Projekt do MSP 2022/2023

- Jan Zbořil
- xzbori20
- xzbori20@stud.fit.vutbr.cz

Pro práci si nejprve importujeme potřebné moduly a data. Data jsem si předem transformoval z excelu do `.csv`, který jsem importoval do pythonu pomocí knihovny Panda. Využívám těchto knihoven:

- `matplotlib` - pro kreslení grafů
- `numpy` - pro numerické operace
- `pandas` - pro práci s datovou sadou a tabulkami
- `scipy stats` - pro provádění statistických testů
- `statsmodels` - pro lineární regresi

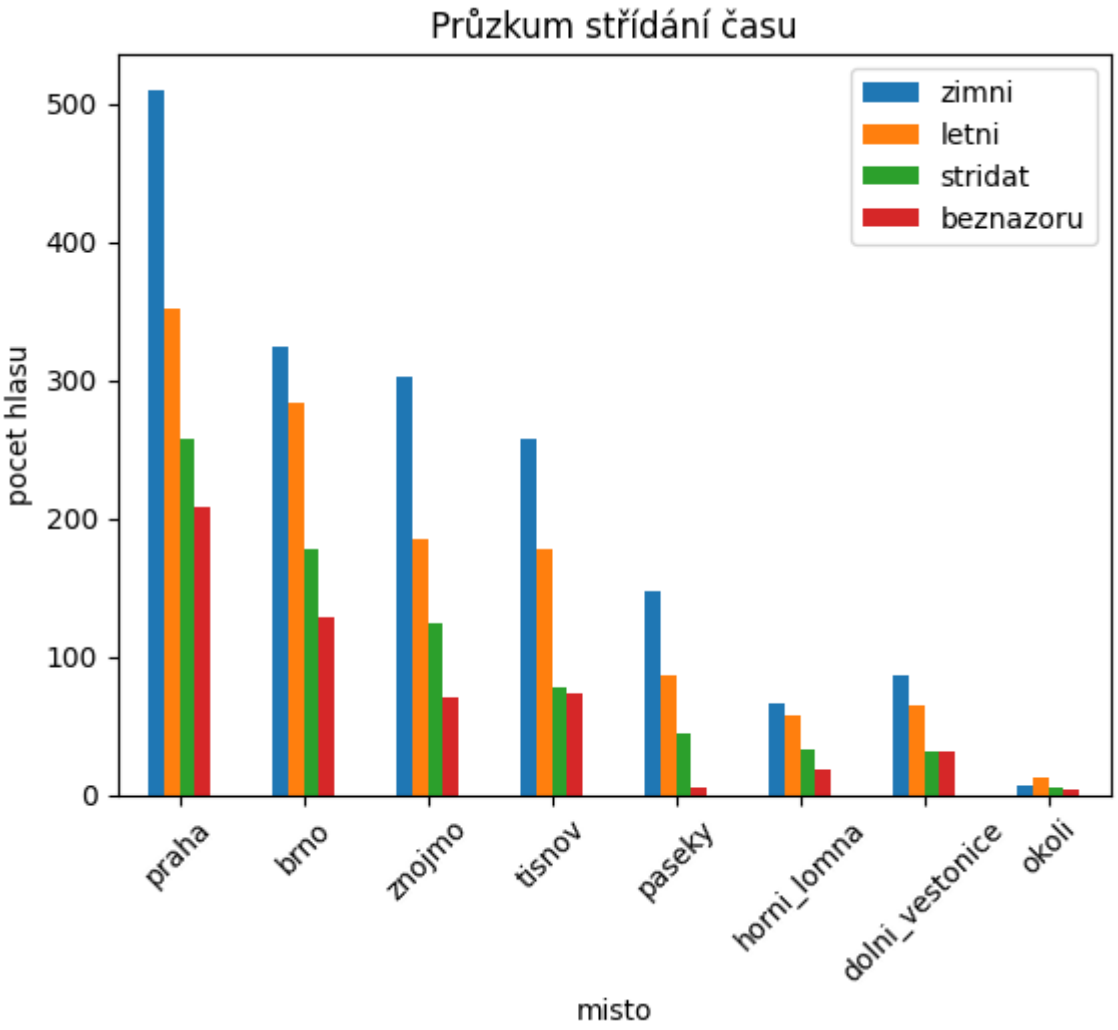
In [1]:
```
# import knihoven
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import stats as st
import statsmodels.api as sm

# inicializace datove say
df1 = pd.read_csv('dataset_1.csv')
df2 = pd.read_csv('dataset_2.csv')

# ulozeni nazvu mest pro dalsi pouziti
MESTA = df1['misto']
MESTA.add(str(0))

# vygeneruje tabulku pro dataset 1
# print(df1.to_markdown())
# print(df1)

# pro zadny vystup
print('')
```

	pocet	zimni	letni	stridat	beznazoru	misto
0	1327	510	352	257	208	praha
1	915	324	284	178	129	brno
2	681	302	185	124	70	znojmo
3	587	257	178	78	74	tisnov
4	284	147	87	44	6	paseky
5	176	66	58	33	19	horni_lomna
6	215	87	65	31	32	dolni_vestonice
7	23	7	12	6	4	okoli

Úkol 1

Z nahraných dat jsem si vygeneroval graf ukazující počet odpovědí v jednotlivých městech.

In [2]:
```
df11 = df1.drop(labels='pocet', axis=1)
ax = df11.plot.bar(xlabel='misto', ylabel='pocet hlasu', \
    title='Průzkum střídání času')
ax.set_xticklabels(df1['misto'], rotation=45)

plt.show()
```



Pro posouzení stejného procentuálního zastoupení - tedy pravděpodobnosti, že obyvatelé podporují střídání časů, letní čas nebo zimní čas - použijeme testy hypotézy homogenity v kontingenční tabulce. Takový test lze také nazvat testem dobré shody nebo χ^2 test.

Pro provedení takového testu potřebujeme nejprve znát sumy řádků a sloupců. Následující kód tyto sumy počítá a navíc provádí různé další pomocné transformace

In [3]:
```
df11 = df1.drop(labels='misto', axis=1)
df11.drop(labels='pocet', axis=1, inplace=True)

sum_rows = df11.sum(axis=0)
sum_cols = df11.sum(axis=1)

try:
    df1.insert(len(df1.columns), 'sum_misto', sum_cols, allow_duplicates=False)
    df_temp = pd.DataFrame(sum_rows)
except:
    pass

df1.rename(columns={'pocet': 'pocet_'}, inplace=True)
df1.drop(labels='pocet_', axis=1, inplace=True)
df1.rename(index={i: MESTA[i] for i in range(0,len(MESTA))}, inplace=True)
df1 = df1.append(df_temp.T, ignore_index=False)
df1.drop(labels='misto', axis=1, inplace=True)
df1['sum_misto'] = df1['sum_misto'].replace(np.nan, sum_cols.sum())
df1.rename({0: 'sum_cas'}, inplace=True, axis=0)

df_f = df1.copy()

# print(df1.to_markdown())
```

```
/tmp/ipykernel_34476/3414897264.py:16: FutureWarning: The frame.append method is d
eprecated and will be removed from pandas in a future version. Use pandas.concat i
nstead.
  df1 = df1.append(df_temp.T, ignore_index=False)
```

	zimni	letni	stridat	beznazoru	sum_misto
praha	510	352	257	208	1327
brno	324	284	178	129	915
znojmo	302	185	124	70	681
tisnov	257	178	78	74	587
paseky	147	87	44	6	284
horni_lomna	66	58	33	19	176
dolni_vestonice	87	65	31	32	215
okoli	7	12	6	4	29
sum_cas	1700	1221	751	542	4214

Pro testování jednotlivých hypotéz ze zadání datovou sadu transformuju na tabulku se dvěma sloupci - počet respodnentů, co preferuje daný čas; a počet respondetů, jež uvedli jinou odpověď. Tabulka také obsahuje sloupec a řádek sum. Následují řešení jednotlivých poúkolů.

---

a) V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co preferují zimní čas.

$\alpha = 0.05$,

H_0: V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co preferují zimní čas.

H_A: V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co NEpreferují zimní čas.

In [4]:
```
df_z = df1.filter(items=['zimni'])
df_z.insert(len(df_z.columns), 'jiny', df1.iloc[:,[1,2,3]].sum(axis=1),\
    allow_duplicates=False)
df_z.insert(len(df_z.columns), 'sum_misto', df1.iloc[:,[4]].sum(axis=1),\
    allow_duplicates=False)

# print(df_z.to_markdown())
```

	zimni	jiny	sum_misto
praha	510	817	1327
brno	324	591	915
znojmo	302	379	681
tisnov	257	330	587
paseky	147	137	284
horni_lomna	66	110	176
dolni_vestonice	87	128	215
okoli	7	22	29
0	1700	2514	4214

Z tabulky odpovědí vypočítám tabulku procent pomocí vzorce

$$\frac{x_{i,j}}{\sum_i x_i} * 100$$

, kde $\sum_i x_i$ je v tabulce vyjádřen sloupcem `sum_misto`. Správnost dokazuje fakt, že sloupec pro sumu obsahuje v tabulce procent pouze hodnoty 100.

In [5]:
```
def dfzPerc(df_z, to_print=False):
    N = df_z.at['sum_cas','sum_misto']

    df_z_perc = pd.DataFrame(data=np.zeros((9, len(df_z.columns))),\
        columns=df_z.columns)
    df_z_perc.rename(index={i: MESTA[i] for i in range(0,len(MESTA))},\
        inplace=True)
    df_z_perc.rename({8: 'sum_cas'}, inplace=True, axis=0)

    suma = 0
    try:
        for i in range(0,9):
            for j in range(0,2):
                df_z_perc.iat[i,j] = ( df_z.iat[i,j] /  df_z.iat[i,2] ) * 100
                suma = suma + df_z_perc.iat[i,j]

            df_z_perc.iat[i,2] = suma
            suma = 0
    except:
        pass

    if to_print:
        print("--------------------")
        print("dfz procenta:")
        print(df_z_perc.to_markdown())
        print("--------------------")
    return df_z_perc

df_z_perc = dfzPerc(df_z)
```

Tabulka procent pro zimní/jiný čas:

	zimni	jiny	sum_misto
praha	38.4326	61.5674	100
brno	35.4098	64.5902	100
znojmo	44.3465	55.6535	100
tisnov	43.7819	56.2181	100
paseky	51.7606	48.2394	100
horni_lomna	37.5	62.5	100
dolni_vestonice	40.4651	59.5349	100
okoli	24.1379	75.8621	100
sum_cas	40.3417	59.6583	100

S pomocí tabulky procent jsem spočítal tabulku očekávaných četností. Požil jsem následující vzorec

$$m_{i,j} = \frac{n_i n_j}{n}$$

In [6]:
```
def ocekavaneDfz(df_z, df_z_perc, to_print=False):
    dfm_z = pd.DataFrame(data=np.zeros((8, len(df_z.columns))),\
        columns=df_z.columns)
    dfm_z.rename(index={i: MESTA[i] for i in range(0,len(MESTA))},\
        inplace=True)
    dfm_z.drop(labels='sum_misto', axis=1, inplace=True)

    for i in range(0,len(df_z.index)-1):
        for j in range(0,len(df_z.columns)-1):
            dfm_z.iat[i,j] = (df_z_perc.iat[len(df_z.index)-1,j] / 100)\
                * df_z.iat[i,len(df_z.columns)-1]

    if to_print:
        print("--------------------")
        print("ocekavane (m) dfm_z")
        print(dfm_z.to_markdown())
        print("--------------------")

    return dfm_z

dfm_z = ocekavaneDfz(df_z, df_z_perc)
```

Tabulka očekávaných četností $m_{i,j}$:

	zimni	jiny
praha	535.335	791.665
brno	369.127	545.873
znojmo	274.727	406.273
tisnov	236.806	350.194
paseky	114.57	169.43
horni_lomna	71.0014	104.999
dolni_vestonice	86.7347	128.265
okoli	11.6991	17.3009

Pro otestování hypotézy H_0 jsem vyžil testu dobré shody, pro nějž je potřeba spočítat hodnotu χ^2 následujícím vzorcem.

$$\chi^2 = \sum_{i=1}^{R} \sum_{j=1}^{S} \frac{(n_{i,j} - m_{i,j})^2}{m_{i,j}}$$

Doplněk kritického oboru je < 0 ; $\chi^2_{0.95}(v) >$, kde $v = (r-1) * (s-1)$ představuje počet stupňů volnosti, kde s je počet sloupců tabulky a r je počet řádků v tabulce. Pokud hodnota χ^2 náleží kritickému oboru, pak hypotézu H_0 nezamítám, jinak hypotézu H_0 zamítám na hladině významnosti α a platí alternativní hypotéza.

In [7]:
```
def chiCalc(df_z, dfm_z):
    chi_z = 0
    alpha = 0.05
    for i in range(0, len(dfm_z.index)):
        for j in range(0, len(dfm_z.columns)):
            chi_z = chi_z + (((df_z.iat[i,j] - dfm_z.iat[i,j]) ** 2)\
                / dfm_z.iat[i,j])

    print("------------------")
    print("CHI^2 = {}, stupne volnosti = {}".format(chi_z,\
        (len(dfm_z.index)-1)*(len(dfm_z.columns)-1)))

    critical_value = st.chi2.ppf(1-alpha, \
        (len(dfm_z.index)-1)*(len(dfm_z.columns)-1))

    print("Doplnek kritickeho oboru = < 0 ; {} >"\
        .format(critical_value))

    if chi_z > critical_value:
        print("H0 zamitam")
    else:
        print("H0 nezamitam")
    print("------------------")

chiCalc(df_z, dfm_z)
```

```
------------------
CHI^2 = 37.82410668971246, stupne volnosti = 7
Doplnek kritickeho oboru = < 0 ; 14.067140449340169 >
H0 zamitam
------------------
```

Obdobně jako u příkladu a) postupujeme také v příkladech b) a c)

---

b) V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co preferují letní čas.

$\alpha = 0.05$,

H_0: V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co preferují zimní čas.

H_A: V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co NEpreferují letní čas.

In [8]:
```
df_z = df1.filter(items=['letni'])
df_z.insert(len(df_z.columns), 'jiny', df1.iloc[:,[0,2,3]].sum(axis=1),\
    allow_duplicates=False)
df_z.insert(len(df_z.columns), 'sum_misto', df1.iloc[:,[4]].sum(axis=1),\
    allow_duplicates=False)

# print(df_z.to_markdown())
```

	letni	jiny	sum_misto
praha	352	975	1327
brno	284	631	915
znojmo	185	496	681
tisnov	178	409	587
paseky	87	197	284
horni_lomna	58	118	176
dolni_vestonice	65	150	215
okoli	12	17	29
sum_cas	1221	2993	4214

In [9]:
```
df_z_perc = dfzPerc(df_z)
dfm_z = ocekavaneDfz(df_z, df_z_perc)
chiCalc(df_z, dfm_z)
```

```
------------------
CHI^2 = 11.429528731041264, stupne volnosti = 7
Doplnek kritickeho oboru = < 0 ; 14.067140449340169 >
H0 nezamitam
------------------
```

---

c) V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co preferují střídání času.

$\alpha = 0.05$,
H_0: V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co preferují střídání času.
H_A: V městech, obcích a v okolí studenta (8. průzkumů) je stejné procentuální zastoupení obyvatel, co NEpreferují střídání času.

In [10]:
```
df_z = df1.filter(items=['stridat'])
df_z.insert(len(df_z.columns), 'jiny', df1.iloc[:,[0,1,3]].sum(axis=1),\
    allow_duplicates=False)
df_z.insert(len(df_z.columns), 'sum_misto', df1.iloc[:,[4]].sum(axis=1),\
    allow_duplicates=False)

# print(df_z.to_markdown())
```

	stridat	jiny	sum_misto
praha	257	1070	1327
brno	178	737	915
znojmo	124	557	681
tisnov	78	509	587
paseky	44	240	284
horni_lomna	33	143	176
dolni_vestonice	31	184	215
okoli	6	23	29
sum_cas	751	3463	4214

In [11]:
```
df_z_perc = dfzPerc(df_z)
dfm_z = ocekavaneDfz(df_z, df_z_perc)
chiCalc(df_z, dfm_z)
```

```
------------------
CHI^2 = 15.153830811721116, stupne volnosti = 7
Doplnek kritickeho oboru = < 0 ; 14.067140449340169 >
H0 zamitam
------------------
```

Pro příklad d) a e) je nutné vytvořit nové kategorie mšst - větší, menší a vesnice, a pracovat s nimi. Hodnoty nových kategorií vzniknou sečtením hodnot původních měst patřících do stejné kategorie.

d) U větších měst, menších měst a obcí (3. průzkumy) je stejné procentuální zastoupení obyvatel, co preferují zimní čas.

$\alpha = 0.05$,
H_0: U větších měst, menších měst a obcí (3. průzkumy) je stejné procentuální zastoupení obyvatel, co preferují zimní čas.
H_A: U větších měst, menších měst a obcí (3. průzkumy) je stejné procentuální zastoupení obyvatel, co NEpreferují zimní čas.

In [12]:
```
df_z = df1.filter(items=['zimni'])
df_z.insert(len(df_z.columns), 'jiny', df1.iloc[:,[1,2,3]].sum(axis=1),\
    allow_duplicates=False)
df_z.insert(len(df_z.columns), 'sum_misto', df1.iloc[:,[4]].sum(axis=1),\
    allow_duplicates=False)

vetsi = df_z.iloc[[0,1],:].sum(axis=0)
mensi = df_z.iloc[[2,3],:].sum(axis=0)
vesnice = df_z.iloc[[4,5,6],:].sum(axis=0)
x = df_z.iloc[[8],:].sum(axis=0)

df_z = pd.DataFrame(([vetsi, mensi, vesnice, x]), columns=df_z.columns)
df_z.rename(index={0: 'vetsi', 1: 'mensi', 2: 'vesnice', 3: 'sum_cas'},\
    inplace=True)

# print(df_z.to_markdown())
```

	zimni	jiny	sum_misto
vetsi	834	1408	2242
mensi	559	709	1268
vesnice	300	375	675
sum_cas	1700	2514	4214

In [13]:

```
def dfzPerc2(df_z, toprint = False):
    N = df_z.at['sum_cas','sum_misto']

    df_z_perc = pd.DataFrame(data=np.zeros((4, len(df_z.columns))),\
        columns=df_z.columns)
    df_z_perc.rename(\
        index={0: 'vetsi', 1: 'mensi', 2: 'vesnice', 3: 'sum_cas'}\
            , inplace=True)


    suma = 0
    try:
        for i in range(0,4):
            for j in range(0,2):
                df_z_perc.iat[i,j] = \
                    ( df_z.iat[i,j] /  df_z.iat[i,2] ) * 100
                suma = suma + df_z_perc.iat[i,j]

            df_z_perc.iat[i,2] = suma
            suma = 0
    except:
        pass

    if toprint:
        print("--------------------")
        print("dfz procenta:")
        print(df_z_perc.to_markdown())
        print("--------------------")
    return df_z_perc

def ocekavaneDfz2(df_z, df_z_perc, toprint = False):
    dfm_z = pd.DataFrame(data=np.zeros((3, len(df_z.columns))),\
        columns=df_z.columns)
    dfm_z.rename(\
        index={0: 'vetsi', 1: 'mensi', 2: 'vesnice', 3: '0'},\
            inplace=True)
    # dfm.drop(labels='misto', axis=1, inplace=True)
    dfm_z.drop(labels='sum_misto', axis=1, inplace=True)

    for i in range(0,3):
        for j in range(0,2):
            # print(df_z_perc.iat[8,j] / 100 , df_z.iat[i,2] )
            dfm_z.iat[i,j] = (df_z_perc.iat[3,j] / 100) \
                * df_z.iat[i,2]

    if toprint:
        print("--------------------")
        print("ocekavane (m) dfm_z")
        print(dfm_z.to_markdown())
        print("--------------------")
    return dfm_z

def chiCalc2(df_z, dfm_z):
    chi_z = 0
    alpha = 0.05
    for i in range(0,3):
        for j in range(0,2):
            chi_z = chi_z + (((df_z.iat[i,j] - dfm_z.iat[i,j]) ** 2)\
                / dfm_z.iat[i,j])

    print("------------------")
```

```
    print("CHI^2 = " , chi_z,)

    critical_value = st.chi2.ppf(1-alpha, \
        df=(len(dfm_z.index)-1)*(len(dfm_z.columns)-1))

    print("Kriticka hodnota = ",critical_value, ",\
        stupne volnosti = ", \
            (len(dfm_z.index)-1)*(len(dfm_z.columns)-1))

    if chi_z > critical_value:
        print("H0 zamitam")
    else:
        print("H0 nezamitam")
    print("------------------")

df_z_perc = dfzPerc2(df_z, False)
dfm_z = ocekavaneDfz2(df_z, df_z_perc, False)
```

Tabulka procent:

	zimni	jiny	sum_misto
vetsi	37.1989	62.8011	100
mensi	44.0852	55.9148	100
vesnice	44.4444	55.5556	100
sum_cas	40.3417	59.6583	100

Tabulka očekávaných četností $m_{i,j}$

	zimni	jiny
vetsi	904.461	1337.54
mensi	511.533	756.467
vesnice	272.307	402.693

In [14]:
```
chiCalc2(df_z, dfm_z)
```

```
------------------
CHI^2 =  21.305122713194553
Kriticka hodnota =  5.991464547107979 , stupne volnosti =  2
H0 zamitam
------------------
```

---

e) H_0: U větších měst, menších měst a obcí (3. průzkumy) je stejné procentuální zastoupení nerozhodnutelných obyvatel.

$\alpha = 0.05$

H_0: U větších měst, menších měst a obcí (3. průzkumy) je stejné procentuální zastoupení nerozhodnutelných obyvatel

H_A: U větších měst, menších měst a obcí (3. průzkumy) NENÍ stejné procentuální zastoupení nerozhodnutelných obyvatel

In [15]:
```
df_z = df1.filter(items=['beznazoru'])
df_z.insert(len(df_z.columns), 'jiny', df1.iloc[:,[1,2,0]].sum(axis=1),\
    allow_duplicates=False)
df_z.insert(len(df_z.columns), 'sum_misto', df1.iloc[:,[4]].sum(axis=1),\
    allow_duplicates=False)


vetsi = df_z.iloc[[0,1],:].sum(axis=0)
mensi = df_z.iloc[[2,3],:].sum(axis=0)
vesnice = df_z.iloc[[4,5,6],:].sum(axis=0)
x = df_z.iloc[[8],:].sum(axis=0)

df_z = pd.DataFrame(([vetsi, mensi, vesnice, x]), columns=df_z.columns)
df_z.rename(index={0: 'vetsi', 1: 'mensi', 2: 'vesnice', 3: 'sum_cas'},\
    inplace=True)

# print(df_z.to_markdown())

df_z_perc = dfzPerc2(df_z)
dfm_z = ocekavaneDfz2(df_z, df_z_perc)
```

Tabulka četností:

	beznazoru	jiny	sum_misto
vetsi	337	1905	2242
mensi	144	1124	1268
vesnice	57	618	675
sum_cas	542	3672	4214

Tabulka četností v procentech:

	beznazoru	jiny	sum_misto
vetsi	15.0312	84.9688	100
mensi	11.3565	88.6435	100
vesnice	8.44444	91.5556	100
sum_cas	12.8619	87.1381	100

Tabulka očekávanýc četností $m_{i,j}$:

	beznazoru	jiny
vetsi	288.364	1953.64
mensi	163.089	1104.91
vesnice	86.8178	588.182

In [16]:
```
chiCalc2(df_z, dfm_z)
```

```
------------------
CHI^2 =  23.73062762989995
Kriticka hodnota =  5.991464547107979 , stupne volnosti =  2
H0 zamitam
------------------
```

f) Na základě odpovědí z okolí studenta zkuste určit z dat, zda student prováděl výzkum ve větším městě, menším městě nebo v obci.

Porovnejte výsledek se skutečností a okomentujte.

Příklad jsem počítal pomocí párového T-testu.

In [17]:
```
vetsi = df_f.iloc[[0,1],:].sum(axis=0)
mensi = df_f.iloc[[2,3],:].sum(axis=0)
vesnice = df_f.iloc[[4,5,6],:].sum(axis=0)
okoli = df_f.iloc[[7],:].sum(axis=0)

df_fvet = pd.DataFrame(([vetsi, okoli]), columns=df_f.columns)
df_fmen = pd.DataFrame(([mensi, okoli]), columns=df_f.columns)
df_fves = pd.DataFrame(([vesnice, okoli]), columns=df_f.columns)

df_fvet.rename(index={0: 'vetsi', 1: 'okoli'}, inplace=True)
df_fmen.rename(index={0: 'mensi', 1: 'okoli'}, inplace=True)
df_fves.rename(index={0: 'vesnice', 1: 'okoli'}, inplace=True)

# for i in [df_fvet, df_fmen, df_fves]:
#     i.loc[len(i.index)] = i.sum(axis=0)
#     i.rename(index={2: 'sum_cas'}, inplace=True)
#     print("{}: ".format(i.columns[0]))
#     print(i.to_markdown())
```

Četnost ve větších městech a okolí: :

	zimni	letni	stridat	beznazoru	sum_misto
vetsi	834	636	435	337	2242
okoli	7	12	6	4	29
sum_cas	841	648	441	341	2271

Četnost v menších městech a okolí:

	zimni	letni	stridat	beznazoru	sum_misto
mensi	559	363	202	144	1268
okoli	7	12	6	4	29
sum_cas	566	375	208	148	1297

Četnost ve vesnicích a okolí:

	zimni	letni	stridat	beznazoru	sum_misto
vesnice	300	210	108	57	675
okoli	7	12	6	4	29
sum_cas	307	222	114	61	704

In [18]:
```
for i in [df_fvet, df_fmen, df_fves]:
    lokalita = i.values[0]
    okoli = i.values[1]
    t_test_rest = st.ttest_rel(lokalita, okoli,\
        alternative='two-sided')
    print(i.index[0])
    print("Statistika = {}, P-Hodnota = {} \n".\
        format(t_test_rest.statistic, t_test_rest.pvalue))
```

```
vetsi
Statistika = 2.5836536202800007, P-Hodnota = 0.06109140982819672

mensi
Statistika = 2.4889989206823886, P-Hodnota = 0.06755677625424947

vesnice
Statistika = 2.454121510787069, P-Hodnota = 0.0701334654846285
```

Po provedení párového T-testu ani jedna z P-hodnot není menší než $\alpha = 0.05$, proto nemůžeme potvrdit, že platí H_0, tedy že jsem prováděl výzkum vě větším/menším městě nebo na vesnici. Nejblíže se však hodnotě $\alpha = 0.05$ blíží P-hodnota pro větší města. Svůj výzkum jsem ve skutečnosti prováděl částečně v Brně (větší město) a v Přerově (menší město, ale počtem obyvatel větší než Znojmo nebo Tišnov). Výsledky testu tedy celkem odpovídají realitě.

---

2. Regrese

a) Určete vhodný model pomocí zpětné metody a regresní diagnostiky.

Pro výpočet vhodného modelu jsem použil metodu, kdy nejprve vypočítám regresní model se všemi parametry a následně odeberu z modelu ten parametr, pro který vyšla ve Studentově T-testu největší P-hodnota. Pro tento test jsem volil hypotézy:
H_0 parametr $\beta_i = 0$
H_A parametr $\beta_i \neq 0$ Parametr β_i s největší P-hodnotou jsem odstranil, protože označuje koeficient, který není pro můj příklad statisticky signifikantní. Nikdy jsem ale neodtstanil konstantu β_1, i když měla největší P-hodnotu. Nastal-li tento případ, odstranil jsem parametr s druhou největší hodnotou. Po každém odstranění nějakého parametru jsem model přepočítal. Takto jsem opakoval, dokud nezbyly statisticky významné koeficienty ($P - hodnota \leq \alpha$) a konstanta β_1

In [19]:

```
x = (df2.iloc[:,0]).to_numpy()
y = (df2.iloc[:,1]).to_numpy()
z = (df2.iloc[:,2]).to_numpy()

xx = []
yy = []
zz = []

for j in x:
    j = j.replace(',','.')
    xx.append(float(j))

for j in y:
    j = j.replace(',','.')
    yy.append(float(j))

for j in z:
    j = j.replace(',','.')
    zz.append(float(j))

x = np.array(xx)
y = np.array(yy)
z = np.array(zz)

def calcModel(F, print_summary=False, useConstant=False):
    if useConstant:
        F = sm.add_constant(F)

    model = sm.OLS(z, F).fit()
    if print_summary:
        print(model.summary())
    else:
        print(model.rsquared)
    return model

F = np.column_stack((x, y, x**2, y**2, x*y))
model = calcModel(F, True, True)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 1.036e+04
Date:                Sun, 04 Dec 2022   Prob (F-statistic):           1.21e-91
Time:                        15:16:38   Log-Likelihood:                -335.87
No. Observations:                  70   AIC:                             683.7
Df Residuals:                      64   BIC:                             697.2
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.7385     14.691      0.050      0.960     -28.610      30.087
x1             1.0324      2.292      0.450      0.654      -3.546       5.611
x2             4.3030      4.327      0.994      0.324      -4.342      12.948
x3             4.0399      0.102     39.524      0.000       3.836       4.244
x4             4.8417      0.381     12.697      0.000       4.080       5.603
x5             6.0869      0.172     35.303      0.000       5.742       6.431
==============================================================================
Omnibus:                        0.829   Durbin-Watson:                   1.954
Prob(Omnibus):                  0.661   Jarque-Bera (JB):                0.824
Skew:                           0.019   Prob(JB):                        0.662
Kurtosis:                       2.470   Cond. No.                         839.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [20]:
```
F = np.column_stack((y, x**2, y**2, x*y))
model = calcModel(F, True, True)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 1.311e+04
Date:                Sun, 04 Dec 2022   Prob (F-statistic):           1.10e-93
Time:                        15:16:39   Log-Likelihood:                -335.98
No. Observations:                  70   AIC:                             682.0
Df Residuals:                      65   BIC:                             693.2
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.2817     10.616      0.498      0.621     -15.921      26.484
x1             4.0109      4.252      0.943      0.349      -4.481      12.503
x2             4.0810      0.046     88.867      0.000       3.989       4.173
x3             4.8417      0.379     12.775      0.000       4.085       5.599
x4             6.1161      0.159     38.521      0.000       5.799       6.433
==============================================================================
Omnibus:                        0.719   Durbin-Watson:                   1.945
Prob(Omnibus):                  0.698   Jarque-Bera (JB):                0.768
Skew:                           0.045   Prob(JB):                        0.681
Kurtosis:                       2.495   Cond. No.                         623.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [21]:
```
F = np.column_stack((x**2, y**2, x*y))
model = calcModel(F, True, True)
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.999
Model:	OLS	Adj. R-squared:	0.999
Method:	Least Squares	F-statistic:	1.751e+04
Date:	Sun, 04 Dec 2022	Prob (F-statistic):	1.18e-95
Time:	15:16:39	Log-Likelihood:	-336.46
No. Observations:	70	AIC:	680.9
Df Residuals:	66	BIC:	689.9
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	12.8588	6.935	1.854	0.068	-0.988	26.706
x1	4.0681	0.044	92.912	0.000	3.981	4.155
x2	5.1603	0.172	30.054	0.000	4.817	5.503
x3	6.1720	0.147	41.939	0.000	5.878	6.466

Omnibus:	1.262	Durbin-Watson:	1.931
Prob(Omnibus):	0.532	Jarque-Bera (JB):	1.218
Skew:	0.194	Prob(JB):	0.544
Kurtosis:	2.484	Cond. No.	389.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [22]:
```
F = np.column_stack((x**2, y**2, x*y))
model = calcModel(F, True, True)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 1.751e+04
Date:                Sun, 04 Dec 2022   Prob (F-statistic):           1.18e-95
Time:                        15:16:39   Log-Likelihood:                -336.46
No. Observations:                  70   AIC:                             680.9
Df Residuals:                      66   BIC:                             689.9
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         12.8588      6.935      1.854      0.068      -0.988      26.706
x1             4.0681      0.044     92.912      0.000       3.981       4.155
x2             5.1603      0.172     30.054      0.000       4.817       5.503
x3             6.1720      0.147     41.939      0.000       5.878       6.466
==============================================================================
Omnibus:                        1.262   Durbin-Watson:                   1.931
Prob(Omnibus):                  0.532   Jarque-Bera (JB):                1.218
Skew:                           0.194   Prob(JB):                        0.544
Kurtosis:                       2.484   Cond. No.                         389.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

Po opakovaném výpoču modelu a odstraňování nevýznamných parametrlů jsem získal model:

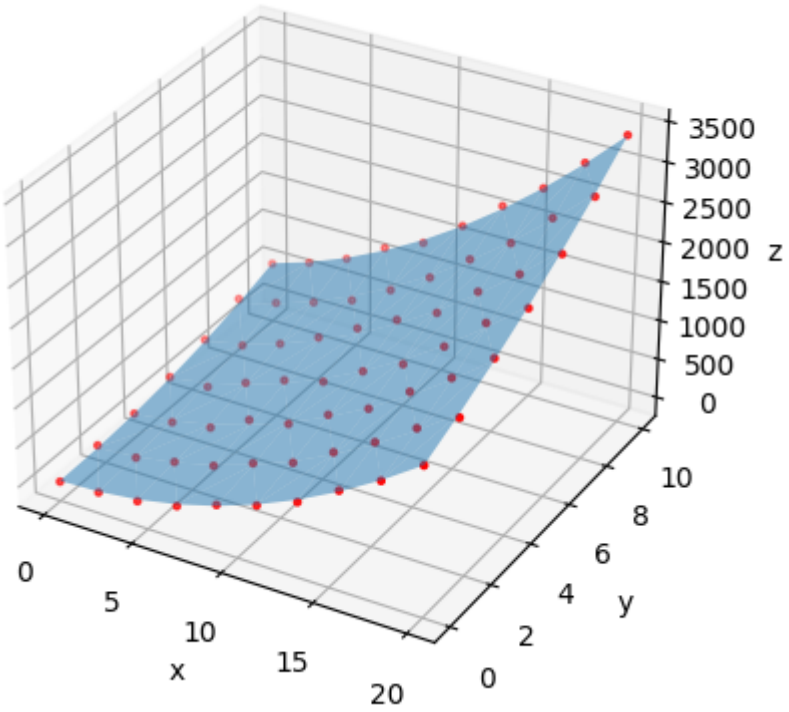
$$Z = \beta_1 + \beta_2 X^2 + \beta_3 Y^2 + \beta_4 XY$$

$$\equiv$$

$$z = 12.8588 + 4.0681x^2 + 5.1603y^2 + 6.1720xy$$

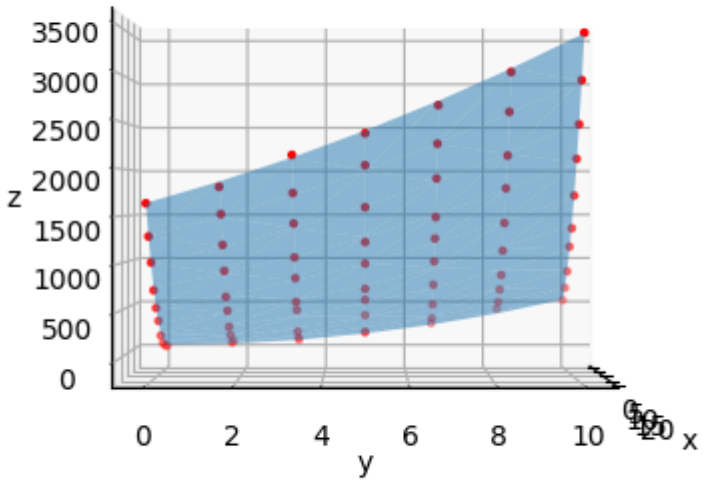
Výsledný regresní model vykreslíme:

In [23]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, marker='.', color='red')
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")

beta1 = model.params[0]
beta2 = model.params[1]
beta3 = model.params[2]
beta4 = model.params[3]

z_line = beta2 * x**2 + beta3 * y**2 + beta4 * x*y + beta1

surf = ax.plot_trisurf(x, y, z_line,  alpha=0.5, linewidth=0, antialiased=True)

plt.show()
```

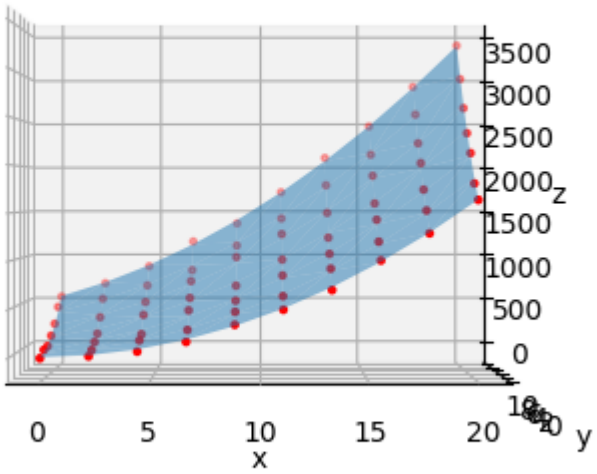


In [24]:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, marker='.', color='red')
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")

surf = ax.plot_trisurf(x, y, z_line,  alpha=0.5, linewidth=0, antialiased=True)
ax.view_init(0, 0)

plt.show()
```



In [25]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, marker='.', color='red')
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")

surf = ax.plot_trisurf(x, y, z_line,  alpha=0.5, linewidth=0, antialiased=True)
ax.view_init(0, -90)

plt.show()
```



b) Pro takto získaný model (dostatečný submodel) uveďte v jedné tabulce odhady regresních parametrů metodou nejmenších čtverců a jejich 95% intervaly spolehlivosti.

In [26]:
```
df_param = pd.DataFrame(model.params, columns=['parametry'])
df_param.index = ['beta0', 'beta1', 'beta2', 'beta3']
df_param.insert(1, 'interval L', model.conf_int()[:,0])
df_param.insert(2, 'interval P', model.conf_int()[:,1])

# print(df_param.to_markdown())
```

Tabulka získaných koeficientů β modelu a jejich 95% intervaly spolehlivosti:

	parametry	interval L	interval P
β_1	12.8588	-0.987861	26.7055
β_2	4.06806	3.98064	4.15548
β_3	5.16028	4.81747	5.5031
β_4	6.17201	5.87819	6.46584

c) Nestranně odhadněte rozptyl závisle proměnné.

In [27]:
```
print("Rozptyl závislé proměnné Z = ", model.mse_resid)
```

```
Rozptyl závislé proměnné Z =  928.8752852687966
```

d) Vhodným testem zjistěte, že vámi zvolené dva regresní parametry jsou současně nulové.

Pro ověření hypotézy, že dva regresní parametry jsou současně nulové jsem využil jsem F-test (Fisherovo-Snedecorovo rozdělení).

Vybrané parametry - β_2 a β_3

H_0 : dva regresní parametry jsou současně nulové

H_A : dva regresní parametry NEjsou současně nulové

In [28]:
```
f = model.f_test("x1=x2=0")

print((f))

if(f.pvalue > 0.05):
    print("H0 nezamitame")
else:
    print("H0 zamitame")
```

```
<F test: F=4912.283415242425, p=1.594596743515046e-72, df_denom=66, df_num=2>
H0 zamitame
```

e) Vhodným testem zjistěte, že vámi zvolené dva regresní parametry jsou stejné.

Vybrané parametry - β_3 a β_4

H_0 : dva regresní parametry jsou stejné

H_A : dva regresní parametry NEjsou stejné

In [29]:
```
print(model.f_test("x2 = x3"))
if(model.f_test("x2 = x3").pvalue > 0.05):
    print("H0 nezamitame")
else:
    print("H0 zamitame")
```

```
<F test: F=11.233116388450709, p=0.0013330450705401292, df_denom=66, df_num=1>
H0 zamitame
```

Výsledky tohoto testu jsou očekávané, protože β_3 = 5.16028, interval = <4.81747 ; 5.5031> a β_4 = 6.17201 , interval = <5.87819 ; 6.46584>