Okruh 1 - Analýza a vizualizace dat

Hana Drdlová,
Jakub Chalmovianský,
Jakub Moučka

2. prosince 2020

Kapitola 1

Analýza a vizualizace dat v Matlabu

V této kapitole se blíže seznámíme s datovými strukturami *table* a *timetable*; podíváme se na široké možnosti exportu a importu dat do textových i tabulkových souborů; představíme si, co nám Matlab nabízí z oblasti základní explorativní analýzy, a také jak se lze vypořádat s chybějícími či odlehlými hodnotami. Přidány jsou také užitečné funkce pro analýzu a vizualizaci dat. Pro grafická zobrazení jsou doplněny funkce, které umožňují vytvářet pokročilejší grafické výstupy. Průběžně je teorie doplněna řešenými příklady a na závěr kapitoly jsou k dispozici neřešené příklady k procvičení.

1.1 Pokročilejší import/export dat v Matlabu

Před samotným exportem a importem datových souborů, se blíže podíváme na datovou tabulku *table*, její výhody vzhledem k obyčejnému poli *array*. Dále se seznámíme s časově orientovanou tabulkou *timetable* a také s možností transformace.

Table

Vytvoříme datovou tabulku, která obsahuje 6 pozorování a 5 proměnných neboli sloupečků různých datových typů (např. *datum_narozeni* je datový typ *datetime*, *cas* je double, *povolání* je *cell array*. Ukázka datové tabulky je zachycena v tabulce 1.1, povšimněme si kódu, kterým je tato tabulka definována, zvláště na část *VariableNames*, neboť v názvu proměnné nesmí být použita mezera. Doporučujeme nahradit podtržítkem, viz tabulka 1.1.

Pokud bychom chtěli nastavit jednotky proměnné *cas* využijeme příkaz *Properties.VariableUnits*.

```
>> T = table(datetime([30558:500:33058]','ConvertFrom',
'excel','Format','dd-MMM-yyyy'),[1:6]',{'serizovac';
'operator';'stazista';'asistent';'konstrukter';'dělník'},
[40:-2:30]',{'muz';'zena';'zena';'muz';'zena';'muz'},
'VariableNames',{'datum_narozeni';'cas';'povolani';'vek';
'pohlavi'});
>> T.Properties.VariableUnits{'cas'} = 'min';
```

datum_narozeni	cas	povolani	vek	pohlavi
30-Aug-1983	1	'serizovac'	40	'muz'
11-Jan-1985	2	'operator'	38	'zena'
26-May-1986	3	'stazista'	36	'zena'
08-Oct-1987	4	'asistent'	34	'muz'
19-Feb-1989	5	'konstrukter'	32	'zena'
04-Jul-1990	6	'delnik'	30	'muz'

Tabulka 1.1: Ukázka datové tabulky

V dalším kroku se podíváme na základní operace s datovou tabulkou. Nejdříve bychom chtěli zjistit, zda se opravdu jedná o datovou tabulku. Odpověď získáme příkazem *istable*, výstupem je logická hodnota. Počet řádků dostaneme funkcí *height* a počet sloupců prostřednictvím *width*, pro obsáhlý soubor se může hodit vypsání několika prvních, *head*, nebo posledních, *tail*, řádků tabulky, defaultně je vypsáno 8 řádků.

```
>> istable(T);
ans =
    1
>> height(T),width(T);
ans =
    6
ans =
    5
>> head(T); % vypise prvnich 8 radku
>> tail(T,3); % vypise posledni 3 radky
```

Ukážeme si několik operací související s datovou tabulkou, doplnění dalšího sloupce s údaji na požadovanou pozici v tabulce, pojmenování řádků tabulky, výběr specifických řádků či sloupců apod. Nejdříve je za sloupec *pohlavi* doplněn sloupec *vyrobky*. Důležité je zachovat stejný počet řádků jako má původní datová tabulka. Následně je vytvořen sloupcový vektor *Prijmeni*, kterým

pomocí vlastností tabulky definujeme názvy řádků (*Properties.RowNames*). Díky tomu se budeme moci k jednotlivým řádkům jednoduše odkazovat pomocí hodnot tohoto vektoru *Prijmeni*. Nový sloupec může být také vytvořen kombinací z již stávajících proměnných, viz sloupec *vykon*. Výsledná podoba tabulky je uvedena v tabulce 1.2.

```
T2 = addvars(T,(25*sin(0.5:0.5:3))','After','pohlavi',
    'NewVariableNames',{'vyrobky'});
Prijmeni = {'Novak';'Novakova';'Fialova';'Kroupa';
    'Kozarova';'Sedlak'};
T2.Properties.RowNames = Prijmeni;
T2.vykon = (T2.vyrobky)./(T2.cas);
T2.Properties.VariableUnits{'vykon'} = 'pocet vyrobku/min';
```

	datum_narozeni	**cas**	**povolani**	**vek**	**pohlavi**	**vyrobky**	**vykon**
'Novak'	30-Aug-1983	1	'serizovac'	40	'muz'	11.986	11.986
'Novakova'	11-Jan-1985	2	'operator'	38	'zena'	21.037	10.518
'Fialova'	26-May-1986	3	'stazista'	36	'zena'	24.937	8.3125
'Kroupa'	08-Oct-1987	4	'asistent'	34	'muz'	22.732	5.6831
'Kozarova'	19-Feb-1989	5	'konstrukter'	32	'zena'	14.962	2.9924
'Sedlak'	04-Jul-1990	6	'delnik'	30	'muz'	3.528	0.588

Tabulka 1.2: Přidání sloupců do datové tabulky

Detailnější informace, základní popisné statistiky jednotlivých sloupců, čili proměnných tabulky, získáme příkazem *summary*. Podoba výstupů tohoto příkazu pro uvažovanou tabulku je zobrazena v tabulce 1.3.

```
>> summary(T2);
ans =
```

datum_narozeni	6x1 datetime	***cas***	6x1 double
Values:		Properties: Units: min.	
Min	30-Aug-1983	Min	1
Median	31-Jan-1987	Median	3.5
Max	4-Jul-1990	Max	6

vek	6x1 double	***vyrobky***	6x1 double	***vykon***	6x1 double
Values:		Values:		Units:	
Min	30	Min	3.528	Min	0.588
Median	35	Median	17.999	Median	6.9978
Max	40	Max	24.937	Max	11.986

povolani	***pohlavi***
6x1 cell array of character vectors	6x1 cell array of character vectors

Tabulka 1.3: Základní popisné statistiky

V případě, že se nám s datovou tabulkou bude lépe pracovat, pokud bude transponovaná, tedy názvy řádků se stanou názvy sloupců a naopak, lze

využít příkaz *rows2vars*. Tuto změnu části datové tabulky 1.2 lze pozorovat v tabulce 1.4. Pro transponování jsou na ukázku použity první 4 řádky každého sloupce.

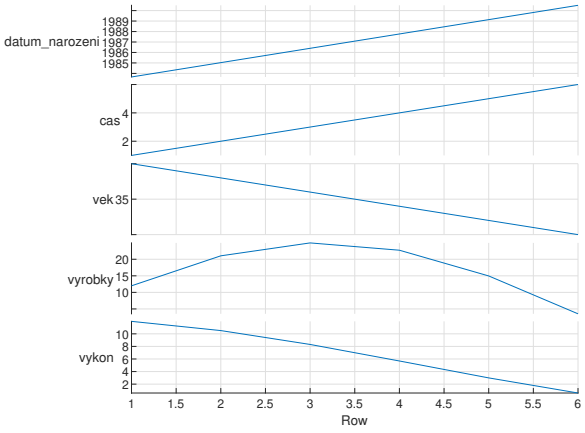
```
>> rows2vars(T2(1:4,:))
```

Names	**Novak**	**Novakova**	**Fialova**	**Kroupa**
'datum_narozeni'	[30-Aug-1983]	[11-Jan-1985]	[26-May-1986]	[08-Oct-1987]
'cas'	[1]	[2]	[3]	[4]
'povolani'	'serizovac'	'operator'	'stazista'	'asistent'
'vek'	[40]	[38]	[36]	[34]
'pohlavi'	'muz'	'zena'	'zena'	'muz'
'vyrobky'	[11.9856]	[21.0368]	[24.9374]	[22.7324]
'vykon'	[11.9856]	[10.5184]	[8.3125]	[5.6831]

Tabulka 1.4: Transponování části datové tabulky *T2*

Dalším užitečným příkazem je funkce *stackedplot*, která číselné proměnné interaktivně zobrazí nad sebe se společnou osou x. Pro lepší orientaci je při vykreslení aktivováno zobrazení mřížky pomocí *grid on*. Grafické vykreslení se nachází na obrázku 1.1.

```
>> stackedplot(T2)
>> grid on
```



Obrázek 1.1: Vertikální vykreslení proměnných se společnou osou x

Dále je ukázán příkaz, kterým se odkážeme na první 3 řádky sloupce *povolani*, výstup je možný nalézt v levé části tabulky 1.5. Využijeme pojmenované řádky a odkážeme se pomocí příjmení. Nejdříve v pravé části tabulky 1.5 jsou pro pana Nováka vybrány pouze 2 sloupce, a to *vek* a *vykon*. V tabulce 1.6 jsou vybrány pro pana a paní Novákovi všechny sloupce. Demonstrativně je předveden příkaz pro seřazování sestupně volbou *descend*, report se nachází v tabulce 1.7.

```
>> T2.povolani(1:3)
ans =
>> T2('Novak',{'vek','vykon'})
ans =
```

'serizovac'
'operator'
'stazista'

	vek	**vykon**
Novak	40	11.986

Tabulka 1.5: Odkazování na datovou tabulku

```
>> T2({'Novak','Novakova'},:)
ans =
```

	datum_narozeni	**cas**	**povolani**	**vek**	**pohlavi**	**vyrobky**	**vykon**
'Novak'	30-Aug-1983	1	'serizovac'	40	'muz'	11.986	11.986
'Novakova'	11-Jan-1985	2	'operator'	38	'zena'	21.037	10.518

Tabulka 1.6: Výběr pana a paní Novákových

```
>> sortrows(T2({'Novakova','Fialova','Kozarova'},:),
'vyrobky','descend')
ans =
```

	datum_narozeni	**cas**	**povolani**	**vek**	**pohlavi**	**vyrobky**	**vykon**
'Fialova'	26-May-1986	3	'stazista'	36	'zena'	24.937	8.3125
'Novakova'	11-Jan-1985	2	'operator'	38	'zena'	21.037	10.518
'Kozarova'	19-Feb-1989	5	'konstrukter'	32	'zena'	14.962	2.9924

Tabulka 1.7: Seřazený výběr z datové tabulky

Cvičení 1.1

Řešenými úkoly si zkusíme odpovědět na následující otázky:

- Kdo vystupuje nejvyšším výkonem mezi zaměstnanci?
- Která žena je mladší než medián všech zaměstnanců?

Řešení:
Při vybírání řádků, které splňují námi definované podmínky, využijeme známé statistické funkce *max* (z podkapitoly 2.2 v Attaway, 2014) a *median* (z podkapitoly 14.1 v Attaway, 2014). Nejdříve hledáme ve sloupci *vykon* nejvyšší hodnotu, poté vybereme z tabulky *T2* příslušný řádek se všemi sloupci. Odpověď na nejvyšší výkon ze zaměstnanců nalezneme v tabulce 1.8. Možná nečekané je využití funkce *strcmp*, která porovnává textové řetězce (pro detailnější informace viz. Attaway (2014), část 7.2.5). Ze sloupce *pohlavi* jsou pomocí funkce *strcmp* vybrány pouze ženy. Pro porovnávání je použit relační operátor (<=). Na závěr je z tabulky *T2* vybrán řádek splňující obě podmínky zároveň (logický operátor &). Odpověď na otázku, která žena je mladší než medián všech zaměstnanců, se nachází v tabulce 1.9.

```
>> T2.vykon == max(T2.vykon);
>> T2(T2.vykon == max(T2.vykon),:);
ans =
```

	datum_narozeni	cas	povolani	vek	pohlavi	vyrobky	vykon
'Novak'	30-Aug-1983	1	'serizovac'	40	'muz'	11.986	11.986

Tabulka 1.8: Nejvyšší výkon ze zaměstnanců

```
>> T2(T2.vek <= median(T2.vek) & strcmp(T2.pohlavi,
    'zena'),:);
ans =
```

	datum_narozeni	cas	povolani	vek	pohlavi	vyrobky	vykon
'Kozarova'	19-Feb-1989	5	'konstrukter'	32	'zena'	14.962	2.9924

Tabulka 1.9: Žena, jež je mladší než medián všech

Timetable

Nyní budeme pracovat s časově orientovanou datovou tabulku, pro níž je povinná proměnná obsahující časový údaj prezentující se datovým typem *datetime* nebo *duration*. Proto využijeme proměnnou *datum_narozeni*. Výsledek je zobrazen v tabulce 1.10. Příkazy, které byly ukázány v případě *table* pro získání základních informací, platí také pro *timetable*.

datum_narozeni	Row	cas	povolani	vek	pohlavi	vyrobky	vykon
30.August.1983	'Novak'	1	'serizovac'	40	'muz'	11.986	11.986
11-Jan-1985	'Novakova'	2	'operator'	38	'zena'	21.037	10.518
26-May-1986	'Fialova'	3	'stazista'	36	'zena'	24.937	8.3125
8-Oct-1987	'Kroupa'	4	'asistent'	34	'muz'	22.732	5.6831
19-Feb-1989	'Kozarova'	5	'konstrukter'	32	'zena'	14.962	2.9924
4-Jul-1990	'Sedlak'	6	'delnik'	30	'muz'	3.528	0.588

Tabulka 1.10: Časově orientovaná tabulka

```
>> Ttime = table2timetable(T2,'RowTimes','datum_narozeni');
Ttime =
```

Při odkazování na sloupec, či řádek, postupujeme analogicky. Pro odkazování na řádky můžeme navíc využít časové proměnné, kdy zadáme přibližný rozsah pomocí příkazu *timerange*, viz tabulka 1.11. Kromě pravidla pro výběr řádků si můžeme také určit, které sloupce se vypíší. V posledním kroku je náš výběr vzestupně seřazený podle proměnné *vyrobky*.

```
Ttime.povolani;
TR = timerange('1983-01-01','1986-12-31');
time_choice = Ttime(TR,{'Row','cas','povolani','vek',
    'vyrobky'});
Ttime_sort = sortrows(time_choice,'vyrobky');
```

datum_narozeni	Row	cas	povolani	vek	vyrobky
30-Aug-1983	'Novak'	1	'serizovac'	40	11.986
11-Jan-1985	'Novakova'	2	'operator'	38	21.037
26-May-1986	'Fialova'	3	'stazista'	36	24.937

Tabulka 1.11: Seřazený časový výběr

Na dalších řádcích si ukážeme, jak lze vybrat jednu profesi ze všech povolání zaměstnanců. Např. *stazista* je vybrán tímto způsobem, import je k dispozici v tabulce 1.12.

```
>> Ttime_sort(strcmp(Ttime_sort.povolani,'stazista'),:);
ans =
```

datum_narozeni	Row	cas	povolani	vek	vyrobky
26-May-1986	'Fialova'	3	'stazista'	36	24.937

Tabulka 1.12: Výběr podle povolání

Pro transformaci časově orientované tabulky zpátky na obyčejnou tabulku slouží příkaz *timetable2table*.

```
>> timetable2table(Ttime)
```

Cvičení 1.2

V řešeném příkladu budeme postupovat velmi podobně, úkolem je vytvořit datovou tabulku s jednou časovou proměnnou a poté ji převést na časově orientovanou tabulku, například to může vypadat jako v tabulce 1.13.

Řešení:

```
>> T = table(seconds(1:4)',{'ABC';'DEF';'XYZ';'JKL'},
    {'abc';'def';'xyz';'jkl'});
>> table2timetable(T,'RowTimes','Var1');
```

Var1	Var2	Var3
1 ses	'ABC'	'abc'
2 sec	'DEF'	'def'
3 sec	'XYZ'	'xyz'
4 sec	'JKL'	'jkl'

Var1	Var2	Var3
1 ses	'ABC'	'abc'
2 sec	'DEF'	'def'
3 sec	'XYZ'	'xyz'
4 sec	'JKL'	'jkl'

Tabulka 1.13: *table* vs. *timetable*

Export dat

V této části kapitoly se budeme věnovat exportu dat. Vzhledem ke skutečnosti, že Matlab přestává podporovat příkazy *xlsread* a *xlswrite*, budou představeny nahrazující funkce. Nejdříve se podíváme na datovou tabulku 1.2 *(T2)*, kterou jsme si vytvořili výše v části 1.1. Ukážeme si, jak ji lze exportovat, a to buď jako textový soubor, u kterého lze volit oddělovač *(Delimiter)*; nebo jako tabulkový soubor Excelu, kde můžeme navíc zvolit list (*Sheet*) a také rozsah polí (*Range*), do kterých bude tabulka vepsána.

```
>> writetable(T2,'table.txt','Delimiter',';')
>> writetable(T2,'table.csv','Delimiter',' ')
>> writetable(T2,'table.xlsx','Sheet',1,'Range','B2:I8')
```

Cvičení 1.3

V řešeném příkladu bude vyzkoušena funkce *writetimetable*, která pracuje analogicky jako funkce *writetable*. Úkolem je exportovat již vytvořenou časově orientovanou datovou tabulku 1.10 *(Ttime)*, vytvořit textový soubor *.csv*, *.txt* s použitím různých oddělovačů a soubor Excelu, u kterého bude vepsána tabulka na 3. list.

Řešení:

```
>> writetimetable(Ttime,'timetable.txt','Delimiter',',')
>> writetimetable(Ttime,'timetable.csv','Delimiter',';')
>> writetimetable(Ttime,'timetable.xlsx','Sheet',3)
```

Cvičení 1.4

Při exportu matice postupujeme obdobně. Vytvořme si libovolnou matici, kterou exportujeme obdobným způsobem pomocí funkce *writematrix* jako *.csv* a *.xlsx*, u něhož si připomeneme volbu listu a také rozsah buněk, do kterých bude matice zapsána.

Řešení:

```
>> M = randn(4,5);
>> writematrix(M,'matrix.csv','Delimiter',';')
>> writematrix(M,'matrix.xlsx','Sheet',2,'Range','C3:G7')
```

Export polí s různými datovými typy *(cell)* provedeme prostřednictvím funkce *writecell*. Podívejme se na příklad z tabulky 1.14:

```
>> C = {4,52,93;'Tesla',datetime('today'),hours(15)};
```

{[4]}	{[52]}	{[93]}
{'Tesla'}	{[08-Sep-2020]}	{[15 hr]}

Tabulka 1.14: Pole s různými datovými typy

```
>> writecell(C)
>> writecell(C,'C_comma.txt','Delimiter',',')
>> writecell(C,'C.xlsx')
```

Import dat

Vytvořené soubory se pokusíme importovat. Nejdříve začneme s datovou tabulkou *(T2)*:

```
>> T1 = readtable('table.txt');
>> T2 = readtable('table.csv');
```

Volbou *ReadRowNames* **true** je možné použít informace z prvního řádku importovaného souboru pro názvy proměnných, tzv. *VariableNames*. Ze zdrojového souboru lze při importu vybrat část dat pomocí *range*, ukázka aplikace uvedených možností se nachází v tabulce 1.15.

```
>> T3 = readtable('table.xlsx',
'ReadVariableNames',false,'ReadRowNames',true,'range',
'B2:E8');
```

	cas	povolani	vek
30.08.1983	1	'serizovac'	40
11.01.1985	2	'operator'	38
26.05.1986	3	'stazista'	36

Tabulka 1.15: Importovaná část datové tabulky

Při importu časově orientované datové tabulky povinným argumentem *RowTimes* nastavíme proměnnou s časovým údajem.

```
>> TT1 = readtimetable('timetable.csv','RowTimes',
'datum_narozeni')
>> TT2 = readtimetable('timetable.txt','RowTimes',
'datum_narozeni')
>> TT3 = readtimetable('timetable.xlsx','RowTimes',
'datum_narozeni','Sheet',3)
```

Cvičení 1.5

Importujme do Matlabu datovou matici, kterou jsme exportovali ve cvičení 1.4, ze souborů *.csv* a *.xlsx* pomocí funkce *readmatrix*. Nezapomeňme, že v případě *.xlsx* souboru se matice nachází na 2. listu.

Řešení:

```
>> M1 = readmatrix('matrix.csv');
>> M2 = readmatrix('matrix.xlsx','Sheet',2);
```

Na závěr se podívejme na import pole buněk různých datových typů funkcí *readcell*. Jelikož jsme v části 1.1 pro export dat vytvořili pole buněk C, které obsahuje i datumovou informaci (buňka s dnešním datumem), je nutné použít volbu *DatetimeType*, kterou definujeme formát datumu, tak aby byl správně z Excelu přečten.

```
>> C1 = readcell('C.txt');
>> C2 = readcell('C.xlsx','Range','A1:C2','DatetimeType',
'text');
>> C3 = readcell('C_comma.txt','Delimiter',',');
```

1.2 Základní explorativní analýza a zpracování dat

V této části se zaměříme na funkce, které nám pomohou při seznámení s datovým souborem. Budeme tak hledat odpovědi na otázky jako kupříkladu: vyskytují se v datovém souboru chybějící hodnoty, odlehlé hodnoty, zajímat nás bude také problematika základních popisných charakteristik zkoumaných dat.

Pro názornost bude dále zpracováván datový soubor *USvideos.csv*, který obsahuje cca 4000 pozorování pro 16 proměnných. Jedná se o seznam nejpopulárnějších videí na platformě Youtube s proměnnými počet zhlédnutí, sdílení, komentářů atd. Seznamme se nejprve s datovým souborem tím, že se podíváme na nějakou jeho část. Využijeme funkci *head*, která nám zobrazí obsah několika prvních řádků zvoleného datového souboru.

```
>> T = readtable('USvideos.csv');
>> head(T,5);
```

Chybějící hodnoty

K vytvoření chybějící hodnoty slouží funkce *missing*. Využití se například nabízí v situaci, pokud se domníváme o některých pozorováních v datovém souboru, že jsou chybná. Taková pozorování budeme chtít vyloučit, tedy nahradit chybějícími pozorováními.

```
>> a = 1:5;
a = 1 2 3 4 5
>> a(2) = missing;
a = 1 NaN 3 4 5
```

Nyní budeme pracovat s datovým souborem *USvideos.csv*, který máme uložený v proměnné *T*. Pro nalezení chybějících hodnot lze použít funkci *ismissing*, jejíž výstupem je datová matice logických hodnot stejné velikosti jako zdrojový soubor. Pro přehlednost nás nejspíše bude zajímat počet chybějících hodnot v každém řádku

```
>> ismissing(T);
>> sum(ismissing(T),2);
```

či sloupci

```
>> sum(ismissing(T),1);
```

ekvivalentně

```
>> sum(ismissing(T))
ans =
    21 21 22 23 24 24 26 26 26 26 26 26 26 26 26 112
```

V některých případech je kromě chybějící položky za chybějící hodnotu chápána i jiná hodnota např. 0, záporná hodnota apod., takovou situaci si zde ukážeme. (Nyní jsou detekovány pouze nulové hodnoty, hodnoty NaN z předchozího kroku nejsou uvažovány.)

```
>> ismissing(T,0);
>> sum(ismissing(T,0));
ans =
    0 0 0 0 0 0 0 0 28 77 111 0 0 0 0 0
```

Funkce *standardizeMissing* vkládá chybějící hodnotu za námi zvolenou hodnotu, v našem případě 0. Čili nejdříve získáváme počet skutečně chybějících hodnot, ve 2. řádku jsou počty vyskytujících se nulových položek a ve 3. řádku je jejich součet, čili chybějících a nulových, viz tabulka 1.16.

```
sum(ismissing(T));
sum(ismissing(T,0));
standardizeMissing(T,0);
sum(ismissing(standardizeMissing(T,0)));
```

21	21	22	23	24	24	26	26	26	26	26	26	26	26	26	112
0	0	0	0	0	0	0	0	28	77	111	0	0	0	0	0
21	21	22	23	24	24	26	26	54	103	137	26	26	26	26	112

Tabulka 1.16: Součty chybějících hodnot

Pro odstranění celých pozorování či proměnných s chybějícími hodnotami můžeme využít funkci *rmmissing*, v tabulce dojde k odstranění celých řádků, volbou dimenze 2 budou odstraněny sloupce. *B* či *B2* jsou logické vektory hodnot, které indikují zda došlo k odstranění řádků/sloupců. Volbou *MinNumMissing* lze zvolit minimální počet chybějících hodnot pro odstranění řádku, na závěr jsou přidány součty chybějících hodnot v každém sloupci.

```
>> [T_new1,B] = rmmissing(T);
>> [T_new11,B2] = rmmissing(T,2);
>> sum(ismissing(T_new11));
ans =
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>> rmmissing(T,'MinNumMissing',2);
>> sum(ismissing(rmmissing(T,'MinNumMissing',2)));
ans =
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 86
```

Pro nalezení a vyplnění chybějících hodnot je využita funkce *fillmissing*, volbou *previous* je chybějící hodnota nahrazena předchozí nechybějící hodnotou. Nahrazení je také možné prostřednictvím konstanty. Pokud chceme vybrat pouze některé sloupce, použijeme volbu *DataVariables* a vypíšeme jejich názvy.

```
>> sum(ismissing(T))
ans =
    21 21 22 23 24 24 26 26 26 26 26 26 26 26 26 112
>> T_new2 = fillmissing(T,'previous');
>> sum(ismissing(T_new2))
ans =
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>> sum(ismissing(fillmissing(T,'constant',1,'DataVariables',
{'views','likes'})))
ans =
    21 21 22 23 24 24 26 0 0 26 26 26 26 26 26 112
```

Cvičení 1.6

Změňme metodu nahrazení chybějících hodnot pouze u číselných proměnných, a to lineární interpolací sousedních hodnot volbou *linear*. (Metoda umožňuje stanovit funkční hodnotu mezi dvěma body.)

Řešení:

```
>> fillmissing(T,'linear','DataVariables',@isnumeric);
>> sum(ismissing(fillmissing(T,'linear','DataVariables',
    @isnumeric)))
ans =
    21 21 22 23 24 24 26 0 0 0 0 26 26 26 26 112
```

Odlehlé hodnoty

Podívejme se na problematiku odlehlých hodnot, jejich identifikaci a následnou možnost úpravy. Připomeňme, že se jedná o hodnotu, která leží 3-5násobek výběrové směrodatné odchylky od průměru respektive mediánu. Z výše uvedeného datového souboru *USvideos.csv* bude pro ukázku použit 8. sloupec s počty shlédnutí, *views*. Nejdříve jsou pomocí funkce *isoutlier* nalezeny odlehlé hodnoty, defaultně jsou hodnoty porovnávány podle mediánové absolutní odchylky *MAD*, což lze nahradit např. průměrem. Dále lze pomocí vektoru nastavit dolní a horní percentil. Pro podrobnější informace (dolní, horní prahové hodnoty a středové hodnoty) lze využít závěrečného zápisu, jeho výstup najdeme v tabulce 1.17.

```
>> isoutlier(T(:,8));
>> sum(isoutlier(T(:,8)))
ans =
    645
>> isoutlier(T(:,8),'mean');
>> sum(isoutlier(T(:,8),'mean'))
ans =
    47
>> sum(isoutlier(T(:,8),'percentiles',[10,90]))
ans =
    824
>> [TF,L,U,C] = isoutlier(T(:,8))
```

```
L,U,C =
```

views	views	views
-9.3255e+05	1.5784e+06	3.2293e+05

Tabulka 1.17: Dolní a horní prahová hodnota a středová hodnota

Pro odhalení a odstranění odlehlých hodnot je vhodné použít funkci *rmoutliers*. Bude využit rovněž 8. sloupec *views* avšak poté, co byly odstraněny chybějící hodnoty. Vektor B je očištěn od odlehlých hodnot a logický vektor TF vyjadřuje, které řádky byly odstraněny.

```
>> sum(isoutlier(T_new1(:,8)))
ans =
    626
>> [B, TF] = rmoutliers(T_new1(:,8))
```

Podívejme se, o kolik se změnil počet odlehlých hodnot po té, co jsou odstraněny odlehlé hodnoty na základě porovnávání dle mediánové absolutní odchylky *MAD*. V našem případě počet odlehlých hodnot poklesl na 291.

```
>> sum(isoutlier(rmoutliers(T_new1(:,8))))
ans =
    291
```

Cvičení 1.7

Pokusme se změnit metodu, dle které jsou vybírány odlehlé hodnoty, úkolem je použít průměr, který však nebude ze všech hodnot, ale bude počítán v rámci posuvného okna délky 10. Využití se nabízí zejména u časových řad, kdy způsob detekování odlehlých hodnot na určité velikosti okna zohlední vývoj řady. Pro takový případ je použita funkce *movmean*, které se budeme více věnovat v části **Základní charakteristiky datového souboru**.

Řešení:

```
>> sum(isoutlier(rmoutliers(T_new1(:,8),'movmean', 10)))
ans =
    626
```

Pro odhalení a nahrazení odlehlých dat slouží funkce *filloutliers*, analogicky můžeme zvolit metodu určování odlehlé hodnoty a způsob nahrazování. Volbou *quartiles* získáme mezikvartilové rozpětí, čili je stanoven dolní (25 %) a horní (75 %) kvartil. Odlehlé hodnoty jsou nahrazeny následující hodnotou. Také je přidán součet odlehlých hodnot podle mediánové absolutní odchylky *MAD*.

```
>> T_new3 = filloutliers(T_new2(:,8),'next','quartiles');
>> sum(isoutlier(T(:,8)))
ans =
    645
>> sum(isoutlier(T_new3))
ans =
    444
```

Cvičení 1.8

Pokusíme se identifikovat odlehlé hodnoty pomocí 25% a 75% percentilu a nahradit je nejbližší neodlehlou hodnotou.

Řešení:

```
>> T_new4 = filloutliers(T_new2(:,8),'nearest','percentiles',
[25,75]);
>> sum(isoutlier(T_new4))
ans =
    0
```

Základní charakteristiky datového souboru

Datový soubor *USvideos.csv*, který v této podkapitole používáme, je nahrán jako datová tabulka. V této části budeme nadále pracovat s 8. sloupcem uvedené tabulky *views* a podíváme se na některé popisné charakteristiky těchto dat. Spojení minima a maxima (funkcí *min* a *max*, které už znáte z podkapitoly 2.2 v Attaway (2014)) představuje funkce *bounds*.

```
>> v = T.views;
>> [m,M] = bounds(v)
```

```
m =
     687
M =
     102012605
```

Pro charakteristiky minimum *(min)*, maximum *(max)*, průměr *(mean)*, medián *(median)*, součet *(sum)*, směrodatnou odchylku *(std)* a rozptyl *(var)* existuje varianta výpočtu ignorující chybějící hodnoty, a to s předponou **nan**. Volbou *omitnan* lze vyloučit a *includenan* zahrnout chybějící hodnoty v základní podobě uvedených charakteristik.

Cvičení 1.10

Spočítejme výše zmíněné charakteristiky pro 8. sloupec s údaji o počtu shlédnutí. Doplňme výsledky o výpočet absolutní odchylky *mad*, defaultně je počítána průměrová absolutní odchylka. S volbou 1 je vypočtena mediánová absolutní odchylka.

Řešení:

```
>> nanmin(v)                >> nanmax                >> nanmean(v)
ans =                       ans =                    ans =
    687                         102012605                1.2453e+06
>> find(nanmin(v)==v)       >> find(nanmax(v)==v)    >> mean(v)
ans =                       ans =                    ans =
    545                         2593                     NaN
>> min(v,[],'omitnan')      >> max(v,[],'omitnan')   >> mean(v,'omitnan')
ans =                       ans =                    ans =
    687                         102012605                1.2453e+06
   ------------------------------------------------------------

   >> nanmedian(v)          >> nansum(v)             >> nanstd(v)
   ans =                    ans =                    ans =
       322934                   5.1282e+09               4.6046e+06
   >> median(v)             >> sum(v)                >> std(v)
   ans =                    ans =                    ans =
       NaN                      NaN                      NaN
   >> median(v,'omitnan')   >> sum(v,'omitnan')      >> std(v,'omitnan')
   ans =                    ans =                    ans =
       322934                   5.1282e+09               4.6046e+06
   ------------------------------------------------------------
```

```
>> nanvar(v)
ans =
    2.1203e+13
>> var(v)
ans =
    NaN
>> var(v,'omitnan')
ans =
    2.1203e+13
```

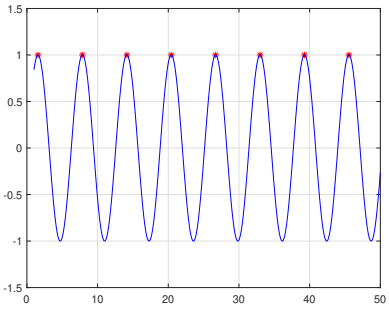
```
>> mad(v)
ans =
    1.4860e+06
>> mad(v,1)
ans =
    282271
```

K základním charakteristikám, které byly uvedeny, jsou doplněny klouzavé varianty s povinným argumentem délky posuvného okna. Na začátku a konci se provádí výpočet s menším než zadaným počtem prvků, tyto hodnoty lze vyloučit volbou *Endpoints* za **discard**. Kromě minima a maxima je vhodné přidat argument *omitnan*, abychom nedostávali *NaN*.

```
movmax(v,100)
movmin(v,200)
movmad(v,3)
movmad(v,500,'omitnan')
movmad(v,500,'omitnan','Endpoints','discard')
movmean(v,1000,'omitnan')
movmedian(v,1000,'omitnan')
movprod(v,2,'omitnan')
movstd(v,1000,'omitnan')
movsum(v,100,'omitnan')
movvar(v,1000,'omitnan')
```

Více prozkoumáme problematiku extrémů, nyní z pohledu lokálních. Funkce *islocalmax* hledá lokální maxima, jestliže je identifikováno maximum, dostáváme kladnou logickou hodnotu. Prozkoumejme tuto funkci na následujícím příkladě funkce *sin* pro číselnou řadu *rada*. Do vektoru x je uložena x-souřadnice lokálních extrémů, do vektoru y je uložena y-souřadnice. Doplněn je také součet lokálních extrémů, jednotlivé body jsou v dalším kroku na obrázku 1.2 graficky znázorněny. Odstranění samotné čáry je možné volbou *LineStyle* **none**. Příkazem *ylim* jsou nastaveny limity vertikální osy y. V případě, že do jednoho grafu přidáváme další křivky apod., je nezbytným příkazem *hold on*, neboť zachová aktuální graf pro přidávání nových grafů. Na závěr je přidána křivka funkce *sin*.

```
rada = 1:0.01:50;
fce = sin(rada);
islocalmax(fce)
x = rada(islocalmax(fce));
y = fce(islocalmax(fce));
sum(islocalmax(fce))
ans =
   8
plot(x,y,'LineStyle','none',
   'Color','red','Marker','*')
ylim([-1.5,1.5])
grid on
hold on
plot(rada,sin(rada),'Color',
   'blue')
```



Obrázek 1.2: Identifikování lokálních maxim

Pokud se nyní vrátíme k 8. sloupci s údaji o počtu shlédnutí datového souboru *USvideos.csv*, obdržíme 1388 lokálních maxim. Samotné hodnoty maxim z vektoru *v* získáme výběrem *v(islocalmax(v))*. Lze stanovit minimální hodnotu, od které se maximum začne vyhledávat prostřednictvím *MinProminence*. Počet shlédnutí od 1 milionu se vyskytuje u 658 videí.

```
>> islocalmax(v);
>> sum(islocalmax(v));
ans =
    1388
>> v(islocalmax(v));
>> sum(islocalmax(v,'MinProminence',1000000));
ans =
    658
```

Cvičení 1.11

Zjistěte hodnoty lokálních extrémů 9. sloupce *likes*, vyšetřete lokální minima podobně jako tomu bylo výše u maxima. Dále nastavte počet nalezených lokálních minim na 25 pomocí volby *MaxNumExtrema*, vypište jejich hodnoty.

```
>> w = T.likes
>> [m,M] = bounds(w)
m =
    0
M =
    2729292
>> islocalmin(w)
>> w(islocalmin(w))
>> sum(islocalmin(w))
ans =
    1385
>> islocalmin(w,'MaxNumExtrema',25)
>> w(islocalmin(w,'MaxNumExtrema',25))
ans =
0 0 22 0 0 26 0 0 0 27 0 0 0 0 0 0 0 17 0 0 19 0 0 0 0
```

Při hledání nejednoho absolutního extrému je vhodné použití funkcí *mink/ maxk*. Tyto funkce hledají k extrémů v datovém vzorku, v případě $k = 2$ *mink* vyhledá 2 globální minima. Poté pro $k = 3$ *maxk* nalezne 3 globální maxima. Volbou *ComparisonMethod* s možností **abs** jsou porovnávány hodnoty v absolutní hodnotě.

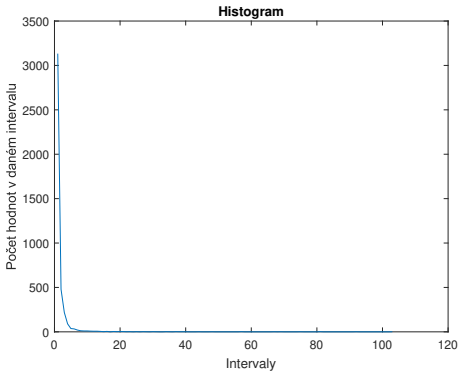
```
>> mink(v,2)
ans =
    687
    704

>> maxk(v,3)
ans =
    91552137
    80605857
    80360459
```

K bližšímu seznámení s distribucí dat v souboru můžeme využít funkci *histcounts*, N udává počet pozorování v jednotlivých intervalech, *edges* doplňuje informace o pravidelných hranicích. Je možné stanovit normované rozdělení, volbou *probability* zajistíme, že součet pozorování v třídících intervalech je menší nebo rovno 1. Grafické znázornění, které je na obrázku 1.3, získáváme příkazem *plot*. Příkazem *title* mohu graf doplnit titulkem. Ve vyšší dimenzi použijeme analogicky funkci *histcounts2*.

```
>> [N,edges] = histcounts(v);
>> [N,edges] = histcounts(v,'Normalization','probability');
>> sum(N)
ans =
    0.9937
```

```
plot(histcounts(v));
title('Histogram');
xlabel('Intervaly');
ylabel('Počet hodnot
    v daném intervalu');
```



Obrázek 1.3: Rozložení dat pomocí histogramu

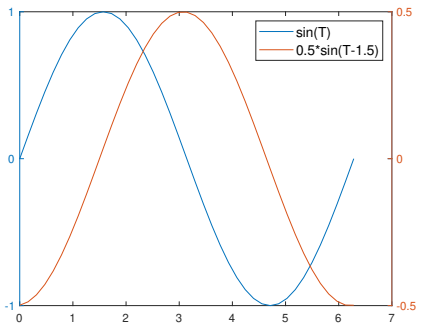
1.3 Vizualizace dat: pokročilé grafy

Tato podkapitola je věnována dodatečným užitečným funkcím, které lze v Matlabu využít k analýze a vizualizaci dat.

Matematické a statistické výstupy

Pokud je naším cílem zkonstruovat graf, který bude mít dvě osy y s různými měřítky, je možné použít funkci *plotyy*. Také je ukázáno, jakým způsobem lze přidat legendu, u které je nastavena pozice a velikost, viz obrázek 1.4.

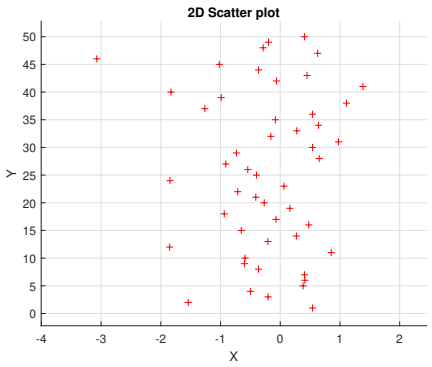
```
T = 0:pi/20:2*pi;
Y1 = sin(T);
Y2 = 0.5*sin(T-1.5);
plotyy(T,Y1,T,Y2,'plot')
legend('sin(T)',
  '0.5*sin(T-1.5)',
  'Location', 'NorthEast',
  'FontSize', 12)
```



Obrázek 1.4: Graf se dvěma osami y

Prozkoumáme nyní možnosti scatter plotů v Matlabu, po zadání 2 proměnných dostáváme dvourozměrný graf na obrázku 1.5. Příkazem *xlim/ylim* lze nastavit limity os.

```
X = randn(50,1);
Y = 1:50;
scatter(X,Y,'r+');
grid on
xlabel('X');
ylabel('Y');
xlim([min(X)-1 max(X)+1]);
ylim([min(Y)-3 max(Y)+3]);
title('2D Scatter plot');
```

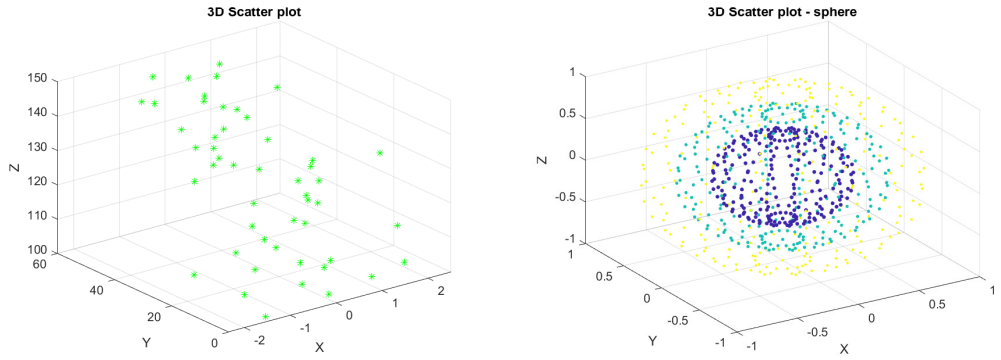


Obrázek 1.5: Dvourozměrný scatter plot

Varianta v 3-dimenzionálním prostoru vychází ze zadání 3 proměnných, názorné příklady nalezneme na obrázku 1.6. Kromě jednoduché ukázky v levé části je přidána složitější v pravé části. K rekonstrukci jsou nutné *x,y,z*-souřadnice koule, které získáme příkazem *sphere(16)*. Dojde k vytvoření matice 17x17, která je využita pro ukázku kreativního bodového diagramu. Volbou *view* lze nastavit úhel pohledu, volba *filled* způsobí, že jednotlivé body se zobrazují jako kolečka s výplní pro lepší viditelnost.

```
X = randn(50,1);
Y = 1:50;
Z = 101:150;
scatter3(X,Y,Z,'g*')
xlabel('X');
ylabel('Y');
zlabel('Z');
title('3D scatter plot');
```

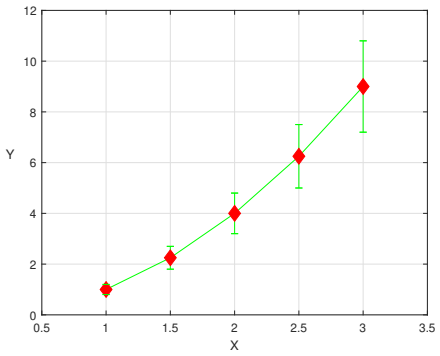
```
[x,y,z] = sphere(16);
X = [x(:)*.5 x(:)*.75 x(:)];
Y = [y(:)*.5 y(:)*.75 y(:)];
Z = [z(:)*.5 z(:)*.75 z(:)];
S = repmat([1 .75 .5]*10,prod(size(x)),1);
C = repmat([1 2 3],prod(size(x)),1);
scatter3(X(:),Y(:),Z(:),S(:),C(:),'filled')
view(-32,32);
```



Obrázek 1.6: Trojrozměrné scatter ploty

Errorbar v Matlabu vytvoříme pomocí funkce *errorbar*. Při zadání argumentu X je vykreslen symetrický *errorbar* proměnné Y v závislosti na X. Je ukázáno, jakým způsobem můžeme měnit základní nastavení, např. barvu čáry a znaku. Volbou *Marker* si zvolíme typ znaku, pomocí *MarkerSize* jeho velikost. U příkazu pro nastavení názvu osy *ylabel* lze volbou *Rotation* určit polohu názvu. Graf nalezneme na obrázku 1.7.

```
X = 1:0.5:3;
Y = X.^2;
err = Y./5;
errorbar(X,Y,err,'Color',
   'green','MarkerSize',10,
   'MarkerEdgeColor','red',
   'MarkerFaceColor','red',
   'Marker','diamond')
xlim([0.5 3.5])
grid on
xlabel('X')
ylabel('Y','Rotation',0);
```

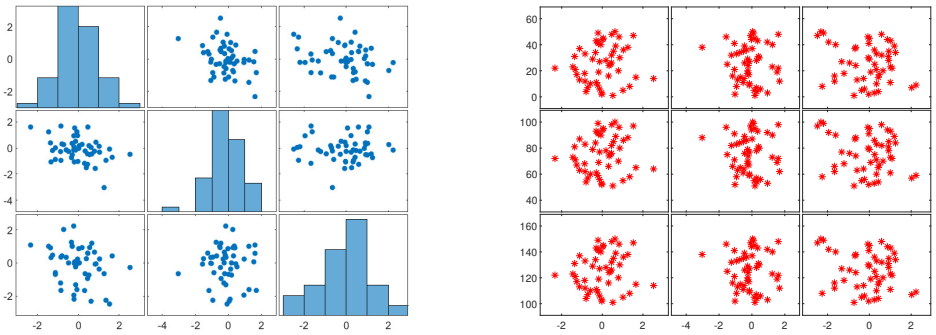


Obrázek 1.7: Errorbar

Posuneme se nyní o kousek dál, v situaci, kdy dostaneme datový vzorek s několika proměnnými, se budeme jistě ptát na rozložení jednotlivých proměnných a také na vzájemný vztah. V případě matice scatter plotů u 1 proměnné, která je tvořena 3 sloupci, vznikne matice 3x3, u které jsou na hlavní diagonále vytvořeny histogramy, podobně jako v levé části obrázku 1.8. Abychom prozkoumali vztah mezi jednotlivými proměnnými je příhodné použít matici

scatter plotů se dvěma proměnnými, viz pravá část obrázku 1.8. Toto lze využít pro grafickou vizualizaci závislosti mezi 2 proměnnými. Jinými slovy jsou vytvořeny bodové grafy Y v závislosti na X. Pro ukázku byly vytvořeny 3 sloupce proměnných X, Y. Tudíž subplot na i-tém řádku a j-tém sloupci je bodový graf i-tého sloupce proměnné Y v závislosti na j-tém sloupci proměnné X.

```
X = randn(50,3);
plotmatrix(X)
X = randn(50,3);
Y = reshape(1:150,50,3);
plotmatrix(X,Y,'r*')
```

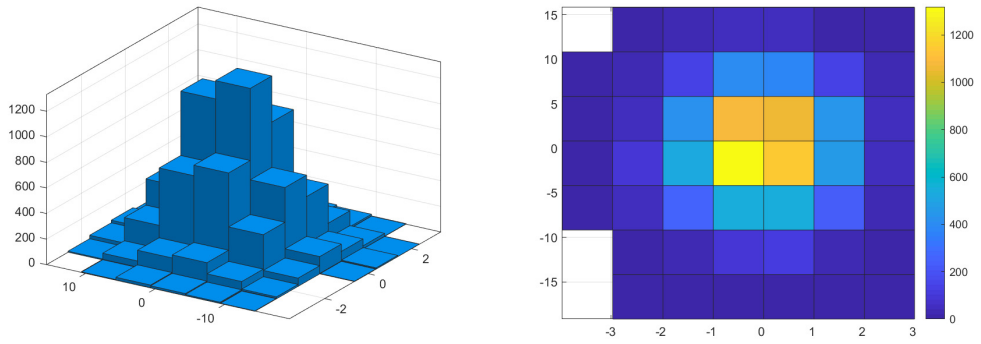


Obrázek 1.8: Matice scatter plotů

Pro vytvoření dvourozměrného histogramu využijeme funkci *histrogram2*, kde argumenty *xVals/yVals* jsou vektory dat, které budou vykresleny. Argumenty *xEdges/yEdges* jsou vektory, které specifikují hrany hranolů histogramu. Tato funkce je uvedena spíše pro zajímavost, jelikož 3D grafy bývají často nevhodné k prezentaci. Grafický výstup je v levé části na obrázku 1.9. Změnu stylu grafu získáme volbou *DisplayStyle*, pomocí **tile** obdržíme obdélníkové pole dlaždic představující zadané hodnoty, viz pravá část obrázku 1.9. Funkce *colorbar* přidá ke grafu barevnou škálu zobrazující stupnici. (Její limity lze nastavit příkazem *caxis*.)

```
hFig = figure;
hAxes = axes(hFig);
xVals = randn(1,10000);
yVals = randn(1,10000)*5;
```

```
xEdges = min(xVals):1:max(xVals);
yEdges = min(yVals):5:max(yVals);
hHist = histogram2(hAxes,xVals,yVals,xEdges,yEdges);
hHist_tile = histogram2(hAxes,xVals,yVals,xEdges,yEdges,
    'DisplayStyle','tile');
colorbar;
caxis([0,600])
```

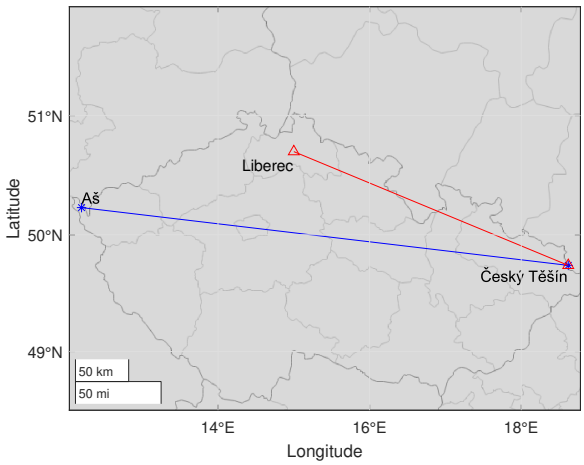


Obrázek 1.9: Bivariační histogramy

Zeměpisné grafické výstupy

V následujících odstavcích se budeme věnovat vytváření grafů s mapovými podklady v Matlabu. Začneme s možností vynesení čáry či bodů do mapy světa, či její části dle zadané zeměpisné šířky *lat* a délky *lon*. Argumenty zeměpisné délky a šířky musí být číselné vektory o stejné velikosti. Pro ukázku jsou zvoleny vrcholy měst - Aš (50 °14' s.š., 12 °12' v.d.) a Český Těšín (49 °45' s.š., 18 °38' v.d.). Grafickou podobu nalezneme na obrázku 1.10.

```
latAs = 50.23;
lonAs = 12.2;
latTesin = 49.76;
lonTesin = 18.65;
geoplot([latAs,latTesin],[lonAs,lonTesin],'b-*')
text(latTesin,lonTesin,'Český Těšín','HorizontalAlignment',
    'right','VerticalAlignment','top');
text(latAs,lonAs,'Aš','HorizontalAlignment','left',
    'VerticalAlignment','bottom');
```



Obrázek 1.10: Vynesení čar do zeměpisné mapy

Cvičení 1.12

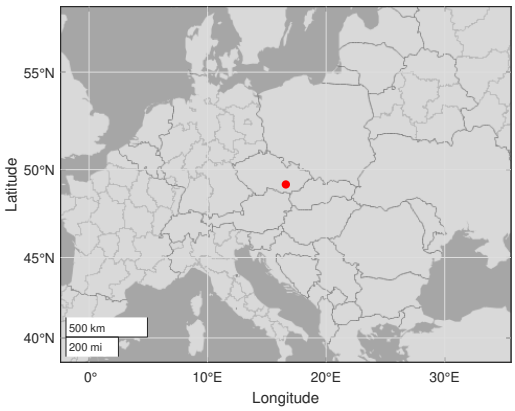
V řešeném příkladu bude cílem doplnit mapu o cestu z Českého Těšína do Liberce podle obrázku 1.10.

Řešení:

```
latLiberec = 50.7;
lonLiberec = 15;
geoplot([latAs,latTesin],[lonAs,lonTesin],'b-*',
    [latTesin,latLiberec],[lonTesin,lonLiberec],'r-^')
text(latTesin,lonTesin,'Český Těšín','HorizontalAlignment',
    'right','VerticalAlignment','top');
text(latAs,lonAs,'Aš','HorizontalAlignment','left',
    'VerticalAlignment','bottom');
text(latLiberec,lonLiberec,'Liberec','HorizontalAlignment',
    'right','VerticalAlignment','cap');
```

Bodový graf v zeměpisných souřadnicích získáme funkcí *geoscatter*. Opět funkce vykreslí mapu světa, či její část, na základě specifikace zeměpisných souřadnic. Lokaci města Brna nalezneme na obrázku 1.11.

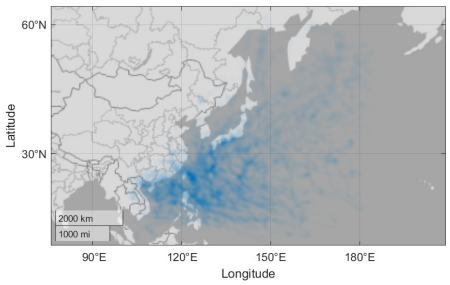
```
latBrno = 49.19;
lonBrno = 16.60;
geoscatter(latBrno,
    lonBrno,'r','filled')
```



Obrázek 1.11: Bodový graf v zeměpisných souřadnicích

Graf geografické hustoty získáme příkazem *geodensityplot*. Pro názornou ukázku jsou použita data *cycloneTracks*, která jsou součástí Matlabu. Jedná se o pozorování vyskytujících se cyklónů mezi lety 2007–2017. Zobrazení hustoty výskytu cyklónů nalezneme na obrázku 1.12.

```
load cycloneTracks
latAll =
  cycloneTracks.Latitude;
lonAll =
  cycloneTracks.Longitude;
geodensityplot(latAll,
  lonAll)
```

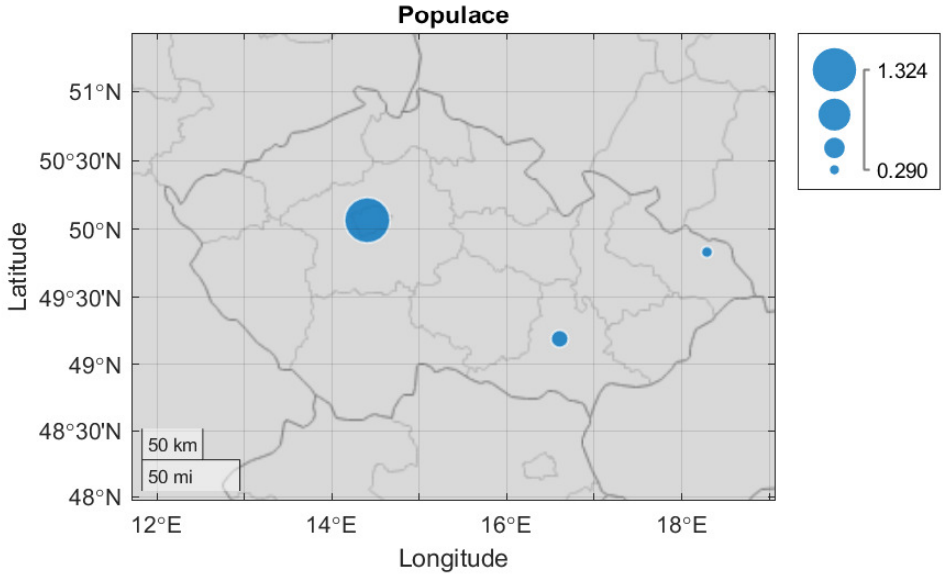


Obrázek 1.12: Graf geografické hustoty

Pokud je naším cílem vizualizovat na mapě vrcholy představující nějaké hodnoty např. počet obyvatel, můžeme využít možnosti funkce *geobubble*. Data mohou být zadána datovou tabulkou. Na obrázku 1.13 jsou na části mapy vyznačeny 3 největší města ČR tak, že velikost kolečka odpovídá počtu obyvatel. V pravé části je obrázek doplněn menší legendou.

```
latPrague = 50.07;
lonPrague = 14.41;
popPrague = 1.324;
latBrno = 49.19;
lonBrno = 16.60;
popBrno = 0.381;
```

```
latOstrava = 49.83;
lonOstrava = 18.29;
popOstrava = 0.29;
geobubble([latPrague,latBrno],[lonPrague,lonBrno],
    [popPrague,popBrno],'Title','Populace');
geobubble([latPrague,latBrno,latOstrava],[lonPrague,
    lonBrno,lonOstrava],[popPrague,popBrno,popOstrava],
    'Title','Populace')
```



Obrázek 1.13: Počet obyvatel ve 3 největších městech ČR

Pokročilé grafické výstupy

Krátce se rovněž zmíníme o pokročilejších funkcích, které jsou nad rámec tohoto kurzu.

- **polarplot(theta,rho,LineSpec)** - Funkce, která vykreslí čáru v polárních koordinátech, kde *theta* indikuje úhel v radiánech a *rho* indikuje poloměr celé plochy. Volitelný argument *LineSpec* slouží ke specifikaci stylu čar, markerů a barvy čáry.

- **polarscatter(theta,rho,sz,c,mkr)** - Funkce, která vykreslí klasický scatter plot v polárních koordinátech, kde *theta* indikuje úhel v radiánech a *rho* indikuje poloměr celé plochy. Volitelné argumenty *sz, c, mkr* nastavují velikost, barvu a tvar markerů.
- **polarhistogram(theta,nbins)** - Funkce, která vykreslí klasický histogram v polárních koordinátech, kde *theta* indikuje úhel v radiánech rovnoměrně rozdělen do stejně velkých výsečí. Volitelný argument *nbins* specifikuje počet výsečí.
- **stairs(X,Y)** - Funkce, která vykreslí schodovitý graf na základě argumentů *X, Y*, což jsou vykreslované vektory dat.
- **contour(X,Y)** - Funkce, která vykreslí konturovaný graf na základě vektorů dat *X, Y*.
- **contour3(X,Y)** - Funkce, která vykreslí 3D konturovaný graf na základě vektorů *X, Y*.
- **surface(X,Y,Z,C)** - Funkce, která vytvoří 2D povrchový graf dle vektorů *X, Y*. V případě 3D povrchového grafu je přidán vektor *Z*. Volitelný argument *C* slouží ke specifikaci barevného nastavení povrchu.
- **fsurf(FUN,INT)** - Funkce, která vykreslí 3D povrchový graf na základě funkce *FUN*. Volitelný argument *INT* specifikuje interval funkce.
- **surfnorm(X,Y,Z)** - Funkce, která vykreslí povrchový graf a zobrazí jeho normály. Argumenty *X,Y,Z* jsou vektory dat.
- **surfl(X,Y,Z,'light',s,k)** - Funkce, která vykreslí 3D stínovaný povrchový graf s osvětlením. Argumenty *X,Y,Z* jsou vektory dat. Volitelný argument *light* produkuje barevně osvětlený povrch s využitím objektu LIGHT. Volitelné argumenty specifikují směr zdroje světla a odrazivost povrchu.
- **surfc(...)** - Funkce, která vytvoří kombinaci povrchového a konturovaného grafu. Stejné argumenty jako pro funkci *surf*. Konturovaný graf je vykreslen pod povrchovým grafem.
- **fmesh(FUN,INT)** - Funkce, která vykreslí pletený *mesh* povrchový 3D graf na základě funkce *FUN*. Volitený argument *INT* specifikuje interval funkce.
- **meshz(...)** - Funkce, která vykreslí pletený *mesh* povrchový 3D graf s oponou. Argumenty stejné jako pro funkci *mesh*.

- **meshc(...)** - Funkce, která vytvoří kombinaci pleteného povrchového a konturovaného grafu. Stejné argumenty jako pro funkci *mesh*.
- **mesh(...)** - Konturovaný graf je vykreslen pod povrchovým grafem.
- **waterfall(...)** - Funkce, která vytvoří vodopádový graf. Graf je téměř stejný jako *mesh* graf a funkce má stejné argumenty jako funkce *mesh*.
- **hidden** - Funkce, která zobrazí nebo odstraní skryté čáry v *mesh* grafu. Nastavení *hidden* **ON** čáry odstraní, *hidden* **OFF** je zobrazí.
- **ribbon(X,Y,width)** - Funkce, která vytvoří 2D čáry jako 3D stužkový graf. Argument *X, Y* jsou vektory dat. Volitelný argument *width* specifikuji šířku stužky.

1.4 Vizualizace dat: pokročilé úpravy grafů

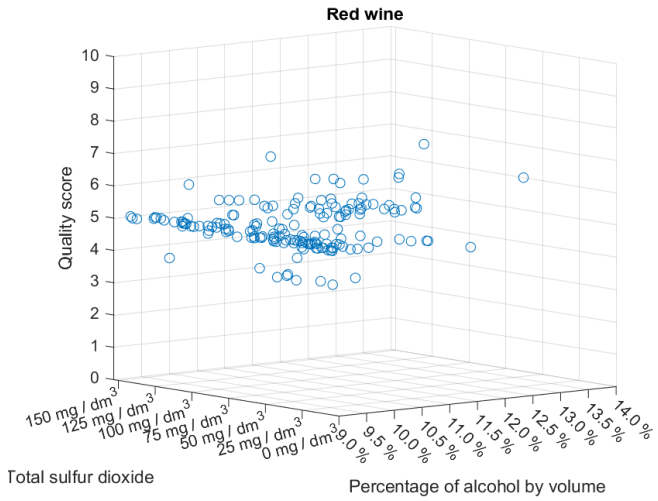
V této části se budeme zabývat pokročilými možnostmi upravování grafických výstupů. Zejména budeme pracovat s datovým souborem o červeném víně *winered.csv*, který disponuje 12 proměnnými, např. množství alkoholu, pH, kvalita vína, výskyt oxidu siřičitého, kyseliny citronové, hustota atd. Do proměnných *x, y, z* si uložíme sloupce, se kterými budeme dále pracovat. Pro upravení měřítka na ose x je vhodné použít příkaz *xticks*, pojmenování os se provádí pomocí _*label*, kde podtržítko zastupuje např. x pro *xlabel* apod.

```
T = readtable('winered.csv')
head(T)
x = T.alcohol;
y = T.totalSulfurDioxide;
z = T.quality;
figure1 = figure;
scatter3(x,y,z)
xticks(0:0.5:15)
yticks(0:25:150)
zticks(0:1:10)
title('Red wine')
xlabel('Percentage of alcohol by volume')
ylabel('Total sulfur dioxide')
zlabel('Quality score')
```

Nastavení popisků os vytvoříme prostřednictvím funkce _*ticklabels*. Abychom upravili formát popisků os, využijeme funkci _*tickformat*, pro dvouciferné

číslo s 1 desetinným místem zapíšeme *2.1f*, pro text za každou hodnotou využijeme *g*. Limity osy *z* upravíme pomocí *zlim*. Pro lepší viditelnost popisky nakloníme díky _*tickangle*. Nastavením všeho zmíněného dostáváme graf na obrázku 1.14.

```
xticklabels({'9 %','9.5 %','10 %','10.5 %','11 %',
    '11.5 %','12 %','12.5 %','13 %','13.5 %','14 %'})
yticklabels({'0 mg / dm^3','25 mg / dm^3','50 mg / dm^3',
    '75 mg / dm^3','100 mg / dm^3','125 mg / dm^3',
    '150 mg / dm^3'})
xtickformat('%2.1f \%')
ytickformat('%g mg / dm^3')
xtickangle(-45)
ytickangle(15)
zlim([0,10])
xtickangle(-45)
ytickangle(15)
```



Obrázek 1.14: Ukázka základních grafických úprav

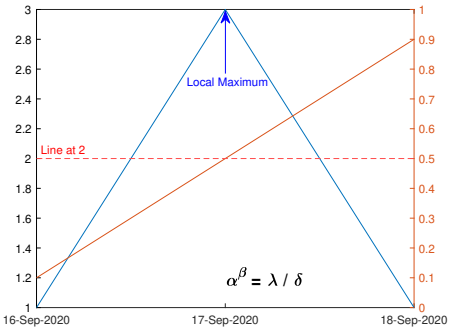
Prozkoumáme funkci *datetick*, která nám pomůže při formátování hodnot na osách grafu, a to na čas podle zvoleného formátu. Využití se nabízí pro report měsíčního/ročního vývoje proměnných. Pro ukázku vytvoříme vektor s časovými údaji, který pro potřeby grafu na obrázku 1.15 převedeme na čísla. Pro znázornění vodorovné či vertikální čáry s konstantní hodnotou lze využít funkci *xline* či *yline*. Můžeme změnit základní nastavení např. styl a barvu

čáry, volbou *LabelHorizontalAlignment* zarovnáme její název vlevo. Funkce *annotation* přidává symboly a případně i text do grafu. Povinnými argumenty jsou vektory obsahující souřadnice bodů, které definují pozici symbolu. Volitelný argument přidává text.

```
y = [1,3,1];
z = [0.1,0.5,0.9];
time = [datetime('yesterday'):1:datetime('tomorrow')];
x = datenum(time);
figure2 = figure;
plot(x,y)
datetick('x',1);
yline(2,'r--','Line at 2', 'LabelHorizontalAlignment',
    'left');
annotation('textarrow',[0.5175,0.5175],[0.75,0.92],
    'color','red','String','Local Maximum');
```

Pro vytvoření grafu se dvěma osami využijeme příkaz *yyaxis right*, který aktivuje vertikální osu vpravo. U pravé osy jsou přeškálovány limity, poté jsme přepnuti zpět na levou vertikální osu pomocí příkazu *yyaxis left*. Zejména pro sazbu matematických znaků lze využít funkci *texlabel*, která převede text do formátu LaTeX. Pomocí funkce *text* je zvětšený a ztučněný popisek přidán do grafu.

```
yyaxis right
hold on
plot(x,z)
ylim([0,1])
yyaxis left
tx = texlabel('alpha^beta
    = lambda / delta');
text(x(2),1.2,tx,
    'FontSize',15,
    'FontWeight','bold')
```

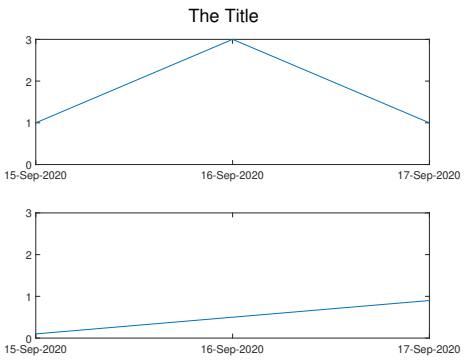


Obrázek 1.15: Přidání různých objektů do grafu

Více grafů v jednom obrázku získáme pomocí příkazu *subplot*. Název grafu zde nastavíme prostřednictvím příkazu *sgtitle*. V případě, že chceme výsledky

z grafů navzájem porovnat, může se hodit funkce *linkaxes*, která sjednotí limity vertikální i horizontální osy. Podívejme se na grafický výstup na obrázku 1.16.

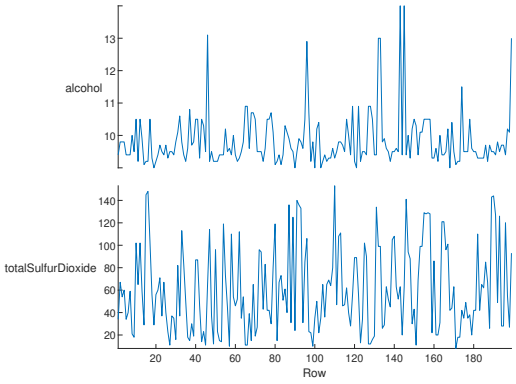
```
figure3 = figure;
ax1 = subplot(2,1,1);
plot(x,y)
datetick('x',1);
ax2 = subplot(2,1,2);
plot(x,z)
datetick('x',1);
sgtitle('The Title');
linkaxes([ax1,ax2],'y');
```



Obrázek 1.16: Více grafů v jednom obrázku

Nyní se podíváme na možnost grafického zobrazení funkcí *stackedplot*. Tato funkce již byla použita v části 1.1 na obrázku 1.1. Jedná se o interaktivní zobrazení několika proměnných nad sebou se společnou osou x. Díky této funkci můžeme sledovat vzájemný vývoj hodnot až pro 25 proměnných. Na obrázku 1.17 leží na vodorovné ose x jednotlivé řádky pozorovaných vzorků vína, volbou *XVariable* lze zvolit jinou proměnnou.

```
figure4 = figure;
stackedplot(T,{'alcohol',
    'totalSulfurDioxide'})
```



Obrázek 1.17: Interaktivní zobrazení proměnných nad sebou

Také si ukažme možnosti ukládání grafických výstupů ve verzích *.fig*, *.png* a *.eps*.

```
>> savefig('Stacked_plot_fig.fig')
>> saveas(figure4,'Stacked_plot','png')
>> saveas(figure4,'Stacked_plot','eps')
```

Formát uloženého obrázku *.fig* umožňuje nadále upravovat v rámci grafických nástrojů Matlabu vlastnosti, které jsme si v této kapitole představili. Pro otevření uloženého grafu slouží funkce *openfig*.

```
>> openfig('Stacked_plot_fig.fig')
```

Pokročilé úpravy grafických výstupů

I zde se krátce zmíníme o dalších pokročilejších funkcích. Nejdříve o těch, pro které je nutností balíček Mapping Toolbox, který otevírá nové možnosti pro práci s mapami.

- **geoshow(lat,lon)** - Funkce, která vykreslí mapu světa, případně její část, dle specifikace argumentů stupňů zeměpisné délky *lat* a šířky *lon*. Dokáže vykreslit mapu v různých tvarech a stylech, např. geografická mapa země nebo dokonce i fotografický snímek měsíce.
- **textm(lat,lon,'String',..)** - Funkce, která je podobná funkci *annotation*, přidává symboly a případně i text do mapy. Argumenty *lat* a *lon* odpovídají stupňům zeměpisné délky a šířky, které určují přidání textu. Argument *String* přidává text do mapy.
- **axesm(Name,Value)** - Funkce, která vytvoří osy na mapě. Volitelné argumenty *Name* a *Value* specifikují název os a vlastnosti jako je velikost písma.
- **shaperead(Filename)** - Funkce, která vrací Nx1 strukturované pole s vlastnostmi a atributy uložené v *Filename*.

Dále uvádíme funkce, které fungují od verze Matlabu 2019b.

- **tiledlayout(m,n,'flow')** - Funkce, která společně s funkcí *nexttile* odpovídá funkci *subplot*. Vytváří prázdný prostor mxn prostor, do kterého se budou vykreslovat grafy.
- **nexttile** - Funkce, která umožní přidat do prázdného prostoru graf a přesunout se v prázdném prostoru do další pozice pro přidání dalšího grafu.

Doplňujeme také funkci, která funguje od verze Matlabu 2020a.

- **exportgraphics(obj,filename,Name,Value)** - Funkce, která ukládá graf nebo obrázek *obj* do souboru *filename* pod určitým názvem *Name*. Volitelný argument *Value* dovoluje nastavit rozlišení.

1.5 Neřešené příklady

1. Vytvořte datovou tabulku s alespoň 2 různými datovými typy o nejbližších hvězdách k Zemi. Výsledek může vypadat jako na obrázku 1.18.

Stars	DistanceInLightYears	StarClass	DiscoveryDate
'Sun (Sol)'	1.58e-05	'yellow dwarf'	NaN
'Proxima Centauri (V645 Centauri)'	4.2441	'red dwarf'	1915
'Alpha Centauri A (Rigil Kentaurus)'	4.365	'yellow dwarf'	150
'Alpha Centauri B (Toliman)'	4.365	'orange dwarf'	1689
'Barnard's Star'	5.9577	'red dwarf'	1916

Tabulka 1.18: Ukázka datové tabulky

2. Zpracujte datový soubor *Airbnb.csv*. Nahraďte prázdné hodnoty medián em, pokuste se opravit podivné hodnoty, zbavit se duplikátů, různých a zároveň stejných hodnot. Nakonec seřaďte datový soubor podle spokojenosti a podívejte se na prvních 10 pozorování. Také prozkoumejte shrnutí datové tabulky, chybějící hodnoty generujte na základě rozložení dat nebo vhodnou metodou. S využitím *nearest* výsledek vypadá jako v tabulce 1.19.

room_id	room_type	neighborhood	reviews	overall_satisfaction	price
1.501e+07	'Private room'	'Brno'	9	5	26
1.2468e+07	'Entire home/apt'	'Brno'	17	5	42
1.5765e+07	'Entire home/apt'	'Brno'	0	5	50
6.3589e+06	'Entire home/apt'	'Brno'	10	5	69
1.5647e+07	'Entire home/apt'	'Brno'	0	5	85
1.2187e+07	'Entire home/apt'	'Brno'	16	5	22
1.098e+07	'Private room'	'Brno'	1	5	29
1.089e+07	'Private room'	'Brno'	7	5	32
1.1782e+07	'Private room'	'Brno'	1	5	22
2.1906e+06	'Private room'	'Brno'	0	5	56

Tabulka 1.19: Seřazení podle spokojenosti zákazníků

3. Z datového souboru *carbig*, který je součástí Matlabu, vytvořte datovou tabulku obsahující všechny proměnné týkajících se automobilů. Ověřte, zda se v souboru nevyskytují chybějící hodnoty, příp. vhodným

způsobem je nahraďte, nejlépe mediánem. A nalezněte následujícím 3 klientům alespoň 3 nejvhodnější automobily, pokud je to možné.

- Sportovní automobil - nejvyšší možné zrychlení, vysoká koňská síla, vysoký zdvihový objem a mající osmiválec. Řešení se nachází v tabulce 1.20.

Acceleration	Cylinders	Displacement	Horsepower	Model
22.2	8	260	90	'oldsmobile cutlass salon brougham '
19	8	260	110	'oldsmobile cutlass supreme '
19	8	350	105	'oldsmobile cutlass ls '

Tabulka 1.20: Vhodné sportovní automobily

- Rodinný automobil - maximálně 3 roky starý, aspoň střední koňská síla a maximálně střední váha. Řešení naleznete v tabulce 1.21.

Cylinders	Horsepower	Model_Year	Origin	Weight	Model
6	110	82	USA	2945	'buick century limited '
6	112	82	USA	2835	'ford granada l '

Tabulka 1.21: Vhodné rodinné automobily

- Německý automobil s vysokou koňskou sílou. Výsledek se nachází v tabulce 1.22.

Acceleration	Cylinders	Horsepower	Origin	Model
16.7	6	120	Germany	'mercedes-benz 280s '
12.5	4	113	Germany	'bmw 2002 '
12.8	4	110	Germany	'bmw 320i '

Tabulka 1.22: Pro milovníky německých aut

4. V tomto příkladu se budete zabývat daty o vývoji počtu obyvatel v Praze a jejím okolí z datového souboru *praha_demografie.xlsx*. Řešte následující úkoly:

   - Doplňte případné chybějící hodnoty mediánem dané obce.
   - Vypočtěte natalitu a mortalitu a uložte je jako nové proměnné.
   - Vypočtěte podíl imigrace počtu obyvatel, jinými slovy kolik obyvatel (stav na konci roku) se přistěhovalo a podíl emigrace, čili kolik obyvatel (stav na začátku roku) se odstěhovalo.

- Roztřiďte obce na malé obce (do 1000 obyvatel), střední obce (do 10000 obyvatel), velké obce (10000+, bez Prahy) a Prahu. Na základě jejich průměrů porovnejte jejich natalitu, mortalitu, imigraci a emigraci za posledních 20 let. Výslednou datovou tabulku 1.23 se souhrnnými statistikami podle skupin získáte pomocí funkce *grpstats*, jejíž povinnými argumenty jsou zdrojová tabulka a proměnná vedoucí k seskupení.

VelikostObce	GroupCount	Natalita	Mortalita	Imigrace	Emigrace
Mala Obce	3640	0.012179	0.0098545	0.060478	0.029994
Praha	20	0.010649	0.010199	0.028234	0.022032
Stredni Obce	80	0.012137	0.009469	0.04838	0.027118
Velka Obce	60	0.011264	0.0095862	0.039333	0.027286

Tabulka 1.23: Souhrnné statistiky průměru dle skupin

- Najděte obci, která je malé nebo střední velikosti, v níž došlo k největšímu růstu a poklesu počtu obyvatel. Zjistěte, v kterém roce se tak stalo. Řešení jsou k dispozici v tabulkách 1.24 a 1.25.

Rok	NazevObce	PrirustekCelkovy	VelikostObce
2008	Jesenice	606	Mala Obce

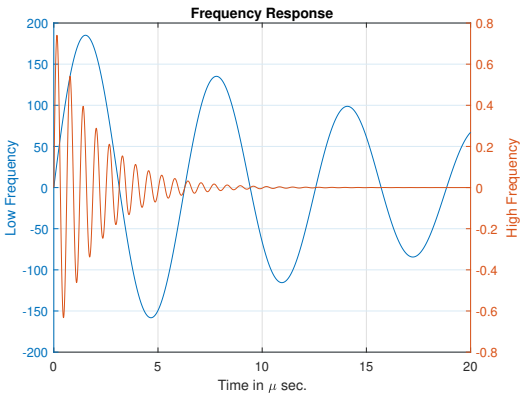
Tabulka 1.24: Největší růst počtu obyvatel

Rok	NazevObce	PrirustekCelkovy	VelikostObce
1975	Mníšek pod Brdy	-113	Mala Obce

Tabulka 1.25: Největší pokles počtu obyvatel

5. Vytvořte graf se 2 křivkami a uložte ho jako *.fig* soubor, přičemž využijte možnosti 2 vertikálních os. Data, které se budete snažit vykreslit do podoby obrázku 1.18, si prvně vytvoříte pomocí následujících příkazů. Vaše řešení by mělo vypadat jako na obrázku 1.18.

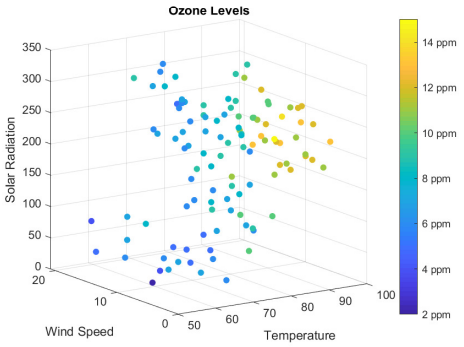
```
x  = 0:0.01:20;
y1 = 200*exp(-0.05*x).*sin(x);
y2 = 0.8*exp(-0.5*x).*sin(10*x);
```



Obrázek 1.18: Graf se dvěma osami y

6. V tomto neřešeném příkladu si zopakujte konstrukci 3D scatter plotu na základě vzorového datového souboru z Matlabu *ozoneData*. Grafický výstup bude vypadat jako na obrázku 1.19. Při konstrukci 3D grafu přidejte barevnou škálu podle indexu pro úroveň ozónu, kterou lze vypočítat takto:

```
load ozoneData
z = (Ozone).^(1/3);
response = z;
nc = 16;
offset = 1;
c = response - min(response);
c = round((nc - 1 - 2*offset)*c/max(c) + 1 + offset);
```



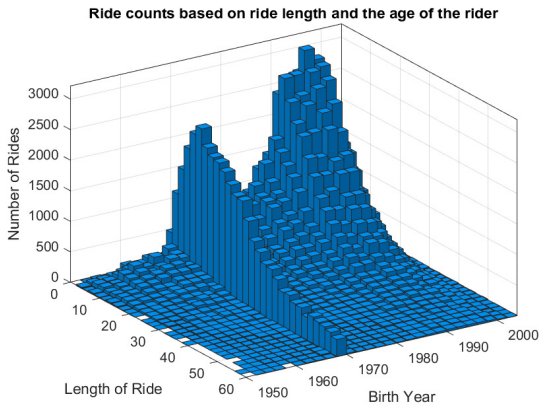
Obrázek 1.19: 3D scatter plot

7. Sestavte bivariační 3D histogram pro všechny uživatele mladších 70 let a jezdili na kole nejdéle hodinu. Soubor *bluebikes.csv* obsahuje potřebná data o Bostonském programu sdílených kol. Nejdříve se seznamte s proměnnými tohoto datového souboru, zejména s proměnnými *tripduration, birthYear*, které jsou součástí tabulky 1.26. Hodnoty proměnné *tripduration* jsou uložené v sekundách, převeďte je na minuty. Výsledek může vypadat jako na obrázku 1.20.

tripduration:	274022 × 1 double
Values:	
Min	61
Median	805
Max	2.8328e+06

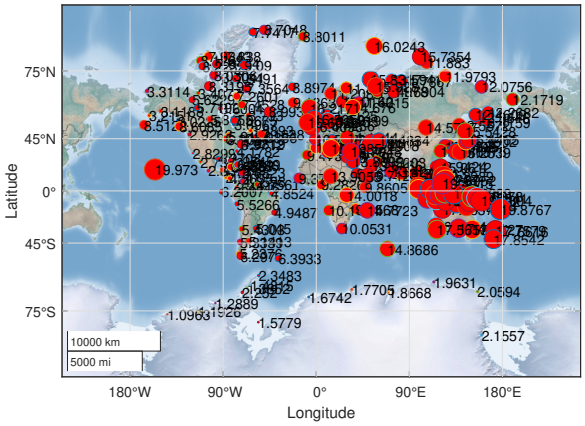
birthYear:	274022 × 1 double
Description: Original column heading: 'birth year'	
Min	1888
Median	1988
Max	2003

Tabulka 1.26: Základní popisné statistiky vybraných proměnných



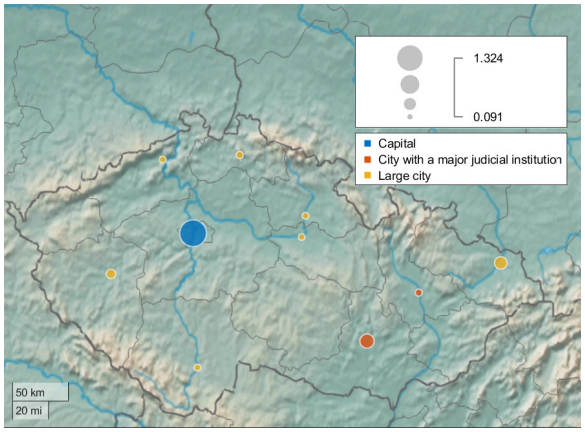
Obrázek 1.20: Bivariační 3D histogram

8. Pro tento neřešený příklad načtěte datový soubor *coastlines*, jenž je opět součástí Matlabu, obsahující údaje o zeměpisných koordinátech všech břehů světa. Pomocí cyklu vygenerujte mapu, do které vykreslete různé velikosti kruhů a přidejte jejich velikost, která bude škálovaná s jejich indexem. Na přesné velikostí kruhů nezáleží. Základní podobu mapy dle obrázku 1.21 změníte funkcí *geobasemap*.



Obrázek 1.21: Zeměpisné koordináty všech břehů světa

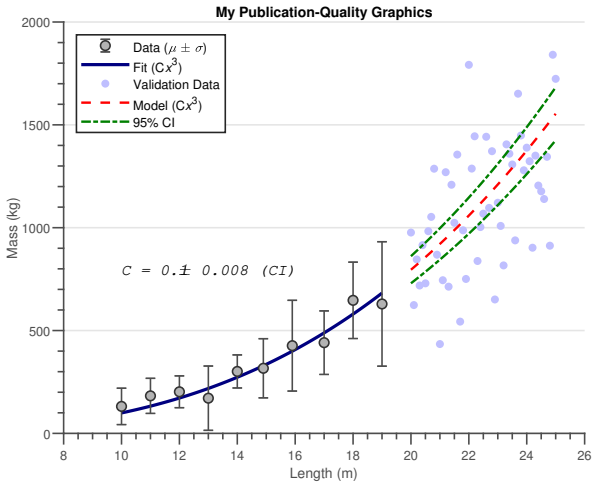
9. S využitím datasetu *cities.mat* vytvořte mapu, ve které vyznačte 10 největších měst ČR, dle počtu obyvatel roztřiďte města podle jejich významnosti. Mapu nastavte tak, aby se zobrazila stejným způsobem jako na obrázku 1.22 včetně přiblížení.



Obrázek 1.22: Největší města ČR podle počtu obyvatel

10. Pro konstrukci grafu na obrázku 1.23 načtěte datový soubor *data.mat*. Zkonstruujte křivku z proměnných *xfit,yfit* a chybový graf s využitím proměnných *xdata_m, ydata_m, ydata_s*. Přidejte další část výsledného zobrazení pomocí proměnných *xVdata, yVdata*. K těmto datům doplňte proložení modelem, využijte proměnné *xmodel, ymodel* z datového souboru. Analogicky přidejte konfidenční interval *(ymodelL,ymodelU)*. Nastavte parametry čar, znaků a písma podle obrázku 1.23.

Přidejte popisky obou os a název grafu, formátujte přidaný text a popisky os pomocí *FontName* na *AvantGarde*. Rovněž přidejte legendu na základě proměnných *hE, hFit, hData, hModel, hCI(1)*. Parametrem *XMinorTick* **on** lze nastavit jemnější stupnici. Pokud nechcete zobrazit celou mřížku, ale pouze zobrazit osy y, nastavte *YGrid* **on**.



Obrázek 1.23: Shrnutí použití grafických nástrojů