhad střední hodnoty bychom tedy mohli prezentovat v následující podobě: Střední hodnota hmotnosti okounů je přibližně 380 gramů. Skutečná střední hodnota je s 95 % pravděpodobností obsažena v intervalu (290; 470) gramů.

Další charakteristikou polohy je medián:

In[157]:=
```
Median[HmotnostOkounu]
```
Out[157]=
```
207.5
```

Vidíme, že se hodnoty aritmetického průměru (382,2) a výběrového mediánu (207,5) dosti liší. Je to dáno tím, že výběrový medián je odhadem mediánu a medián v případě nesymetrického rozdělení není totéž co střední hodnota. Situace, kdy je střední hodnota větší než medián, je typická pro rozdělení protáhlejší směrem napravo. V anglické terminologii se jim říká right-skewed (případně right-tailed) distributions. Jsou to rozdělení s kladnou šikmostí. O koeficientu šikmosti pojednáme v sekci Charakteristiky šikmosti.

Vzhledem k nesymetrii rozdělení je dobré informaci o mediánu, tedy hodnotě, která rozděluje hodnoty na dvě stejně velké skupiny „malých“ a „velkých“, doplnit informací o minimální a maximální hodnotě.

In[158]:=
```
{Min[HmotnostOkounu], Max[HmotnostOkounu]}
```
Out[158]=
```
{5.9, 1100.}
```

Minimální zjištěná hmotnost okouna byla tedy 5,9 gramů, maximální pak 1100 gramů. Všimněme si, že hodnota mediánu 207,5 není uprostřed intervalu (5,9; 1100,0). Vzdálenost od mediánu k minimální hodnotě je výrazně menší než jeho vzdálenost k maximální hodnotě. I to je typickým znakem rozdělení s kladnou šikmostí.

Zamysleme se ještě nad hodnotou minima: 5,9 gramů jistě není hmotností dospělého okouna. Předpoklad, že jde pouze o dopělé jedince, zde není splněn. To je důležité si uvědomit při formulaci závěrů. Musíme si vždy být vědomi, že máme k dispozici výběr z nějaké populace a právě o této populaci hovoříme při formulaci závěrů našich studií. Zde sledovanou populací nejsou jen dospělí okouni, ale zřejmě „všichni okouni“.

2) CHARAKTERISTIKY VARIABILITY

Při stanovování intervalového odhadu střední hodnoty jsme, stejně jako v případě vyšetřování délky cejnů, použili (výběrovou) směrodatnou odchylku. Ta však nyní nemá stejnou interpretaci jako v

případě normálního rozdělení. Interval, který získáme odečtením respektive přičtením „dvojnásobku“ výběrové směrodatné odchylky k aritmetickému průměru nyní vypadá takto:

```
In[159]:= prumerHO + {-1, 1} * KvantilN * sHO
Out[159]= {-299.079, 1063.56}
```

Tento interval zřejmě nevystihuje rozsah hodnot hmotnosti okounů. Pro veličiny, které nemají normální rozdělení, se tedy takovýto interval nepoužívá. Daleko lepší charakteristikou variability je pro veličiny, které nemají normální rozdělení, velikost intervalu určeného minimem a maximem hodnot, tzv. rozpětí, případně velikost intervalu určeného dolním a horním kvartilem, tzv.mezikvartilové rozpětí.

```
In[160]:= Rozpeti = Max[HmotnostOkounu] - Min[HmotnostOkounu]
Out[160]= 1094.1

In[161]:= MezikvartiloveRozpeti = InterquartileRange[HmotnostOkounu]
Out[161]= 575.
```

Z těchto čísel je patrné, že
1) rozdíl mezi největším (z hlediska hmotnosti) a nejmenším okounem v našem souboru byl přibližně 1100 gramů,
2) vynecháme-li ze souboru čtvrtinu nejmenších a čtvrtinu největších okounů, bude rozdíl mezi největším a nejmenším ze zbývajících okounů asi 600 gramů.

3) CHARAKTERISTIKY ŠIKMOSTI

Už z histogramu byla patrná nesymetrie rozdělení hodnot hmotnosti okounů. O této nesymetrii jsme se pak „přesvědčili“ z nerovnosti mediánu a střední hodnoty a také z nesymetrické polohy mediánu v intervalu určeném minimem a maximem naměřených hodnot. Nesymetrii rozdělení můžeme jednoduše popsat pomocí koeficientu šikmosti (anglicky Skewness). V nápovědě k příkazu **Skewness[]** se dozvíme, že koeficient šikmosti je definován jako podíl třetího centrálního momentu a druhého centrálního momentu umocněného na 3/2. Jde tedy o „bezrozměrnou“ veličinu. Symetrická rozdělení, jako je například normální rozdělení, mají šikmost nulovou.

```
In[162]:= Mean[(HmotnostOkounu - Mean[HmotnostOkounu])^3] /
           Mean[(HmotnostOkounu - Mean[HmotnostOkounu])^2]^(3 / 2)
Out[162]= 0.82158

In[163]:= Skewness[HmotnostOkounu]
Out[163]= 0.82158
```

Vidíme, že rozdělení hmotnosti okounů má kladnou šikmost. To odpovídá našemu výše uvedenému zjištění, že medián je menší než střední hodnota a že má blíže k minimu než k maximu zjištěných hodnot.

Někdy se může stát, že rozdělení sledované veličiny je symetrické, unimodální a přesto nemůže být považováno za normální, neboť hodnoty jsou hodně „nahuštěny“ kolem střední hodnoty a dále je zde poměrně hodně „odlehlých pozorování“ – extrémě velkých či extrémě malých hodnot. Takové rozdělení má velkou špičatost. Tu můžeme vyjádřit tzv. koeficientem špičatosti, jenž patří mezi tzv. charakteristiky špičatosti. Získáme jej příkazem **Kurtosis[]**. Tato charakteristika je však důležitá hlavně u symetrických rozdělení, což není případ námi sledované hmotnosti okounů. Více o koeficientu špičatosti se lze dočíst v nápovědě k příkazu **Kurtosis[]**. O špičatosti je rovněž zmínka v kapitole 3.2.

Krabicový graf (boxplot)

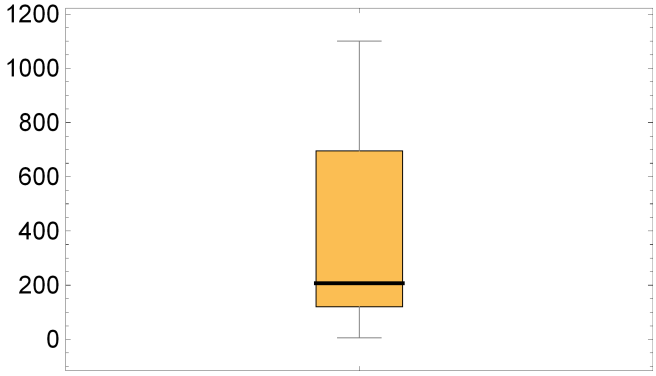
Krabicový graf je grafickým zobrazením nejdůležitějších číselných charakteristik rozdělení – mediánu, horního a dolního kvartilu, minima a maxima.

Z krabicového grafu je patrno, která pozorování můžeme považovat za odlehlá. Pokud soubor nějaká odlehlá pozorování vůbec obsahuje, jsou vyznačena tečkami. Detaily týkající se určení toho, které pozorování lze považovat za odlehlé, lze nalézt ve většině základních učebnic pojednávajících o popisné statistice (např. [3] na straně 150) nebo třeba pomocí vyhledávače WolframAlpha při zadání hesla „box plot“. *Mathematica* nabízí další „vylepšení“ boxplotu – je dobré nahlédnout do nápovědy k příkazu **`BoxWhiskerChart[]`**, pomocí něhož software *Mathematica* krabicové grafy vykresluje.

Podle polohy mediánu vůči kvartilům respektive vůči minimu a maximu můžeme posuzovat symetrii rozdělení.

In[164]:=
```
BoxWhiskerChart[HmotnostOkounu,
 {{"Outliers", "●"}, {"MedianMarker", 1, Thick}},
 ChartBaseStyle → EdgeForm[Black],
 BaseStyle → {FontSize → 13}]
```

Out[164]=



Podívejme se nejprve na jednotlivé volby, které jsme udělali:
- volba {{„Outliers“, „●“},{„MedianMarker“,1,Thick}} říká, jak mají být vyznačena případná odlehlá pozorování (symbolem ●) a jak má být vyznačen medián (tlustou čarou přes celou šířku obdélníku).
- volba ChartBaseStyle→EdgeForm[Black] říká, že obdélník („krabička“) má být ohraničen černou čarou,
- volba BaseStyle→{FontSize→13} upravuje velikost písma.

Co z výše zobrazeného krabicového grafu vidíme:
- nesymetrii rozdělení, neboť medián je podstatně blíže dolnímu kvartilu než hornímu, blíž minimu než maximu (dolní „vous“ je kratší než horní „vous“),
- žádné pozorování nelze označit za odlehlé,
- na číselné ose můžeme přečíst orientační hodnoty minima (0), dolního kvartilu (100), mediánu (200), horního kvartilu (700) a maxima (1100). Přesné hodnoty se zobrazí, umístíme-li na obrázek kurzor: minimum (5,9), dolní kvartil (120), medián (197), horní kvartil (690), maximum (1100).

Pozorný čtenář si možná povšiml menší „nesrovnalosti“. Zatímco funkce **`Median[]`** vrátila hodnotu 207,5, v krabicovém grafu je jako medián uváděna hodnota 197. Jak je to možné? Připomeňme zde, že teoretickou hodnotu mediánu neznáme a pouze ji z dat odhadujeme. Je více způsobů, jak tento odhad provést. Jednou z možností je vzít „prostřední“ hodnotu z hodnot uspořádaných podle velikosti. Co když je ale hodnot sudý počet (jako je tomu zde: 56)? Můžeme třeba vzít dvě prostřední hodnoty a ty průměrovat. Tento postup zřejmě používá funkce **`Median[]`**:

```
In[165]:= nHO = Length[HmotnostOkounu]
Out[165]= 56

In[166]:= Sort[HmotnostOkounu][[{nHO / 2, nHO / 2 + 1}]]
Out[166]= {197., 218.}

In[167]:= Mean[Sort[HmotnostOkounu][[{nHO / 2, nHO / 2 + 1}]]]
Out[167]= 207.5
```

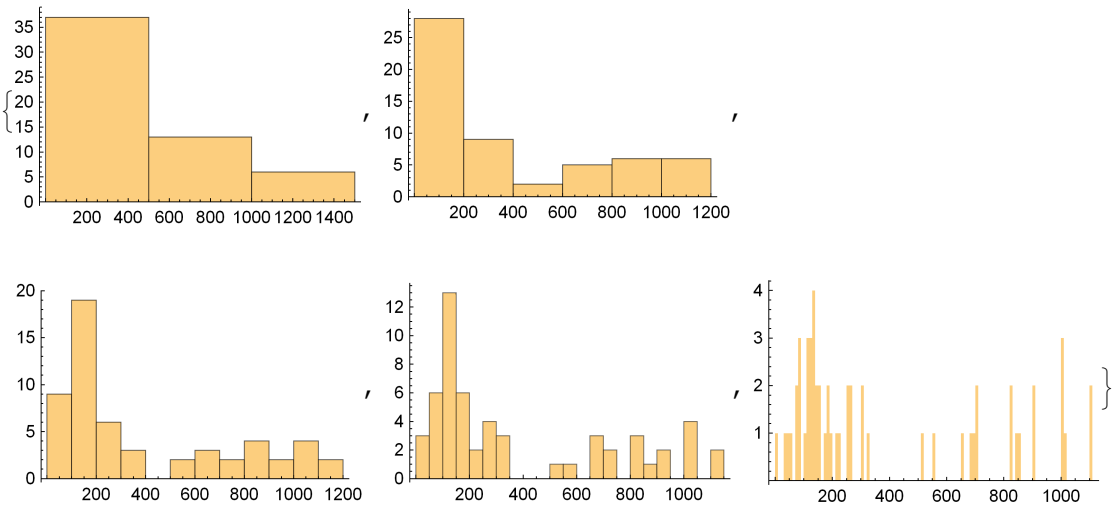
Jinou možností je vzít menší z obou prostředních hodnot, tedy v našem případě hodnotu 197. Ta je použita k odhadu mediánu v krabicovém grafu. Stejný výsledek také dostaneme, když pomocí funkce **`Quantile[]`** budeme odhadovat 0,5-kvantil (což je právě medián):

```
In[168]:= Quantile[HmotnostOkounu, 0.5]
Out[168]= 197.
```

Zatím jsme si ukázali dva grafické nástroje vhodné pro vizualizaci rozdělení kvantitativní veličiny – histogram a krabicový graf neboli boxplot. Kterému z nich dát přednost? Nejlépe je používat oba. Předností krabicového grafu je jeho jednoduchost. Histogram oproti tomu poskytuje „detailnější“ informaci o rozdělení hodnot. Je dobré mít na paměti, že tvar histogramu je do značné míry určen počtem dílků, na které rozdělíme x-ovou osu. Není přitom zdaleka pravda, že čím jemnější dělení zvolíme, tím názornější a „přesnější“ informaci dostaneme, jak snad napoví následující srovnání:

```
In[169]:= PocetKategorii = {3, 5, 9, 21, 101};
          Table[Histogram[HmotnostOkounu, i], {i, PocetKategorii}]
```

Out[170]=



Shrnutí

Kvantitativní znaky (jako například hmotnost či délka ryby) popisujeme:

- číselně pomocí tzv. číselných charakteristik polohy (střední hodnota, medián, ...), variability (rozptyl či směrodatná odchylka, rozpětí, mezikvartilové rozpětí), šikmosti (koeficient šikmosti) a špičatosti (koeficient špičatosti)
- graficky pomocí histogramu či krabicového grafu.

Užitečné příkazy softwaru *Mathematica*:

`Mean[], Median[], Min[], Max[],`
`Variance[], StandardDeviation[], InterquartileRange[],`
`Skewness[], Kurtosis[], Histogram[], BoxWhiskerChart[]`.

2.3 ... má však cenné údaje

„Statistika nuda je, má však cenné údaje...“ zpívá se v refrénu písně, jejíž text v roce 1980 pro televizní pohádku Princové jsou na draka napsal Zdeněk Svěrák. V prvních dvou kapitolách jsme věnovali pozornost tomu, jak popsat a přehledně prezentovat data kvalitativní i kvantitativní povahy – dělali jsme tzv. popisnou statistiku. Popsali jsme náš datový soubor. A možná nám to připadalo nudné. Popisná statistika opravdu trochu „nuda je“. Co je zajímavého na tom, že okounů bylo 56 a plotic 20? Samozřejmě se můžeme ptát na otázky typu „Jaká je nejběžnější ryba českých rybíků?“ (tento údaj z námi analyzovaných dat nevyčteme) nebo „Jaké velikosti běžně dorůstá okoun?“, avšak to opravdu zajímavé, na co statistika může poukázat, jsou souvislosti mezi různými znaky.

V pohádce Princové jsou na draka, v níž si drak činil nároky na princeznu, statistika odhalila zajímavou souvislost:

> Když drak si z nosu síru pouští
> a Honza na něj číhá v houští,
> tak statistika předpovídá,
> že nestvůra už neposnídá.

Jak asi pohádkový statistik došel ke svým závěrům? Samozřejmě nemohl přečíst všechny pohádky o dracích a princeznách, ale pár jich jistě našel. A vedl si záznamy (vytvořil datový soubor). U každé pohádky, v níž vystupoval drak a princezna, si poznamenal, zda v pohádce vystupuje také Honza a jak dopadl drak. Popisná statistika, v níž postupně shrnul zastoupení pohádek s Honzou a zastoupení pohádek, v nichž drak nedopadne dobře, nic zajímavého nepřinesla. Pak ale pohádkový statistik začal zkoumat, zda mezi výskytem Honzy a koncem draka neexistuje nějaká souvislost, tuto souvislost skutečně objevil a využil ji k předpovědi drakova konce. Tato předpověď byla pro dvořany (a zejména pak pro princeznu) nesmírně cenná, neboť po jejím zveřejnění drak o princeznu úplně ztratil zájem. Poznamenejme, že pohádka Princové jsou na draka tak paradoxně patřila k těm, kde se Honza vyskytoval a přesto drak neskončil s useknutou hlavou.

Nezbývá nám tedy, než dát Zdeňku Svěrákovi za pravdu v tom, že statistika může být občas nuda, jindy však může přinést nesmírně zajímavé a cenné informace. Ty bývají skryty zejména ve vzájemné souvislosti dvou různých veličin. O tom, jak hledat souvislosti mezi dvěma veličinami, pojednává následující kapitola.

Než se dáme do čtení kapitoly zabývající se vyšetřováním souvislostí mezi dvěma veličinami, měli bychom se zamyslet ještě nad jednou otázkou: *Je „nudná“ část práce statistika zahrnující popisnou statistiku jednotlivých veličin nutná, nebo ji můžeme s klidem přeskočit?* Odpověď se snad ukrývá v následujícím příměru: tak, jako v běžném hovoru, ani ve vědě nezačínáme „odprostředka“. Sdělení typu „Kdybych to věděl, udělal bych opak.“ je bezpředmětné, pokud nebylo nejprve vyřčeno, koho se sdělení týká, jakou informaci dotyčný „nevěděl“ a co pak udělal. Stejně tak chceme-li vyslovit něco o vztahu dvou veličin, měli bychom nejprve popsat, o jaké veličiny jde a jaké je jejich rozdělení v námi sledovaném souboru. Tento „popis“ do značné míry vymezuje populaci, pro kterou jsou výsledky dalších analýz (týkajících se vztahů mezi sledovanými veličinami) relevantní. Například výsledky vědecké studie týkající se účinnosti nějakého léku, který byl podáván pacientům, jejichž věk byl v rozmezí 30 až 40 let, se nedají použít na populaci pacientů ve věku 60 až 80 let. Můžeme se sice dohadovat, že léčebný efekt bude i u této skupiny podobný, jde však jen o náš ničím nepodložený dohad.

3 Popis vztahu dvou a více znaků

Tak jako při popisu jedné veličiny, kdy jsme rozlišovali znaky kvalitativní a kvantitativní, i při popisu vztahu mezi dvěma znaky budeme volit různé metody podle charakteru těchto znaků. Budeme přitom rozlišovat tři případy:

- oba znaky jsou kvalitativní,
- jeden znak je kvalitativní a jeden kvantitativní,
- oba znaky jsou kvantitativní.

V závěrečné části této kapitoly se krátce zmíníme o grafických možnostech při vyšetřování vztahu více znaků najednou.

3.1 Dva kvalitativní znaky

Popis vztahu dvou kvalitativních znaků si ukážeme na příkladu, kde prvním sledovaným znakem je druh ryby a druhým znakem je, zda máme či nemáme k dispozici údaj o pohlaví ryby. Je totiž možné, že u některých druhů ryb lze pohlaví určit obtížněji než u jiných.

Vytvoření vektoru Pohlavi

Informace o pohlaví ryb je uložena v 9. sloupci naší datové tabulky ryby. Využijme však skutečnosti, že pod názvem Ryby máme data uložená „po druzích“.

In[171]:=
```
pohlavi3 = Ryby[[All, All, 9]]
```

Out[171]=
```
{{NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 1, 1, NA, 1, NA, NA, NA, 1,
   NA, NA, NA, NA, 1, NA, NA, NA, 0, 0, NA, 1, 0, NA}, {NA, NA, NA, NA, 0, NA},
  {NA, NA, NA, NA, NA, NA, 0, 0, 0, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA},
  {NA, 1, 1, 1, NA, NA, 0, 0, NA, 0, 0}, {1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0},
  {NA, 0, NA, NA, NA, 1, NA, NA, NA, NA, 0, 0, NA, NA, NA, 0, 0},
  {NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 1, 0, 0, NA, NA,
   NA, 0, 0, 0, NA, NA, NA, NA, NA, NA, 0, NA, NA, 0, 0, NA, 0, NA,
   0, 0, NA, NA, 0, 0, 0, 0, 0, 0, NA, NA, 0, 0, 0, 0, 0, 0, 0, 1, 0}}
```

Výše uvedený vektor jsme pojmenovali pohlavi3. Trojka zde symbolizuje počet kategorií, kterých může takto chápaná veličina pohlaví nabývat (jsou to kategorie 0, 1 a NA značící postupně samici, samce a chybějící hodnotu). V zápise Ryby[[All,All,9]] první „All“ znamená, že budeme pracovat se všemi sedmi druhy, druhé „All“ říká, že pracujeme se všemi řádky, a konečně 9 znamená 9. sloupec datové tabulky (kde jsou uloženy údaje o pohlaví ryb).

Nyní bychom chtěli místo vektoru, který obsahuje 3 možné hodnoty 0, 1 a NA, definovat vektor, který má jen dvě hodnoty, a to „chybí“ (pokud je ve vektoru pohlavi3 uvedeno NA) a „známe“ (pokud je ve vektoru pohlavi3 uvedeno 0 nebo 1). Potřebujeme tedy funkci, která vrátí hodnotu „známe“, jestliže má na vstupu celé číslo (anglicky integer), v opačném případě vrátí hodnotu „chybí“. Takovou funkci můžeme definovat pomocí příkazu **If[]**:

In[172]:=
```
If[IntegerQ[#], "zname", "chybi"] &
```

Out[172]=
```
If[IntegerQ[#1], zname, chybi] &
```

Přesvědčme se, že tato funkce dělá to, co chceme:

```
In[173]:= If[IntegerQ[#], "zname", "chybi"] &["NA"]
Out[173]= chybi

In[174]:= If[IntegerQ[#], "zname", "chybi"] &[0]
Out[174]= zname

In[175]:= If[IntegerQ[#], "zname", "chybi"] &[1]
Out[175]= zname
```

Nyní tuto funkci použijeme k přetvoření vektoru pohlavi3. K tomu nám poslouží funkce **Map[]**:

```
In[176]:= Pohlavi = Map[If[IntegerQ[#], "zname", "chybi"] &, pohlavi3, {2}]
Out[176]= {{chybi, chybi, chybi, chybi, chybi, chybi, chybi, chybi, chybi, chybi, chybi,
    chybi, chybi, zname, zname, chybi, zname, chybi, chybi, chybi, zname,
    chybi, chybi, chybi, chybi, zname, chybi, chybi, chybi, zname, zname,
    chybi, zname, zname, chybi}, {chybi, chybi, chybi, chybi, zname, chybi},
   {chybi, chybi, chybi, chybi, chybi, chybi, zname, zname, zname, zname,
    zname, chybi, zname, chybi, chybi, zname, chybi, chybi, zname, chybi},
   {chybi, zname, zname, zname, chybi, chybi, zname, zname, chybi, zname, zname},
   {zname, zname, zname, zname, zname, zname, zname,
    zname, zname, zname, zname, zname, zname, zname},
   {chybi, zname, chybi, chybi, chybi, zname, chybi, chybi, chybi,
    chybi, zname, zname, chybi, chybi, chybi, zname, zname},
   {chybi, chybi, chybi, chybi, chybi, chybi, chybi, chybi, chybi, chybi,
    chybi, chybi, zname, zname, zname, chybi, chybi, chybi, zname,
    zname, zname, chybi, chybi, chybi, chybi, chybi, chybi, zname,
    chybi, chybi, zname, zname, chybi, zname, chybi, zname, zname,
    chybi, chybi, zname, zname, zname, zname, zname, zname, chybi, chybi,
    zname, zname, zname, zname, zname, zname, zname, zname, zname}}
```

Ve výše uvedeném příkazu je pomocí symbolu {2} zadáno, že námi určená funkce **If[...]** se nemá použít na vektor pohlavi3 jako celek, ale na jeho dílčí části (vektor pohlavi3 je tvořen sedmi vektory). Kdybychom symbol {2} vynechali, funkce **If[...]** by byla postupně použita na 7 vektorů tvořících vektor pohlavi3. To jsou ale vektory, tedy nikoliv celá čísla. Výsledkem by tak byl nic neříkající vektor sedmi hodnot „chybí“:

```
In[177]:= Map[If[IntegerQ[#], "zname", "chybi"] &, pohlavi3]
Out[177]= {chybi, chybi, chybi, chybi, chybi, chybi, chybi}
```

Kontingenční tabulka

Informaci o vztahu dvou kvantitativních znaků zobrazíme nejlépe pomocí tzv. kontingenční tabulky. Podklady pro tuto tabulku bychom si v některých případech mohli relativně snadno opatřit kombinací příkazů **Tally[]** a **Map[]**:

```
In[178]:= Map[Tally, Pohlavi ]
Out[178]= {{{chybi, 26}, {zname, 9}}, {{chybi, 5}, {zname, 1}},
   {{chybi, 12}, {zname, 8}}, {{chybi, 4}, {zname, 7}}, {{zname, 14}},
   {{chybi, 11}, {zname, 6}}, {{chybi, 29}, {zname, 27}}}
```

Problém je zde zejména v tom, že u jednoho druhu (druh č. 5 – koruška), je počet pozorování v kategorii „chybí“ nula. Druhým případným problémem je, že pořadí, v jakém se ve výstupu objevují hodnoty „chybi“ a „zname“ záleží na pořadí, v jakém se vyskytují v datovém souboru.

Počty ryb jednotlivých druhů se známou resp. neznámou hodnotou u znaku pohlaví určíme pomocí

příkazu **Count[]**:

```
In[179]:= Map[Count[#, "chybi"] &, Pohlavi]
Out[179]= {26, 5, 12, 4, 0, 11, 29}

In[180]:= Map[Count[#, "zname"] &, Pohlavi]
Out[180]= {9, 1, 8, 7, 14, 6, 27}
```

Vytvořme z těchto údajů tabulku:

```
In[181]:= kategorie = {"zname", "chybi"};
          Table[Map[Count[#, c] &, Pohlavi], {c, kategorie}]
Out[182]= {{9, 1, 8, 7, 14, 6, 27}, {26, 5, 12, 4, 0, 11, 29}}
```

Pro další zpracování se nám bude hodit transpozice této tabulky:

```
In[183]:= Tab1 = Transpose[Table[Map[Count[#, c] &, Pohlavi], {c, kategorie}]]
Out[183]= {{9, 26}, {1, 5}, {8, 12}, {7, 4}, {14, 0}, {6, 11}, {27, 29}}
```

Zobrazme nyní kontingenční tabulku:

```
In[184]:= TableForm[Tab1, TableHeadings → {cesky, kategorie},
           TableAlignments → {Right, Center}]
Out[184]//TableForm=
```

	zname	chybi
cejn	9	26
jelec	1	5
plotice	8	12
cejnek	7	4
koruška	14	0
štika	6	11
okoun	27	29

Nyní bychom chtěli tabulku uloženou pod názvem Tab1 rozšířit o řádkové a sloupcové součty. Řádkové součty získáme tak, že místo každé dvojice čísel obsažené ve vektoru Tab1 budeme uvažovat trojici, jejíž třetí člen je součtem prvních dvou:

```
In[185]:= Tab2 = Tab1 /. {a_, b_} → {a, b, a + b}
Out[185]= {{9, 26, 35}, {1, 5, 6}, {8, 12, 20},
           {7, 4, 11}, {14, 0, 14}, {6, 11, 17}, {27, 29, 56}}
```

Sloupcové součty bychom pak mohli získat pomocí příkazu **Total[]**:

```
In[186]:= Map[Total, Transpose[Tab2]]
Out[186]= {72, 87, 159}
```

Přidejme tento řádek k tabulce uložené pod názvem Tab2:

```
In[187]:= Tab2 = Append[Tab2, Map[Total, Transpose[Tab2]]]
Out[187]= {{9, 26, 35}, {1, 5, 6}, {8, 12, 20}, {7, 4, 11},
           {14, 0, 14}, {6, 11, 17}, {27, 29, 56}, {72, 87, 159}}
```

Přidejme ještě řádek obsahující názvy sloupců a sloupec obsahující názvy řádků:

```
In[188]:= nazvy1 = Append[kategorie, "celkem"]
Out[188]= {zname, chybi, celkem}
```

In[189]:=
```
nazvy2 = Flatten[{"", cesky, "celkem"}]
```

Out[189]=
```
{, cejn, jelec, plotice, cejnek, koruška, štika, okoun, celkem}
```

In[190]:=
```
Tab2 = Prepend[Tab2, nazvy1];
```

In[191]:=
```
Tab2 = Join[Transpose[{nazvy2}], Tab2, 2];
```

Pomocí příkazu **Grid[]** tabulku připravíme do podoby vhodné k prezentaci:

In[192]:=
```
Grid[Tab2, Dividers → {{-2 → True, 2 → True}, {-2 → True, 2 → True}},
 Alignment → {{Left, Right, Right, Right}},
 Background → {{LightBlue}, {LightBlue}}, Frame → True]
```

Out[192]=

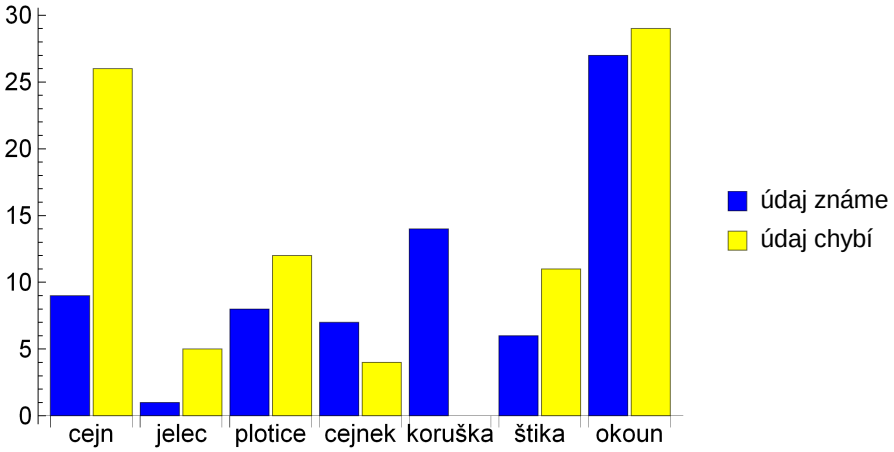
	zname	chybi	celkem
cejn	9	26	35
jelec	1	5	6
plotice	8	12	20
cejnek	7	4	11
koruška	14	0	14
štika	6	11	17
okoun	27	29	56
celkem	72	87	159

Vizualizace vztahu dvou kvalitativních znaků

Popisnou statistiku kvalitativního znaku jsme graficky znázorňovali pomocí tzv. sloupkového diagramu. Tento grafický nástroj můžeme použít i pro vizualizaci údajů z kontingenční tabulky:

In[193]:=
```
BarChart[Tab1, ChartStyle → {Blue, Yellow}, ChartLabels → {cesky, None},
 ChartLegends → {"údaj známe", "údaj chybí"}, BaseStyle → {FontSize → 13}]
```

Out[193]=

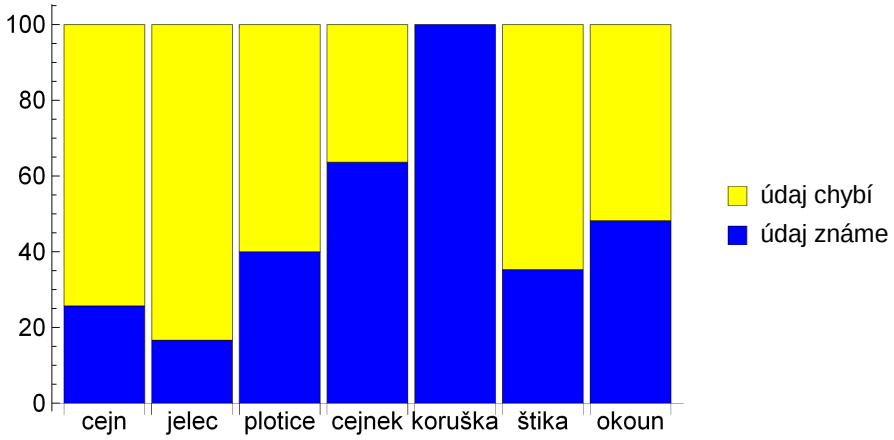


Vidíme, že u druhu koruška bylo určeno pohlaví všech ryb v souboru. Také u druhu cejnek bylo více ryb, u nichž informaci o pohlaví známe, než těch, u nichž tuto informaci neznáme. U ostatních druhů byla převaha ryb, u nichž není informace o pohlaví známa.

Podíl pozorování, u nichž je údaj o pohlaví znám, můžeme snadno zobrazit pomocí grafu, který se někdy označuje termínem „součtový sloupcový graf“ či „skládaný sloupcový graf“ (anglicky stacked bar chart):

In[194]:=
```
BarChart[Tab1, ChartStyle → {Blue, Yellow}, ChartLabels → {cesky, None},
 ChartLegends → {"údaj známe", "údaj chybí"}, BaseStyle → {FontSize → 13},
 ChartLayout -> "Percentile"]
```

Out[194]=



Z tohoto grafu sandno zjistíme, že největší podíl ryb, u nichž informaci o pohlaví známe, je u druhu koruška (100 %), naopak mezi zástupci druhu jelec je tento podíl nejmenší (méně než 20 %).

Cvičení – samice a samci

Zatím jsme zkoumali pouze to, zda informace o pohlaví ryb je či není uvedena. Většinou nás ale zajímají srovnání mezi samci a samicemi a chtěli bychom tedy analyzovat především údaje o těch rybách, u nichž je nám informace o pohlaví známa. Zabývejme se tedy nyní veličinou (kvalitativní), jež má tři kategorie (samice/samec/neznámo), obsahující informace o pohlaví ryb.

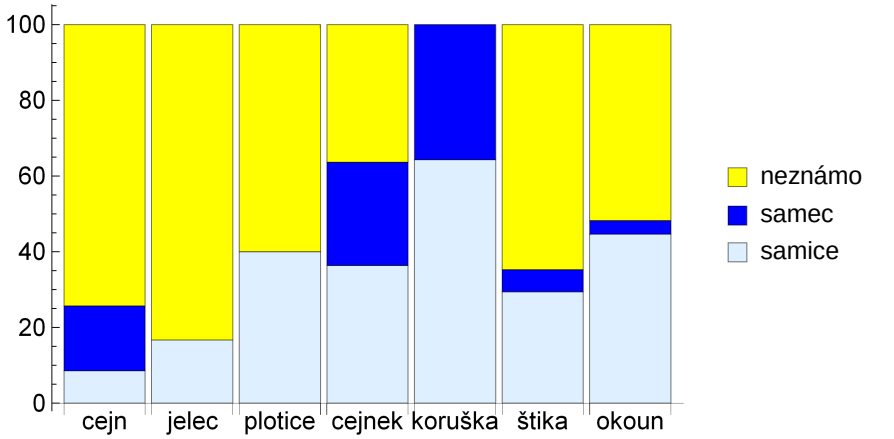
In[195]:=
```
kategorie2 = {0, 1, "NA"};
```

In[196]:=
```
Tab2 = Transpose[Table[Map[Count[#, c] &, pohlavi3], {c, kategorie2}]]
```

Out[196]=
```
{{3, 6, 26}, {1, 0, 5}, {8, 0, 12}, {4, 3, 4}, {9, 5, 0}, {5, 1, 11}, {25, 2, 29}}
```

In[197]:=
```
BarChart[Tab2, ChartLayout -> "Percentile",
 ChartStyle → {LightBlue, Blue, Yellow}, ChartLabels → {cesky, None},
 ChartLegends → {"samice", "samec", "neznámo"}, BaseStyle → {FontSize → 13}]
```

Out[197]=



Tento graf přehledně zobrazuje veškerou informaci v datech týkající se zastoupení pohlaví v rámci jednotlivých druhů. Kromě informace o množství (podílu) chybějících hodnot z něj můžeme vyčíst také informace o zastoupení samic (resp. samců) mezi rybami, u nichž je informace o pohlaví známa. Porovnáním velikosti světle modrého dílu (samice) a tmavě modrého dílu (samci) dospějeme k závěru, že u šesti ze sedmi sledovaných druhů byly mezi rybami, u nichž bylo pohlaví

určeno, častěji samice než samci, neboť světle modré části jsou „větší“ než tmavě modré. Jediným druhem, u kterého mezi rybami s určeným pohlavím převládali samci, je cejn. U dvou druhů – jelce a plotice – nebyl mezi rybami, jejichž pohlaví bylo určeno, žádný samec.

Některé z výše uvedených skutečností můžeme dále dokumentovat číselně. Např. početní převaha samic mezi rybami, jejichž pohlaví bylo určeno, u šesti ze sedmi druhů se zřejmě (i když ne nutně) projeví v jejich celkové početní převaze:

```
In[198]:= ZastoupeniPohlaviCelkem = Tally[Flatten[pohlavi3]]
Out[198]= {{NA, 87}, {1, 17}, {0, 55}}

In[199]:= CelkovePocty = ZastoupeniPohlaviCelkem[[All, 2]]
Out[199]= {87, 17, 55}

In[200]:= N[CelkovePocty / Total[CelkovePocty], 3]
Out[200]= {0.547, 0.107, 0.346}
```

Z celkového počtu 159 ryb pohlaví nebylo určeno u 87 ryb, což představuje více než polovinu ze všech námi zkoumaných ryb (54,7 %). Dále bychom mohli říci, že 55 ryb (34,6 % celkového počtu) bylo určeno jako samice a 17 ryb (10,7 % celkového počtu) bylo určeno jako samci.

Zatímco číslo 54,7 % je jistě užitečné, neboť nám sděluje, že u více než poloviny ryb nebylo pohlaví určeno (nebo o něm neexistují záznamy), čísla 34,6 % a 10,7 % už tak moc informativní nejsou. Daleko lepší by byla informace, jaká část z ryb, u nichž informaci o pohlaví máme, byli samci a jaká část byly samice. Tedy, kolik procent je 55 (samic) z 17+55=72 (ryb, u nichž bylo pohlaví určeno), respektive kolik procent je 17 (samců) ze 72 (určených ryb):

```
In[201]:= N[CelkovePocty[[2 ;; 3]] / Total[CelkovePocty[[2 ;; 3]]], 3]
Out[201]= {0.236, 0.764}
```

Můžeme tedy říci, že z ryb, u nichž bylo pohlaví určeno, bylo 76,4 % samic a 23,6 % samců.

Závěr našich pozorování je poněkud neradostný – pokud bychom měli srovnávat váhu či rozměry samců a samiček určitého druhu ryb, máme k dispozici velmi málo pozorování – u druhu okoun máme k dispozici 27 pozorování, z tohoto počtu jsou však pouze 2 samci. U ostatních druhů máme pozorování ještě méně. Podobná srovnání tak zřejmě z těchto dat nezískáme.

Shrnutí

Vztah dvou kvalitativních znaků popisujeme:

- pomocí kontingenční tabulky,
- graficky pomocí sloupcového (sloupkového) diagramu případně skládaného (součtového) sloupcového grafu.

Užitečné příkazy softwaru *Mathematica*:

Count[], Total[], Tally[], Grid[]
BarChart[], případně s argumentem **ChartLayout->“Percentile“**.

3.2 Kvalitativní a kvantitativní znak

Klademe-li si otázky typu: „Je cejn větší než cejnek?“ nebo „Jaký druh ryb máme lovit, abychom chytili co největší rybu?“, pak nás zajímá vztah jednoho kvalitativního a jednoho kvantitativního znaku. Kvalitativním znakem je druh ryby, kvantitativním znakem je „velikost“ ryby. Velikost může

být charakterizována různě, třeba jako délka těla (včetně ocasu či bez ocasu). Zjednodušeně se dá říci, že pro popis vztahu kvalitativního a kvantitativního znaku se hodí srovnání charakteristik kvantitativního znaku pro různé úrovně kvalitativního znaku. V našem případě budeme tedy srovnávat charakteristiky délky jednotlivých druhů ryb.

Číselný vektor, se kterým budeme pracovat, uložme pod názvem Delky:

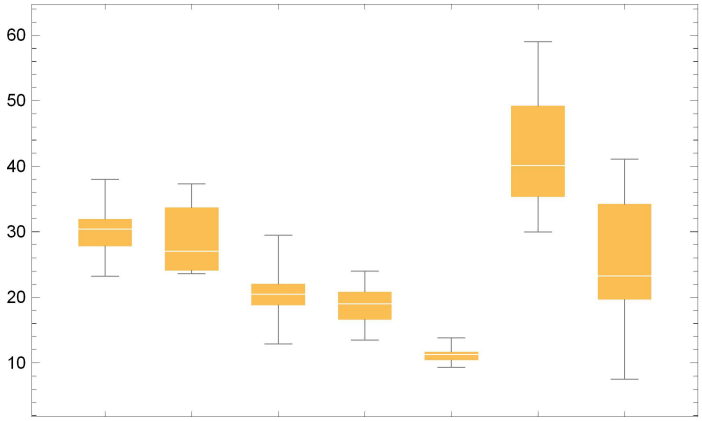
In[202]:=
```
Delky = Ryby[[All, All, 4]]
```

Out[202]=
```
{{23.2, 24., 23.9, 26.3, 26.5, 26.8, 26.8, 27.6, 27.6, 28.5, 28.4,
  28.7, 29.1, 29.5, 29.4, 29.4, 30.4, 30.4, 30.9, 31., 31.3, 31.4, 31.5,
  31.8, 31.9, 31.8, 32., 32.7, 32.8, 33.5, 35., 35., 36.2, 37.4, 38.},
 {23.6, 24.1, 25.6, 28.5, 33.7, 37.3}, {12.9, 16.5, 17.5, 18.2, 18.6, 19.,
  19.1, 19.4, 20.4, 20.5, 20.5, 21., 21.1, 22., 22., 22.1, 23.6, 24., 25., 29.5},
 {13.5, 14.3, 16.3, 17.5, 18.4, 19., 19., 19.8, 21.2, 23., 24.},
 {9.3, 10., 10.1, 10.4, 10.7, 10.8, 11.3, 11.3, 11.4, 11.5, 11.7, 12.1, 13.2, 13.8},
 {30., 31.7, 32.7, 34.8, 35.5, 36., 40., 40.,
  40.1, 42., 43.2, 44.8, 48.3, 52., 56., 56., 59.},
 {7.5, 12.5, 13.8, 15., 15.7, 16.2, 16.8, 17.2, 17.8, 18.2, 19., 19., 19., 19.3,
  20., 20., 20., 20., 20., 20.5, 20.5, 20.7, 21., 21.5, 22., 22., 22.6, 23., 23.5,
  25., 25.2, 25.4, 25.4, 25.4, 25.9, 26.9, 27.8, 30.5, 32., 32.5, 34., 34., 34.5,
  34.6, 36.5, 36.5, 36.6, 36.9, 37., 37., 37.1, 39., 39.8, 40.1, 40.2, 41.1}}
```

Představu o rozdělení délek u jednotlivých druhů nejrychleji získáme porovnáním krabicových grafů délek jednotlivých druhů. Příkaz pro vykreslení krabicového grafu již známe – jde o příkaz **`BoxWhiskerChart[]`**. Díky tomu, že vektor Delky sestává ze sedmi „podvektorů“, vykreslí se sedm krabicových grafů odpovídajících jednotlivým částem vektoru Delky.

In[203]:=
```
BoxWhiskerChart[Delky]
```

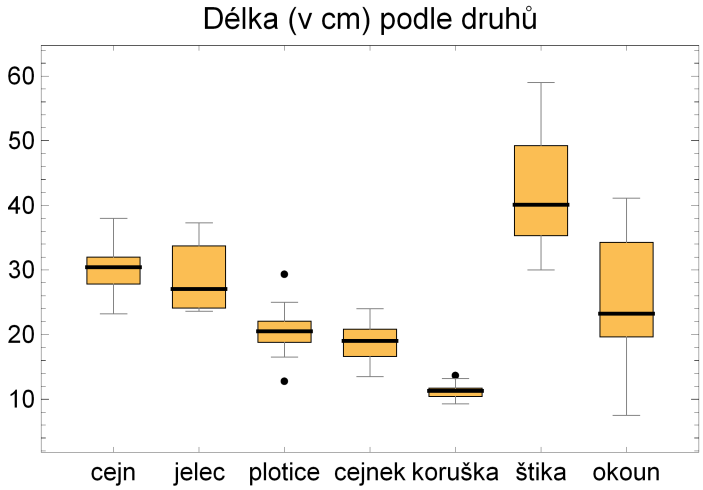
Out[203]=



Výše uvedený příkaz byl schválně ponechán ve své nejjednodušší podobě. Máte pocit, že graf, který jsme získali, dobře poslouží k získání představy o rozdělení délek jednotlivých druhů? Pokud ano, je vidět, že s daty o rybách již nějaký čas pracujete. Pro statistika, který data analyzuje, je výše uvedený obrázek dostatečný, protože je již v problematice zorientován a ví, co která „krabička“ znamená. Pokud však bude chtít tento graf použít k prezentaci, musí ho dobře popsat, aby se v něm zorientoval i člověk, který s daty o rybách tak dlouho nepracoval. Na jednoduchý příkaz se „nabalí“ spousta specifikací finálního vzhledu grafu.

In[204]:=
```
BoxWhiskerChart[Delky,
 {{"Outliers", "•"}, {"MedianMarker", 1, Thick}},
 ChartBaseStyle → EdgeForm[Black],
 BaseStyle → {FontSize → 13}, ChartLabels → cesky,
 PlotLabel → "Délka (v cm) podle druhů"]
```

Out[204]=



Co je z grafu vidět? Jednak to, co bychom viděli při vykreslení sedmi samostatných grafů. Pro konkrétní druh v něm můžeme najít nejdůležitější číselné charakteristiky polohy (viz kapitola 2), z jejich vztahu pak vyčteme informaci o variabilitě a symetrii (či asymetrii) rozdělení. A taky o jeho špičatosti – velké množství odlehlých pozorování by nasvědčovalo velké špičatosti, naopak „vousy“ výrazně kratší než délka „krabičky“ a žádná odlehlá pozorování budou svědčit o malé špičatosti.

Z grafu je však vidět také něco, co by sedm samostatných krabicových grafů poskytlo jen stěží – přímé srovnání charakteristik délky jednotlivých druhů. Vidíme zde například, že cejni jsou zhruba stejně velcí jako jelci, o něco menší jsou plotice a cejnci. Nejmenší jsou korušky, naopak největší jsou štiky. Velikost okounů, jejíž medián je blízký mediánu velikosti plotic či cejnků, vykazují velkou variabilitu („krabička“ je hodně protáhlá). Rovněž velikost štik má velkou variabilitu. Naopak u korušek je variabilita velikosti malá. Rozdělení celkových délek se zdají u všech druhů symetrická. U plotic si povšimněme dvou odlehlých pozorování. Jak již bylo zmíněno, náchylnost k odlehlým pozorováním je typická pro znaky s velkou špičatostí. Srovnáme-li koeficienty špičatosti u jednotlivých druhů, uvidíme, že u plotic je tento koeficient opravdu největší:

In[205]:=
```
Map[Kurtosis, Delky]
```

Out[205]=
```
{2.74947, 1.76188, 4.16834, 2.18506, 2.88541, 2.05231, 1.97046}
```

Koeficient špičatosti se tedy s výjimkou plotic pohybuje v rozmezí přibližně 1,8 až 2,9. U plotic je však 4,2.

Jak je vidět z výše uvedeného příkladu, je dobré svá pozorování z grafu dokumentovat číselně srovnáním číselných charakteristik jednotlivých druhů ryb.

Pro základní srovnání druhů podle velikosti nám postačí srovnání základních charakteristik polohy, tedy buďto aritmetických průměrů (jakožto odhadů střední hodnoty) či výběrových mediánů. V zásadě můžeme použít jak aritmetický průměr, tak medián. Vzhledem k výše vypozorované symetrii rozdělení délek u jednotlivých druhů by neměl být v jejich použití velký rozdíl.

In[206]:=
```
MedianyDelky = Map[Median, Delky]
```

Out[206]=
```
{30.4, 27.05, 20.5, 19., 11.3, 40.1, 23.25}
```

Tato informace stačí statistikovi, který s daty pracuje, k tomu, aby viděl, že největší jsou štiky (medián 40,1) a nejmenší korušky (medián 11,3). Dá však trochu práce, než tuto informaci

připravíme do úhledné tabulky obsahující názvy jednotlivých druhů seřazené dle mediánu celkové délky. Postupně si ukážeme nejdůležitější kroky přípravy takové tabulky:

Přidání názvů druhů:

```
In[207]:= TabulkaM = Transpose[{cesky, MedianyDelky}]
Out[207]= {{cejn, 30.4}, {jelec, 27.05}, {plotice, 20.5},
  {cejnek, 19.}, {koruška, 11.3}, {štika, 40.1}, {okoun, 23.25}}
```

Seřazení podle (mediánů) celkové délky:

```
In[208]:= DleMedianuDelky = Reverse[SortBy[TabulkaM, Last]]
Out[208]= {{štika, 40.1}, {cejn, 30.4}, {jelec, 27.05},
  {okoun, 23.25}, {plotice, 20.5}, {cejnek, 19.}, {koruška, 11.3}}
```

Přidání záhlaví tabulky:

```
In[209]:= TabulkaMed = Prepend[DleMedianuDelky, {"druh", "medián délky"}]
Out[209]= {{druh, medián délky}, {štika, 40.1}, {cejn, 30.4}, {jelec, 27.05},
  {okoun, 23.25}, {plotice, 20.5}, {cejnek, 19.}, {koruška, 11.3}}
```

Určení podoby tabulky (pomocí příkazu **Grid[]**) a podoby prezentovaných čísel (pomocí příkazu **NumberForm[]**):

```
In[210]:= NumberForm[Grid[TabulkaMed,
    Frame → True, Dividers → {2 → True, 2 → True},
    Alignment → {{Left, Center}},
    Background → {None, {LightBlue}}], {Infinity, 1}]
```

Out[210]//NumberForm=

druh	medián délky
štika	40.1
cejn	30.4
jelec	27.1
okoun	23.3
plotice	20.5
cejnek	19.0
koruška	11.3

Detailnější informace o charakteristikách polohy celkové délky jednotlivých druhů můžeme shrnout do tabulky, v níž kromě mediánu uvedeme u každého druhu také zjištěné hodnoty minima, maxima a dolního a horního kvartilu. Potřebné „podklady“ pro tuto tabulku získáme snadno. Stačí si uvědomit, jak bychom podobnou pětici čísel získali pro údaje o jednom konkrétním druhu, například u druhu číslo 1, jímž je cejn:

```
In[211]:= v = Delky[[1]];
In[212]:= {Min[v], Quartiles[v], Max[v]}
Out[212]= {23.2, {27.8, 30.4, 31.975}, 38.}
In[213]:= Flatten[{Min[v], Quartiles[v], Max[v]}]
Out[213]= {23.2, 27.8, 30.4, 31.975, 38.}
```

Nyní bychom místo „v“ chtěli postupně brát jednotlivé části vektoru „Delky“. Pomůžeme si definováním „ryzí funkce“ (viz kapitola 1.2) a příkazem **Map[]**:

In[214]:=
```
DelkaInfo = Map[Flatten[{Min[#], Quartiles[#], Max[#]}] &, Delky]
```

Out[214]=
```
{{23.2, 27.8, 30.4, 31.975, 38.},
 {23.6, 24.1, 27.05, 33.7, 37.3}, {12.9, 18.8, 20.5, 22.05, 29.5},
 {13.5, 16.6, 19., 20.85, 24.}, {9.3, 10.4, 11.3, 11.7, 13.8},
 {30., 35.325, 40.1, 49.225, 59.}, {7.5, 19.65, 23.25, 34.25, 41.1}}
```

Tento vektor již obsahuje všechny informace (čísla) potřebné pro zamýšlenou tabulku. Je však ještě třeba opět tabulku připravit do podoby vhodné pro prezentaci.

In[215]:=
```
t1 = Table[Flatten[{cesky[[i]], DelkaInfo[[i]]}], {i, 1, 7}]
```

Out[215]=
```
{{cejn, 23.2, 27.8, 30.4, 31.975, 38.},
 {jelec, 23.6, 24.1, 27.05, 33.7, 37.3}, {plotice, 12.9, 18.8, 20.5, 22.05, 29.5},
 {cejnek, 13.5, 16.6, 19., 20.85, 24.}, {koruška, 9.3, 10.4, 11.3, 11.7, 13.8},
 {štika, 30., 35.325, 40.1, 49.225, 59.}, {okoun, 7.5, 19.65, 23.25, 34.25, 41.1}}
```

In[216]:=
```
t2 = Prepend[t1, {"druh", "Min", "Q1", "Med", "Q3", "Max"}];
```

In[217]:=
```
NumberForm[Grid[t2,
  Frame → True, Dividers → {2 → True, 2 → True},
  Alignment → {{Left, Right}},
  Background → {None, {LightBlue}}], {Infinity, 1}]
```

Out[217]//NumberForm=

druh	Min	Q1	Med	Q3	Max
cejn	23.2	27.8	30.4	32.0	38.0
jelec	23.6	24.1	27.1	33.7	37.3
plotice	12.9	18.8	20.5	22.1	29.5
cejnek	13.5	16.6	19.0	20.9	24.0
koruška	9.3	10.4	11.3	11.7	13.8
štika	30.0	35.3	40.1	49.2	59.0
okoun	7.5	19.7	23.3	34.3	41.1

Jako cvičení přenecháváme čtenáři úlohu sestavit tabulku, v níž bude u každého druhu uvedena jedna charakteristika polohy, jedna charakteristika variability, jedna charakteristika šikmosti a jedna charakteristika špičatosti.

Dalším cvičením by mohlo být zpracovat podobně údaje o hmotnosti jednotlivých druhů. Zde se vyskytne menší problém v tom, že u druhu cejn je jedna chybějící hodnota. Tu je třeba před analýzou vyloučit.

In[218]:=
```
Hmotnosti = Ryby[[All, All, 3]];
```

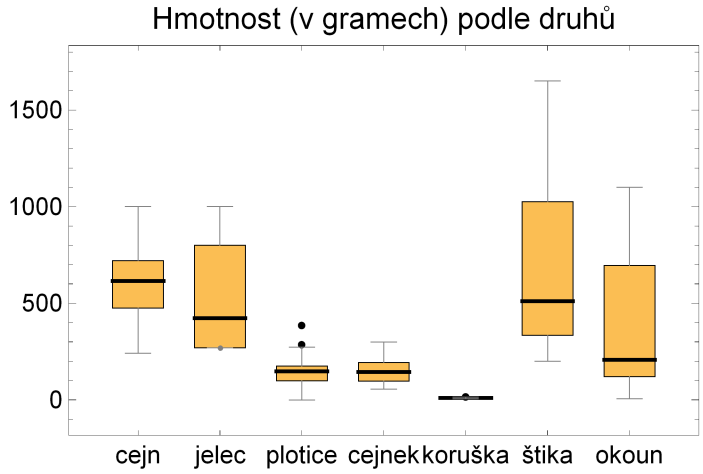
Čtenář může vyzkoušet, jak by nyní pracoval například příkaz **`Map[Median,Hmotnosti]`**.

In[219]:=
```
Hmotnosti = Map[Select[#, NumberQ] &, Hmotnosti];
```

Nyní už můžeme s vektorem Hmotnosti pracovat stejně, jako jsme předtím pracovali s vektorem Delky. Začneme opět srovnáním krabicových grafů. Komentář k následujícímu obrázku je ponechán čtenářům jako cvičení.

In[220]:=
```
BoxWhiskerChart[Hmotnosti,
 {{"Outliers", "•"}, {"MedianMarker", 1, Thick}},
 ChartBaseStyle → EdgeForm[Black],
 BaseStyle → {FontSize → 13}, ChartLabels → cesky,
 PlotLabel → "Hmotnost (v gramech) podle druhů"]
```

Out[220]=



Shrnutí

Vztah kvalitativního a kvantitativního znaku popisujeme:

- pomocí číselných charakteristik kvantitativního znaku uvedených zvlášť pro různé hodnoty kvalitativního znaku (prezentovány mohou být formou tabulky),
- graficky pomocí krabicového grafu.

Užitečné příkazy softwaru *Mathematica*:

`Map[], BoxWhiskerChart[]`.

3.3 Dva kvantitativní znaky

V předchozí kapitole 3.2 jsme mimo jiné zjistili, že „největší“ z námi sledovaných druhů ryb jsou štiky. Rozhodneme se tedy dozvědět se o těchto „velkých“ rybách víc. Jaký je třeba vztah mezi hmotností a délkou těchto ryb? Roste štika rychleji do délky nebo „do výšky“? Jaký je vztah mezi celkovou délkou těla a délkou ocasu u štik? I takovéto otázky si může klást člověk, jehož zájem vzbudily štiky. Ať si položí kteroukoliv z výše uvedených otázek, zajímá ho vztah mezi dvěma kvantitativními znaky. Popis takového vztahu (pokud nějaký vůbec existuje) může být učiněn pomocí nějaké matematické formule (rovnice). Nalezení této formule většinou vyžaduje použití nějakého druhu regrese. Téma regrese je příliš rozsáhlé, než aby bylo součástí tohoto textu. Zájemcům doporučujeme seznámení se s teoretickými základy regrese v monografii [4] a poměrně rosáhlou (anglicky psanou) dokumentaci k tématu „Experimental Data Analysis“ dostupnou na webových stránkách společnosti Wolfram: http://reference.wolfram.com/applications/eda/. Tato kapitola má skromnější cíl – ukázat, jaké kroky je třeba udělat ještě předtím, než (často nevhodně) „proložíme data přímkou“.

Prvním krokem bývá obvykle „vykreslení dat“, tedy vizualizace číselných údajů, jež máme analyzovat. V případě dvou kvantitativních znaků jako je hmotnost a délka použijeme tzv. bodový graf (v anglické terminologii scatter plot, česky se též používá označení XY-bodový graf či korelační diagram). Pomocí něj velmi rychle získáme představu o tom, zda vůbec existuje vztah mezi zkoumanými znaky, jak je „silný“, případně jaká formule by byla vhodná pro jeho popis (tvar závislosti).

Hmotnost a délka štik

Zkoumáme tyto dva vektory:

In[221]:=
```
x1 = Ryby[[6, All, 3]];
x2 = Ryby[[6, All, 4]];
```

Každá štika v našem souboru je charakterizována dvojicí číselných údajů – hmotností a délkou. Budeme zde pracovat s údajem o délce „bez ocasu“ (který beztak mnoho neváží), mohli bychom ale samozřejmě uvažovat i celkovou délku. V průběhu analýzy se bude hodit mít uložen právě vektor dvojic popisujících jednotlivé štiky. Získat ho můžeme různými způsoby:

In[223]:=
```
Transpose[{x1, x2}]
```

Out[223]=
```
{{200., 30.}, {300., 31.7}, {300., 32.7}, {300., 34.8}, {430., 35.5}, {345., 36.},
 {456., 40.}, {510., 40.}, {540., 40.1}, {500., 42.}, {567., 43.2}, {770., 44.8},
 {950., 48.3}, {1250., 52.}, {1600., 56.}, {1550., 56.}, {1650., 59.}}
```

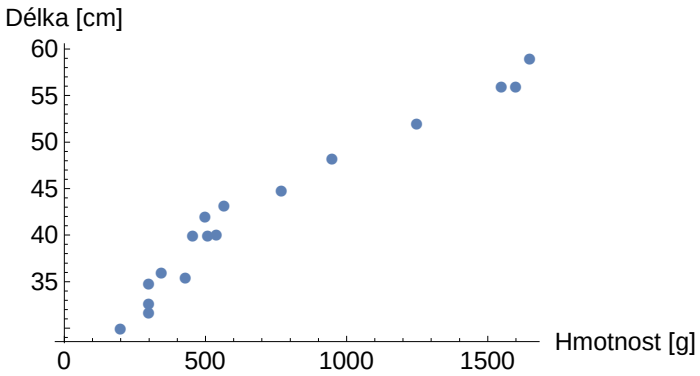
In[224]:=
```
HmotnostDelka = Ryby[[6, All, {3, 4}]]
```

Out[224]=
```
{{200., 30.}, {300., 31.7}, {300., 32.7}, {300., 34.8}, {430., 35.5}, {345., 36.},
 {456., 40.}, {510., 40.}, {540., 40.1}, {500., 42.}, {567., 43.2}, {770., 44.8},
 {950., 48.3}, {1250., 52.}, {1600., 56.}, {1550., 56.}, {1650., 59.}}
```

Nyní již přikročíme k vykreslení bodového grafu. Použijeme k němu příkaz **`ListPlot[]`**. V základní podobě by vypadal takto: **`ListPlot[HmotnostDelka]`**. My však přidáme několik dalších specifikací, aby byl graf v podobě připravené pro prezentaci:

In[225]:=
```
ListPlot[HmotnostDelka,
 PlotMarkers → Automatic, BaseStyle → {FontSize → 13},
 AxesLabel → {"Hmotnost [g]", "Délka [cm]"}]
```

Out[225]=



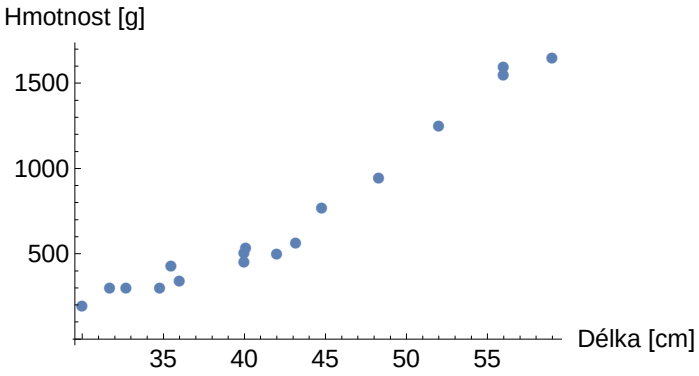
Většinou nás zajímá opačná závislost – „když chytím padesáticentimetrovou štiku, kolik bude asi vážit“? Statistik by měl mít vždy rozmyšleno, který znak bude považovat za „vysvětlující veličinu“ a který za „vysvětlovanou“. Funkce **`ListPlot[]`** použije první z dvojice hodnot jako x-ovou souřadnici bodu, druhou jako y-ovou. Bude tedy lepší, když budeme pracovat s dvojicemi:

In[226]:=
```
DelkaHmotnost = Ryby[[6, All, {4, 3}]]
```

Out[226]=
```
{{30., 200.}, {31.7, 300.}, {32.7, 300.}, {34.8, 300.}, {35.5, 430.}, {36., 345.},
 {40., 456.}, {40., 510.}, {40.1, 540.}, {42., 500.}, {43.2, 567.}, {44.8, 770.},
 {48.3, 950.}, {52., 1250.}, {56., 1600.}, {56., 1550.}, {59., 1650.}}
```

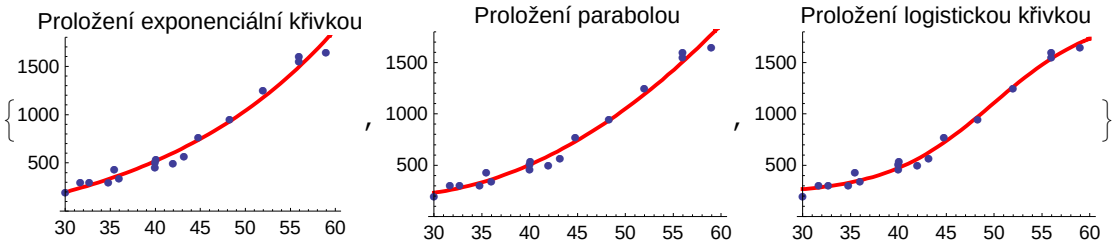
In[227]:=
```
graf1 = ListPlot[DelkaHmotnost,
  PlotMarkers → Automatic, BaseStyle → {FontSize → 13},
  AxesLabel → {"Délka [cm]", "Hmotnost [g]"}]
```

Out[227]=



TVAR ZÁVISLOSTI

Z bodového grafu vidíme, že závislost mezi délkou a hmotností štik rozhodně není lineární. Nejde tedy o přímou úměru a „proložení přímkou“ zde není vhodné. Tvar závislosti spíše připomíná logistickou křivku. Mohli bychom rovněž uvažovat o parabole či exponenciální křivce. Kterou z těchto křivek použít je v tuto chvíli těžké rozhodnout. Zásadní rozdíl mezi logistickou křivkou a ostatními dvěma je v tom, jak popíší závislost hmotnosti na délce pro malé délky (pod 30 cm) a pro velké délky (nad 55 cm). Zatímco graf exponenciální funkce bude pro velké hodnoty délky strmě stoupat, stoupání paraboly bude o něco pozvolnější a stoupání logistické křivky bude jen mírné. Který z těchto modelů bude pro velké hodnoty nejlepší? Nevíme. Je třeba si uvědomit, že nemáme žádná pozorování štik větších než 60 cm (nebo menších než 30 cm). Náš popis, pokud jej provedeme správně, bude vhodný pro štiky, jejichž velikost se pohybuje v rozmezí 30 až 60 cm. Možná bude dobrý i pro štiky mírně menší či mírně větší. Velkých „extrapolací“, tedy použití popisu (ve formě rovnice) pro štiky délek výrazně menších než 30 cm či větších než 60 cm, bychom se však dopouštět neměli.



Z ilustračního obrázku (jehož zdrojový kód lze najít v dodatku na konci této kapitoly) lze vypozorovat, že proložení exponenciální křivkou a parabolou jsou pro hodnoty 30 až 60 cm od sebe navzájem prakticky nerozeznatelné. Proložení logistickou křivkou vypadá malinko „přesvědčivěji“. Je však nutné si uvědomit, že k popisu logistické křivky byly zapotřebí čtyři parametry, zatímco u exponenciální křivky a paraboly jsme si vystačili se třemi parametry. Přesnost modelu tak byla získána na úkor jednoduchosti.

SÍLA ZÁVISLOSTI

Číselně můžeme sílu závislosti popsat pomocí korelačního koeficientu. Běžně používaný Pearsonův korelační koeficient je vhodným ukazatelem síly lineární závislosti. Námi zkoumaný vztah (mezi délkou a hmotností štik) však lineární není, a proto použijeme raději Spearmanův korelační koeficient. Ten je vhodný pro hodnocení síly libovolné monotónní závislosti. Monotonií zde rozumíme vlastnost, že hodnoty jednoho znaku rostou (či klesají) s rostoucími hodnotami druhého znaku. Tento růst (pokles) přitom nemusí být lineární.

In[228]:=
```
N[SpearmanRho[x1, x2]]
```

Out[228]=
```
0.985241
```

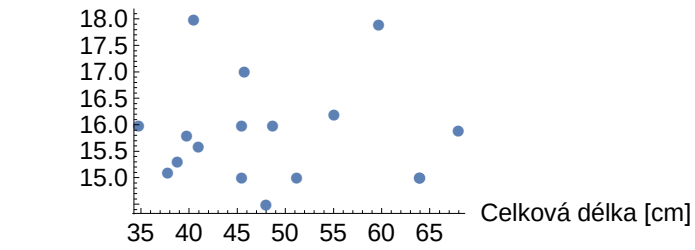
Hodnota Spearmanova korelačního koeficientu je velmi blízká jedničce, to svědčí o silné souvislosti (asociaci) mezi hmotností a délkou štik. Kladné znaménko Spearmanova koeficientu přitom interpretujeme tak, že čím je větší hodnota jedoho znaku (např. délky), tím je obecně větší také hodnota druhého znaku (hmotnosti). Slovíčkem „obecně“ zde vyjadřujeme skutečnost, že nejde o deterministické pravidlo, tedy mohou být dvě štiky, z nichž delší je zároveň lehčí. Většinou však delší štika bude zároveň těžší. Toto zjištění není samozřejmě nijak překvapivé. Jak již bylo řečeno, jde jen o první krok při popisu závislosti hmotnosti štik na jejich délce.

Délka a „výška“ štik

Jednou z otázek, které jsme si na začátku této kapitolky položili, byla otázka, zda štiky rostou rychleji do délky nebo „do výšky“. Údaj o výšce štiky máme uložen ve sloupečku číslo 7 naší datové tabulky. Výška je zde vyjádřena v procentech celkové délky. Mohli bychom uvažovat následovně: pokud ryba roste stejně rychle do výšky jako do délky, nebude údaj o výšce (v procentech) záviset na hodnotě délky. Úvaha je správná, je zde však háček (nikoliv jen rybářův). Ten háček je již v samotné formulaci otázky, kterou si klademe. Abychom mohli něco usuzovat o růstu ryb, museli bychom mít k dispozici dlouhodobá pozorování – u každé ryby bychom potřebovali několik chronologicky řazených údajů o jejích rozměrech. Zde však máme o každé rybě údaje jen v jednom časovém okamžiku. O růstu tak nemůžeme mnoho říci. Přesto však můžeme zkoumat vztah mezi výškou štik (vyjádřenou v procentech celkové délky) a jejich celkovou délkou. Začneme opět bodovým grafem:

In[229]:=
```
graf2 = ListPlot[Ryby[[6, All, {6, 7}]],
  PlotMarkers → Automatic, BaseStyle → {FontSize → 13},
  AxesLabel → {"Celková délka [cm]", "Výška [% celkové délky]"}]
```

Out[229]=



Žádná zřejmá souvislost z grafu není patrna. Dá se tedy očekávat nízká korelovanost sledovaných veličin. Ať použijeme Pearsonův či Spearmanův korelační koeficient, budou oba zřejmě blízké nule.

In[230]:=
```
Correlation[Ryby[[6, All, 6]], Ryby[[6, All, 7]]]
```

Out[230]=
```
-0.0384564
```

In[231]:=
```
N[SpearmanRho[Ryby[[6, All, 6]], Ryby[[6, All, 7]]]]
```

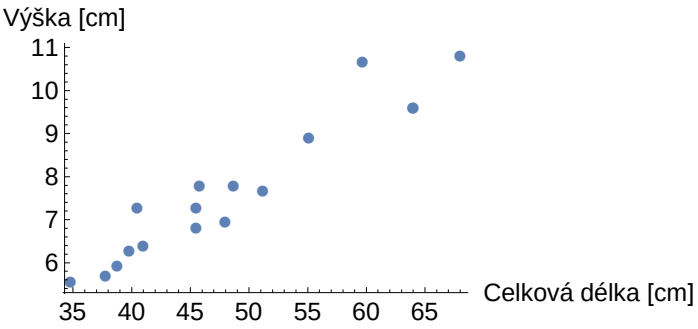
Out[231]=
```
-0.107057
```

Naše očekávání se potvrdilo – obě hodnoty jsou blízké nule. Pozor však na interpretaci tohoto výsledku. Neznamená, že by „výška“ štik nesouvisela s jejich délkou. Znamená však, že poměr výšky a délky těla štik je přibližně stejný pro všechny štiky bez ohledu na jejich celkovou délku. To ale znamená, že vztah mezi „výškou a délkou štik“ je lineární. Čím je větší délka, tím je také větší výška, a to přímo úměrně. Toto pozorování můžeme potvrdit následujícím bodovým grafem, v němž je vykreslena závislost výšky těla štik na jejich celkové délce. Připomeňme, že výšku vyjádřenou v centimetrech jsme vypočítali ze sloupců číslo 6 a 7 naší datové matice v kapitole 1 a uložili ji jako

desátý sloupec datové matice.

In[232]:=
```
graf3 = ListPlot[Ryby[[6, All, {6, 10}]],
  PlotMarkers → Automatic, BaseStyle → {FontSize → 13},
  AxesLabel → {"Celková délka [cm]", "Výška [cm]"}]
```

Out[232]=



Aniž bychom se pouštěli do vysvětlování teorie, zkusme přece jen odhadnout parametry přímky popisující závislost mezi výškou a celkovou délkou štik. K tomuto odhadu použijeme funkci **LinearModelFit[]**.

In[233]:=
```
model = LinearModelFit[Ryby[[6, All, {6, 10}]], x, x];
```

Rovnici popisující vztah mezi výškou a celkovou délkou získáme následovně:

In[234]:=
```
model["BestFit"]
```

Out[234]=
```
0.113681 + 0.156003 x
```

V této rovnici symbol „x“ vyjadřuje celkovou délku štiky. Střední hodnotu výšky štik o celkové délce x pak vypočteme právě jako 0.113681+0.156003 x. Například střední výška štik o délce 50 cm je rovna 7,9 cm:

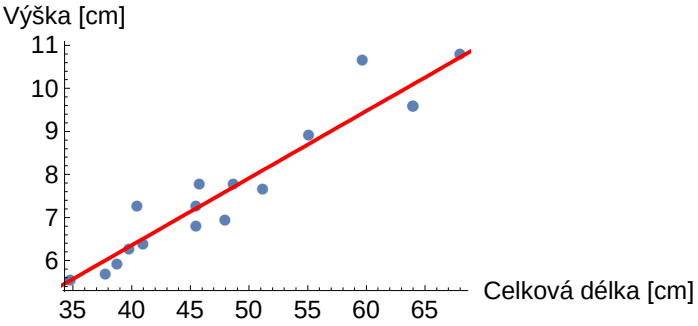
In[235]:=
```
model["BestFit"] /. x → 50
```

Out[235]=
```
7.91382
```

Graf proložení dat přímkou získáme takto:

In[236]:=
```
Show[graf3, Plot[model[x], {x, 30, 70}, PlotStyle → {Thick, Red}]]
```

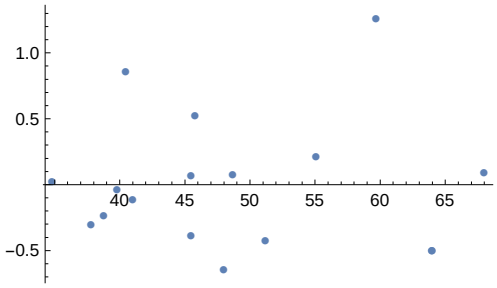
Out[236]=



Z tohoto grafu vidíme, že přímka docela dobře vystihuje tvar sledované závislosti. Neměli bychom zapomínat na předpoklady lineárního modelu, jehož parametry odhadujeme. O základních nástrojích diagnostiky modelu (tedy vyšetření vhodnosti modelu) se lze dočíst např. v kapitolách 8 a 9 monografie [4]. Zde alespoň ukažme graf reziduí. Z něj můžeme aspoň orientačně posoudit splnění nejdůležitějších předpokladů lineárního regresního modelu. Ty se týkají náhodné složky – jde o předpoklady normality, nekorelovanosti a homoskedasticity.

In[237]:=
```
ListPlot[Transpose[{Ryby[[6, All, 6]], model["FitResiduals"]}],
 PlotMarkers → {Automatic, 6}]
```

Out[237]=



Na první pohled není z výše uvedeného grafu patrno porušení některého z předpokladů.
Dalším úkolem statistika by bylo nalezení tzv. pásu spolehlivosti kolem regresní přímky (viz kapitola 5.3 monografie [5]) používaného k vyjádření nejistoty v odhadech středních hodnot pro různé hodnoty vysvětlující proměnné (v našem případě celkové délky).

Vraťme se však ještě k rovnici popisující závislost výšky a celkové délky štik. Jde o rovnici 0.113681 +0.156003 x. Absolutní člen (0,113) zde nemá valného významu. Směrnice (0,156) však již interpretaci má. Budeme-li pozorovat skupinu štik určité velikosti a druhou skupinu štik přesně o jeden centimetr větších než jsou štiky v první skupině, pak očekávaný rozdíl ve výškách štik mezi oběma skupinami bude 0,156 cm. Vyjádřeme nyní tuto hodnotu v procentech (změna ve výšce 0,156 cm odpovídá 15,6 % ze změny 1 cm délky) a porovnejme ji s průměrem hodnot ze 7. sloupce naší datové tabulky, který obsahuje údaje o výšce štik v podobě procent celkové délky:

In[238]:=
```
Mean[Ryby[[6, All, 7]]]
```

Out[238]=
```
15.8412
```

Podobnost těchto dvou čísel není vůbec náhodná. Detailní vysvětlení ponecháme hloubavému čtenáři jako domácí cvičení.

Shrnutí

Vztah dvou kvantitativních znaků popisujeme:

- pomocí korelačního koeficientu (Pearsonova nebo Spearmanova), případně pomocí rovnice regresní závislosti,
- graficky pomocí bodového grafu (označovaného též termíny XY-bodový graf či korelační diagram).

Užitečné příkazy softwaru *Mathematica*:

`Correlation[], SpearmanRho[], LinearModelFit[], ListPlot[]`.

3.4 Popis vztahu vícero znaků

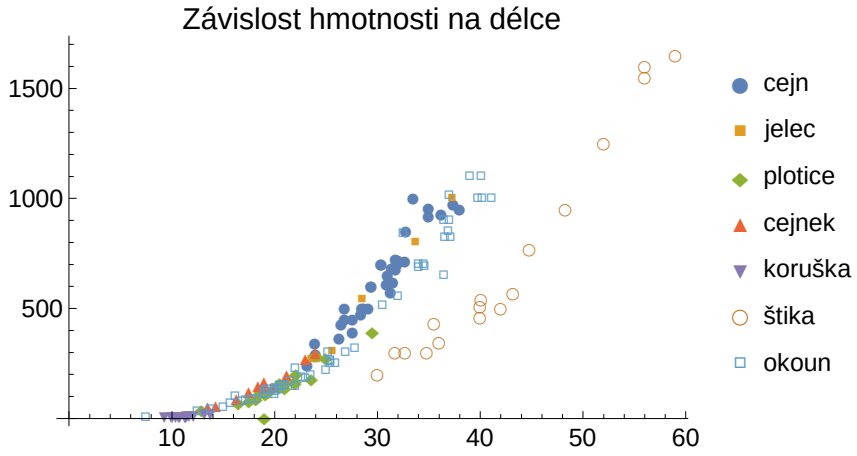
V kapitole 2 jsme pojednali o tom, jak popsat rozdělení jednoho (kvalitativního či kvantitativního) znaku. Kapitola 3 pak pojednává o popisu vztahu dvou znaků. Bylo by ještě na místě pojednat o situaci, kdy zkoumáme souvislosti mezi vícero znaky najednou. Aniž bychom si činili nárok na úplnost, pojednejme zde alespoň o několika „nástrojích“, které nám mohou být v takové situaci užitečné. Budeme se věnovat situaci, kdy máme větší množství kvantitativních znaků a pouze jeden či dva znaky kvalitativní, jako je tomu v případě dat o rybách.

V kapitole 3.3 jsme vyšetřovali vztah mezi délkou a hmotností štik. Co kdybychom se však nechtěli omezit pouze na štiky? Vykresleme tedy bodový graf závislosti hmotnosti na délce pro všechny druhy. Jednotlivé druhy přitom můžeme rozlišit barvou vykreslovaných „bodů“ případně tvarem vykreslovaných symbolů – nemusí jít o „puntík“, můžeme vykreslovat trojúhelníky, čtverečky, ale

třeba i čísla (viz nápověda pro heslo PlotMarkers).

In[239]:=
```
ListPlot[Ryby[[All, All, {4, 3}]], PlotLegends → cesky,
 PlotMarkers → Automatic, BaseStyle → {FontSize → 13},
 PlotLabel → "Závislost hmotnosti na délce"]
```

Out[239]=

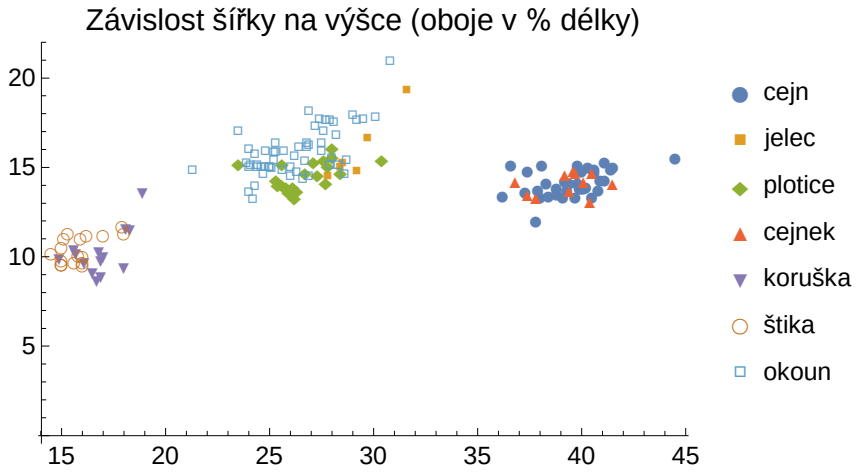


Vidíme, že ani u dalších druhů ryb není vztah mezi délkou a hmotností lineární. Pozorování týkající se štik (fialové kroužky) jsou poněkud stranou všech ostatních pozorování. Podíváme-li se na všechny ryby velikosti v rozmezí 30 až 40 cm, vidíme, že štiky jsou z nich nejlehčí. To odpovídá naší představě štiky jako „štíhlé dlouhé ryby“. Srovnáme-li štiku s okounem nebo cejnem stejné délky, bude štika s velkou pravděpodobností vážit méně. Okouni a cejni stejné délky budou mít i podobnou hmotnost, o něco těžší budou spíše cejni.

Možná jsme si to ani neuvědomili, ale výše uvedený graf ilustruje souvislost tří znaků – dvou kvantitativních (hmotnosti a délky) a jednoho kvalitativního (druh ryby). Uveďme zde ještě jeden příklad, v němž kvantitativními znaky budou tentokrát výška (v procentech celkové délky) a šířka (v procentech celkové délky).

In[240]:=
```
ListPlot[Ryby[[All, All, {7, 8}]], PlotLegends → cesky,
 PlotMarkers → Automatic, BaseStyle → {FontSize → 13},
 PlotLabel → "Závislost šířky na výšce (oboje v % délky)"]
```

Out[240]=



Na tomto grafu jsou patrné tři skupiny ryb (resp. druhů ryb) podobného vztahu výšky a šířky těla k jeho délce. První skupinu tvoří štiky a korušky. Podíl výšky i šířky těla k jeho délce je malý. Štiky a korušky tedy můžeme označit za „ryby protáhlého tvaru“. Druhou skupinu tvoří ryby druhů jelec, plotice a okoun. Třetí skupinu pak tvoří ryby s největším podílem výšky a délky těla (kolem 40 %). Jde o cejny a cejnky. Tyto druhy bychom tedy s trochou nadsázky označili za ryby „vysoké“. Právě tato podobnost (při rozdílných velikostech) je možná klíčem k českému názvosloví cejn/cejnek.

Pro zajímavost můžeme porovnat výsledky těchto svých pozorování s „rybími portréty“ z kapitolky 1.4. O protáhlém tvaru štik a korušek a zařazení cejnka mezi „vysoké ryby“ není z námi nalezených fotografií pochyb. Snad jen cejna bychom očekávali poněkud „vyššího“. Obrázek cejna na českých stránkách wikipedie (http://cs.wikipedia.org/wiki/Cejn_velký) však naše případné pochyby rozptýlí.

V našem původním datovém souboru je šest kvantitativních veličin, jejichž vztahy bychom mohli zkoumat. Existuje tedy 15 různých dvojic znaků, jejichž závislost bychom mohli chtít prozkoumat. Vykreslovat postupně 15 bodových grafů je poněkud zdlouhavé. Pro získání představy o datech poslouží jejich vykreslení do jediného obrázku. Čtverec, jež tvoří rámec obrázku, rozdělíme na šest krát šest menších čtverců (pěti svislými a pěti vodnorovnými čarami). Získáme tak vlastně jakousi matici o rozměrech 6 krát 6, jejímiž prvky však nebudou čísla, ale jednotlivé bodové grafy: v i-tém řádku a j-tém sloupci takovéto matice bude bodový graf zobrazující data týkající se i-tého a j-tého znaku, přičemž na x-ové ose jsou zaznamenávány hodnoty j-tého znaku, zatímco na y-ové jsou zaznamenávány hodnoty i-tého znaku.

Než budeme grafy vykreslovat, je třeba z dat vynechat pozorování, u nějž je chybějící hodnota hmotnosti. Příkaz **PairwiseScatterPlot[]**, který budeme používat, totiž funguje pouze pro „úplná“ data, tedy data bez chybějících hodnot. Které pozorování (řádek datové tabulky) máme vyloučit, určíme pomocí příkazu **Position[]**:

In[241]:=
```
index = Position[ryby[[All, 3]], "NA"][[1]]
```

Out[241]=
```
{14}
```

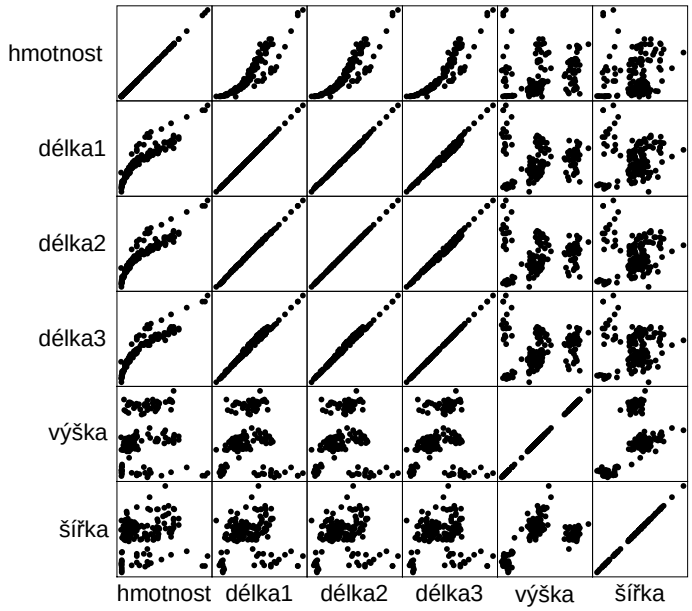
In[242]:=
```
ryby2 = Drop[ryby, index];
```

Nyní již můžeme použít příkaz **PairwiseScatterPlot[]** z knihovny StatisticalPlots. Pracujeme se sloupci 3 až 8 datové matice uložené pod názvem ryby2.

In[243]:=
```
Needs["StatisticalPlots`"]
```

In[244]:=
```
PairwiseScatterPlot[ryby2[[All, 3 ;; 8]],
 DataLabels → {"hmotnost", "délka1", "délka2", "délka3", "výška", "šířka"},
 BaseStyle → {FontSize → 12}]
```

Out[244]=



Výše uvedený obrázek považujme za pouhou pomůcku statistika při jeho práci s daty.

Pokud se setkáváte s podobným obrázkem poprvé, všimněte si, že
1) grafy na hlavní diagonále nejsou nijak zajímavé (rozmyslete si proč),

2) graf v i-tém řádku a j-tém sloupci a graf v j-tém řádku a i-tém sloupci zachycují závislost dvou stejných veličin a liší se pouze v tom, která veličina je brána jako nezávisle proměnná (ta je zaznamenávána na vodorovné ose) a která jako závisle proměnná (ta je zaznamenávána na svislé ose). Například graf v 1. řádku a 2. sloupci zachycuje závislost hmotnosti na délce (přesněji na hodnotě délka1), zatímco graf v 2. řádku a 1. sloupci zachycuje závislost délky (délky1) na hmotnosti. Jde tedy vlastně o tutéž závislost.

Z výše uvedeného grafu je patrná velmi silná vzájemná závislost znaků délka1, délka2, délka3. Jde o závislost lineární. Rovněž velmi silná, ale nelineární, je závislost mezi znaky popisujícími délku a hmotnost. Body v téměř všech grafech v 5. a 6. řádku (respektive sloupci), tedy v grafech týkajících se výšky a šířky ryb, se zdají být rozděleny do několika skupin (shluků). Je možné, že jednotlivé shluky představují jednotlivé druhy ryb. V tom případě by zkoumané závislosti byly různé pro různé druhy ryb. O tom, zda jednotlivé shluky bodů odpovídají jednotlivým druhům, se přesvědčíme, když druhovou příslušnost v grafech odlišíme barevně tak, jako jsme to učinili u grafů závislosti hmotnosti na délce a závislosti šířky na výšce. *Mathematica* bohužel nemá příkaz, kterým bychom toto barevné odlišení snadno provedli. Jak v takovém případě postupovat si pro přehlednost ukážeme nikoliv na šestici, ale na trojici sledovaných znaků – celkové délce, výšce a šířce.

Budeme postupovat tak, že postupně vykreslíme sedm grafů – pro každý druh jeden, specifické barvy – a ty pak „spojíme“ pomocí příkazu **Show[]** dohromady. Abychom toto spojení mohli provést, musíme zaručit, že všech sedm grafů bude mít stejné měřítko všech os. To zajistíme tak, že najdeme „meze“ určující, pro jaké rozsahy hodnot se mají grafy vykreslovat, a tyto maze pak použijeme při volání funkce **PairwiseScatterPlot[]**. Prvním krokem našeho postupu je tedy vymezení rozsahů hodnot, pro které se mají sledované znaky vykreslovat:

```
In[245]:= d = ryby[[All, {4, 7, 8}]];

In[246]:= m = Map[Min, Transpose[d]]
          M = Map[Max, Transpose[d]]

Out[246]= {7.5, 14.5, 8.7}

Out[247]= {59., 44.5, 20.9}

In[248]:= meze = Transpose[{m, M}]

Out[248]= {{7.5, 59.}, {14.5, 44.5}, {8.7, 20.9}}
```

Proměnná meze tedy obsahuje tři dvojice. První dvojice určuje rozmezí hodnot pro znak celková délka, druhá pro znak výška a třetí pro znak šířka.

Vymezení rozsahu vykreslovaných hodnot je nutné. Kdybychom jej totiž neučinili, grafy by nebyly porovnatelné. Například celková délka u cejnů se pohybuje od 12 cm do 44,5 cm. Kdybychom do tohoto obrázku chtěli dokreslit další hodnoty, byly by některé z nich (například o největších štikách, jejichž délka přesahuje 60 cm) mimo graf. Když dopředu určíme, že se hodnoty celkové délky pohybují v rozmezí 8,8 až 68,0 cm, žádnou z hodnot „neztratíme“.

Zkušebně si můžeme vykreslit námi požadovaný graf pro jeden konkrétní druh ryby, například pro cejna:

```
In[249]:= PairwiseScatterPlot[Ryby[[1, All, {4, 7, 8}]],
            PlotStyle → Hue[1], DataRanges → meze, DataTicks → Automatic,
            DataLabels → {"délka [cm]", "výška [%]", "šířka [%]"},
            BaseStyle → {FontSize → 13}];
```

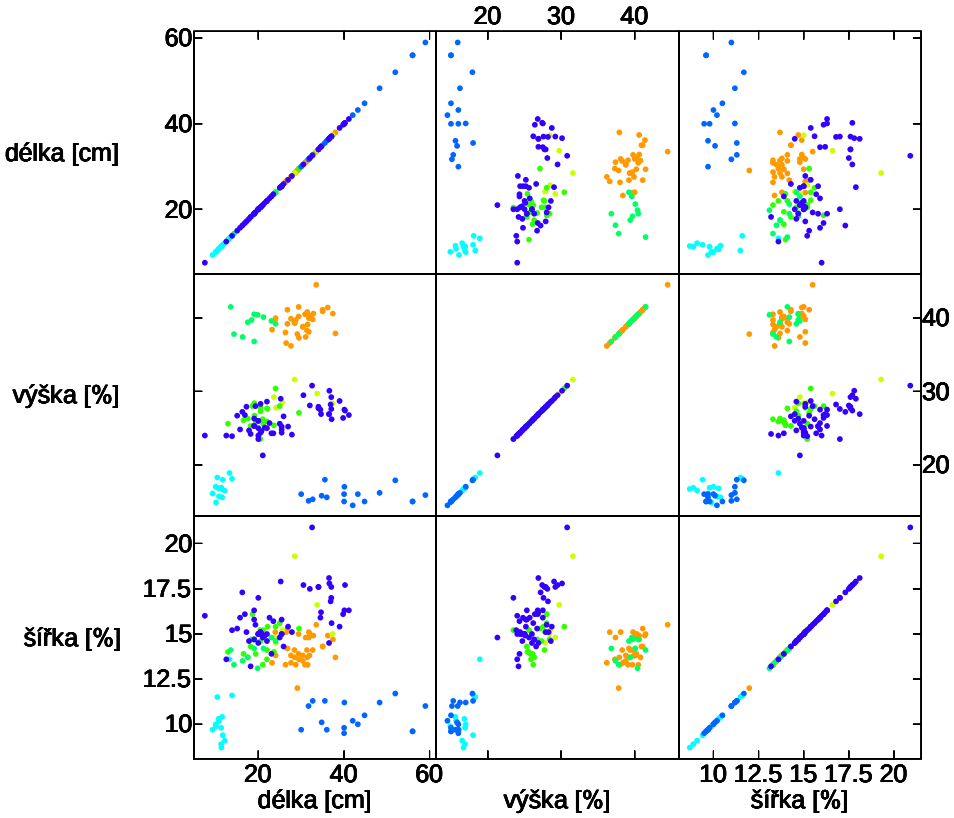
Výstup z důvodu úspory místa v tištěné podobě nezobrazujeme. Povšimněte si způsobu určení vykreslovaných oblastí pomocí volby *DataRanges→meze*. Novinkou je pro nás rovněž volba *PlotStyle→ Hue[1]* určující barvu vykreslovaných bodů (viz nápověda k příkazu **Hue[]**). Pomocí volby *DataTicks→Automatic* pak do grafů doplníme číselné popisy os.

Nyní už můžeme pomocí příkazu **Table[]** výše uvedený graf vykreslit pro všech sedm druhů. Výsledných sedm grafů uložíme pod názvem casti a poté je spojíme příkazem **Show[]** dohromady:

In[250]:=
```
casti = Table[PairwiseScatterPlot[Ryby[[i, All, {4, 7, 8}]],
    PlotStyle → Hue[0.1 * i], DataRanges → meze, DataTicks → Automatic,
    DataLabels → {"délka [cm]", "výška [%]", "šířka [%]"},
    BaseStyle → {FontSize → 13}], {i, 1, 7}];
```

In[251]:=
```
Show[Table[casti[[j]], {j, 1, 7}]]
```

Out[251]=



Jeden z vykreslených grafů jsme viděli už dříve. Který to byl, nechť určí čtenář jako cvičení. Rozdíl je pouze v použitých barvách a symbolech.

Rovněž jako cvičení je ponechán komentář tohoto obrázku. Tak, jako jsme komentovali závislost šířky na výšce (u jednotlivých druhů), můžeme nyní komentovat závislost výšky na délce a závislost šířky na délce. Kterou závislost byste vybrali, kdybyste podle ní měli rozhodovat o druhové příslušnosti? Změnily dvě nově sledované závislosti (délka-výška, délka-šířka) nějak naši představu o tvaru a velikosti ryb sledovaných druhů? Doporučujeme čtenáři promyslet odpovědi na tyto otázky a případně pokládat si otázky další.

3.5 Dodatek

Zdrojový kód k obrázku obsahujícímu srovnání proložení dat exponenciální křivkou, parabolou a logistickou křivkou.

In[252]:=
```
model1 = LinearModelFit[DelkaHmotnost, {x, x^2}, x];
model2 = NonlinearModelFit[DelkaHmotnost, a * b^x + c, {a, b, c}, x];
model3 = a / (1 + Exp[-b * (x - c)]) + k;
ParametryModelu3 = FindFit[DelkaHmotnost, model3,
   {{a, 1500}, {b, 0.2}, {c, 50}, {k, 100}}, x];
```

In[256]:=
```
g0 = ListPlot[DelkaHmotnost, PlotMarkers → {Automatic, 6},
    BaseStyle → {FontSize → 10}, PlotRange → {All, {0, 1800}}];
g1 = Plot[model1[x], {x, 30, 60}, PlotStyle → {Thick, Red}];
g2 = Plot[model2[x], {x, 30, 60}, PlotStyle → {Thick, Red}];
g3 = Plot[Evaluate[model3 /. ParametryModelu3],
    {x, 30, 60}, PlotStyle → {Thick, Red}];
```

In[260]:=
```
{Show[g0, g2, g0, PlotLabel → "Proložení exponenciální křivkou"],
  Show[g0, g1, g0, PlotLabel → "Proložení parabolou"],
  Show[g0, g3, g0, PlotLabel → "Proložení logistickou křivkou"]};
```

4 Závěr

Anotace

Publikace je věnována především těm, kteří se chtějí pustit do analýzy dat ve výpočetním prostředí Mathematica, může však posloužit i těm, kteří tento software k dispozici nemají či nechtějí využívat, ale chtějí se dozvědět něco o jednoduchých metodách popisné statistiky. Čtenář se dozví, jaké prostředky (číselné charakteristiky, grafy či tabulky) volit pro popis rozdělení jednoho, dvou či více sledovaných znaků, ať již kvalitativních či kvantitativních.

Závěr

V rámci svého (stále ještě poměrně krátkého) působení na vysoké škole se pravidelně ujímám role vedoucího či oponenta bakalářských a diplomových prací. Role oponenta mi umožňuje bez zaujatosti hodnotit výsledky samostatné práce studentů. Často přitom vidím, že student vynaložil velké úsilí k tomu, aby pochopil nějaký poměrně složitý model, který statistika používá ke zjednodušení popisu světa. Zdánlivý protimluv v předešlé větě „složitý model ke zjednodušení" není přitom nepochopitelný. Zjednodušovat opravdu nemusí být jednoduché. Zjednodušení totiž nesmí být přílišné, abychom neopomenuli něco podstatného. Přitom by výsledek měl být opravdu „jednoduše pochopitelný" – měl by být přístupný i těm, kteří nevědí, jakou cestou se k zjednodušení dojde.

Úsilí, které je nutno při studiu statistiky vynaložit, je tedy značné. Někdy však jako by toto úsilí bylo spíše studentům na škodu – příliš se soustředí na „technické detaily" a zapomínají na cíl celé analýzy, jímž je zjednodušení (či zpřehlednění) informace o studovaném problému ukryté v číslech – vstupních datech.

Často by přitom stačilo málo, aby vynaložené úsilí nepřišlo na zmar. Stačí si hned na začátku analýz udělat základní představu o tom, jaké informace jsou v datech skryty, a to pomocí jednoduchých popisných statistik, přehledných grafů či tabulek. O nich především pojednává tato publikace. Výsledky získané komplikovanějšími postupy je pak vhodné konfrontovat se základní představou získanou jednoduchými postupy. Složité analýzy často „jen" potvrdí to, co jsme už tušili, ať už na základě intuice, zkušenosti či jednoduchých analýz. Popisná statistika a „zdravý (selský) rozum" mají v procesu analýzy dat nezastupitelnou roli.

Summary

Working on a position of a teaching assistant at the University I am regularly involved as a supervisor or an opponent at defense of students' thesis of bachelor or master degree. As an opponent I am able to evaluate the results of student's independent research works. Doing that I observe quite often that students make every effort to master some rather complicated model used by statistics to simplify the description of reality. The apparent contradiction mentioned above, namely, "complex model used to simplification" is not incomprehensible. Simplification indeed does not have to be simple to do. Simplification must not be excessive, otherwise something essential might be missed. However the result should be "easy to understand"- it should be accessible even to those who do not know by what way the simplification is being achieved.
Studying statistics requires tremendous efforts to be made. But sometimes the effort has a negative

impact on students - they focus their minds on the so called “technical details” and forget the aim of the analysis, that is to simplify (or to make more transparent) the information contained in data.
Quite often just a small change in the approach can prevent the waste of the students ‘efforts. It is enough to gain the basic idea of the information contained in the data at the very beginning of the analysis. It can be done by such simple methods as computing simple characteristics, plotting transparent graphs and making simple tables. This is the core idea of this research work. The results obtained by more complex analyses should be always compared to the basic idea that has been gained by these simple methods. The results of the complex analyses often just confirm the information gained by intuition or some simple methods. Descriptive statistics and “common sense” play a key role in the process of data analysis.

5 Použitá literatura

[1] Tvrdík, J.: Medici, lékaři a statistika, Informační bulletin České statistické společnosti, 2012, roč. 23, č. 3, s.74-78.

[2] Taleb, N. N.: Černá labuť: Následky vysoce nepravděpodobných událostí. Praha: Paseka, 2011. 478 s.

[3] Hron, K., Kunderová, P.: Základy počtu pravděpodobnosti a metod matematické statistiky. Olomouc: Univerzita Palackého v Olomouci, 2013. 330 s.

[4] Zvára, K.: Regrese. Praha: Matfyzpress, 2008. 254 s.

[5] Anděl, J.: Základy matematické statistiky. Praha, Matfyzpress, 2005. 360 s.

Mgr. Ondřej Vencálek, Ph.D.

Základy analýzy dat v softwaru Mathematica

Výkonný redaktor Prof. RNDr. Zdeněk Dvořák, DrSc., Ph.D.
Odpovědná redaktorka Mgr. Jana Kreiselová
Technická redakce autor

Určeno pro studenty a akademické pracovníky VŠ

Vydala a vytiskla Univerzita Palackého v Olomouci
Křížkovského 8, 771 47 Olomouc
www.upol.cz/vup
e-mail: vup@upol.cz

Olomouc 2015
1. vydání

z. č. 2015/0081

Edice - Odborné publikace

ISBN 978-80-244-4474-1

Neprodejná publikace