Explorativna analyza

Jakub Ševcech

Materialy dostupne na [https://github.com/sevo/pewe-presentations (https://github.com/sevo/pewe-presentations)](https://github.com/sevo/pewe-presentations)

Najskor oznamenie

[http://web.tuke.sk/fei-cit/wikt2017/challenge.html (http://web.tuke.sk/fei-cit/wikt2017/challenge.html)](http://web.tuke.sk/fei-cit/wikt2017/challenge.html)

Ciel prezentacie:

zrozumitelne vysvetlit a kategorizovat rozne techniky explorativnej analyzy dat podla toho, ake data spracovavame.

Povieme si o:

- Metrikach na zobrazenie roznych vlastnosti atributov
- Vizualizacii dat a o tom ako ich interpretovat
- Prejdeme si zopar statistickych testov na to aby sme zistili ci su v datach nejake vzory alebo ci existuju vyznamne rozdiely medzi vzorkami

Struktura prezentacie

Analyza po jednom (Univariate analysis)

- Spojite atributy
- Kategoricke atributy

Analyza po paroch atributov (Bivariate analysis)

- Spojite - Spojite
- Kategoricke - Spojite
- Kategoricke - Kategoricke

```
In [1]: %matplotlib inline
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import scipy.stats as stats
        import seaborn
        from sklearn import linear_model as lm
```

```
In [2]: plt.rcParams['font.family'] = 'DejaVu Sans'
        plt.rcParams['figure.figsize'] = (5, 3)
        from IPython.display import Image, SVG, display
```

# Analyza po jednom (Univariate analysis)

Zobrazenie vlastnosti jedneho atributu

Spojite atributy

Chceme zobrazit aky je tvar rozdelenia dat, ci sa zoskupuju okolo nejakeho **centra**, aka je **rozptylenost** hodnot

Zobrazenie centralnosti:

- mean (priemer)
- median (median, stredna hodnota, prostredna hodnota, centralna hodnota): hodnota, ktora rozdeluje vyssie a nizsie hodnoty
- mode (Modus, modálna hodnota, najpravdepodobnejšia hodnota): najcastejsia hodnota (hodnota s najvacsou pravdepodobnostou vyskytu)
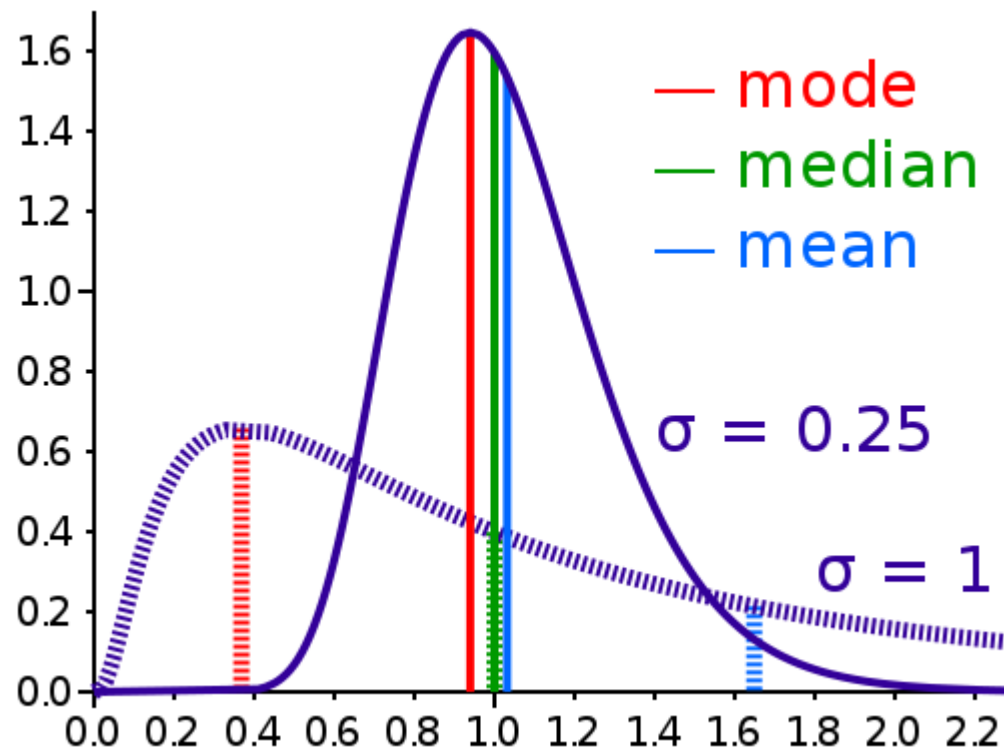
Rozptylenost

- range (rozsah): max - min
- quartile (kvartil): hodnota, od ktorej je 25% resp 75% hodnot vacsich
- percentile (percentil): hodnota, od ktorej je XX% hodnot vacsich
- inter quartile range (medzikvartilove rozpätie): rozdiel medzi 25% a 75% kvartilom, menej nachylne na outlierov ako rozsah
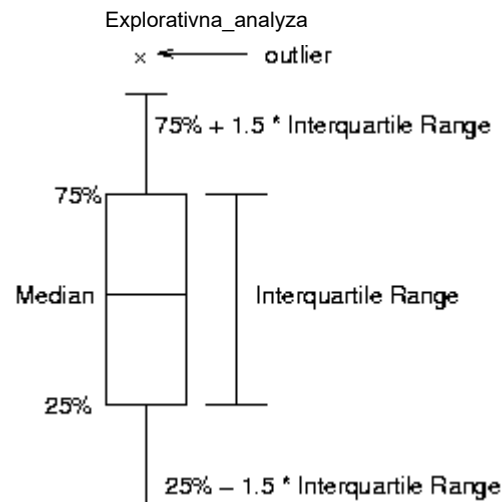
- variance (variancia): priemerna kvadraticka odchylka od priemeru

$$E[(X - E[X])^2]$$

- standard deviation (standardna odchylka): druha odmocnina variancie, je v jednotkach meranej premennej
- skewness (vychylenost?): metrika symetrickosti rozdelenia, ci je rozdelenie navazene na jednu stranu
- kurtosis (zplostenost?): ake mnozstvo dat je vo chvoste rozdelenia

(zdroj obrazku: https://taps-graph-review.wikispaces.com/Box+and+Whisker+Plots (https://taps-graph-review.wikispaces.com/Box+and+Whisker+Plots))

Skewness a Kurtosis

Skewness

Skewness je metrika toho, ako je rozdelenie symetricke. Uplne symetricke rozdelenie ma hodnotu skewness rovnu 0. V podstate to porovnava relativnu velkost dvoch chvostov rozdelenia. Rozdelenie naklonene do lava bude mat skewness vacsiu ako 0, naklonene doprava bude mat menej ako 1.

In [3]:
```python
sample_size = 10000

norm = stats.norm(0, 1)
x = np.linspace(-5, 5, 100)
sample = norm.rvs(sample_size)

plt.plot(x, norm.pdf(x))
plt.hist(sample, normed=True, bins=20)
plt.title("Normalne rozdelenie: ""Skewness %.5f" % (stats.skew(sample), ))
```
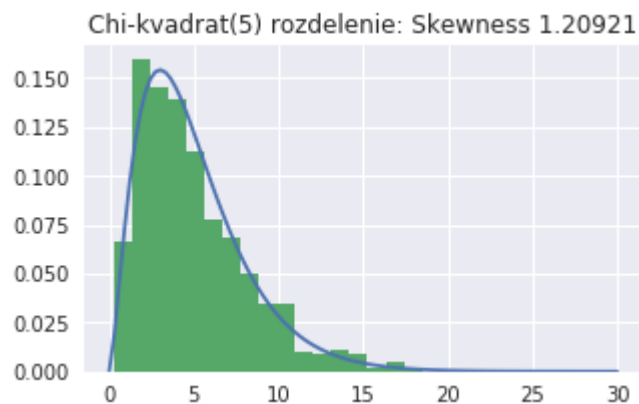
Out[3]: <matplotlib.text.Text at 0x7fbb173c6b38>
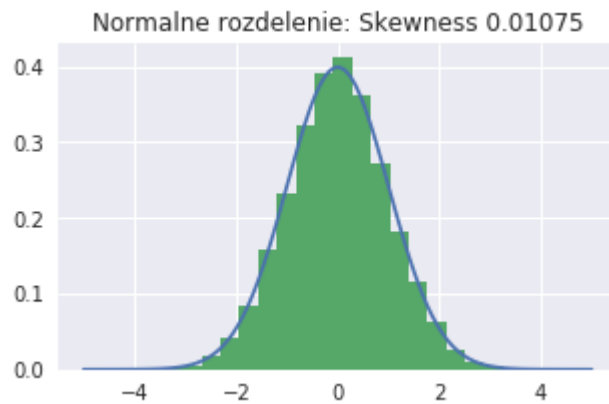
In [4]:
```python
sample_size = 1000

chi2 = stats.chi2(5)
x = np.linspace(0, 30, 100)
sample = chi2.rvs(sample_size)

plt.plot(x, chi2.pdf(x))
plt.hist(sample, normed=True, bins=20)
plt.title("Chi-kvadrat(5) rozdelenie: ""Skewness %.5f" % (stats.skew(sample), ))
```
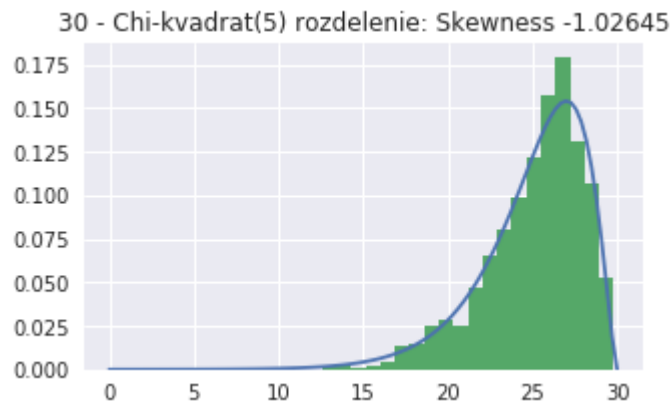
Out[4]: <matplotlib.text.Text at 0x7fbb172b4b70>

Chi-kvadrat(5) rozdelenie: Skewness 1.20921

In [5]:
```python
sample_size = 1000

chi2 = stats.chi2(5)
x = np.linspace(0, 30, 100)
sample = 30 - chi2.rvs(sample_size)

plt.plot(x, chi2.pdf(30 - x))
plt.hist(sample, normed=True, bins=20)
plt.title("30 - Chi-kvadrat(5) rozdelenie: ""Skewness %.5f" % (stats.skew(sample), ))
```

Out[5]: <matplotlib.text.Text at 0x7fbb171f2f98>

30 - Chi-kvadrat(5) rozdelenie: Skewness -1.02645

Kurtosis

Kurtosis hovori aka je kombinovana velkost chvostov. Meria mnozstvo dat sustredene v chvostoch. Velmi casto sa porovnava k hodnote kurtosis normalneho rozdelenia, ktora je 3. Ak je to viac ako 3, tak viac dat je sustredenych na okrajoch. Ak menej ako 3, tak je menej dat v okrajoch.
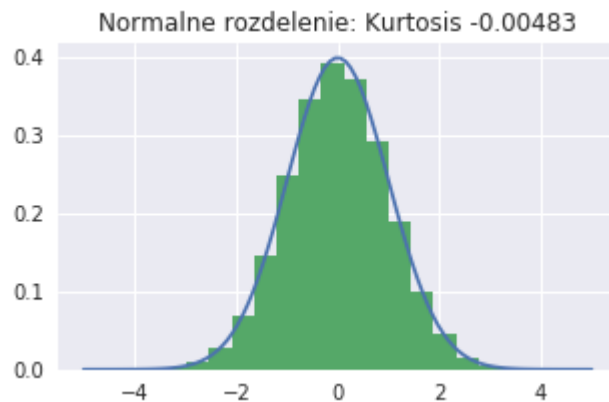
Casto sa pouziva aj excess kurtosis, co je rozdiel oproti normalnemu rozdeleniu, cize kurtosis - 3.

In [6]:
```python
sample_size = 100000

norm = stats.norm(0, 1)
x = np.linspace(-5, 5, 100)
sample = norm.rvs(sample_size)

plt.plot(x, norm.pdf(x))
plt.hist(sample, normed=True, bins=20)
plt.title("Normalne rozdelenie: ""Kurtosis %.5f" % (stats.kurtosis(sample), ))
```

Out[6]: <matplotlib.text.Text at 0x7fbb17130748>

Normalne rozdelenie: Kurtosis -0.00483

Ocakavali sme, ze dostaneme hodnotu kurtosis okolo 3. V skutocnosti funkcia stats.kurtosis pocita pri predvolenych nastaveniach excess kurtosis

In [7]:
```python
sample_size = 100000

norm = stats.norm(0, 1)
x = np.linspace(-5, 5, 100)
sample = norm.rvs(sample_size)

plt.plot(x, norm.pdf(x))
plt.hist(sample, normed=True, bins=20)
plt.title("Normalne rozdelenie: ""Kurtosis %.5f" % (stats.kurtosis(sample, fisher=False), ))
# musime prestavit parameter fisher na False
```

Out[7]:   <matplotlib.text.Text at 0x7fbb17059e10>
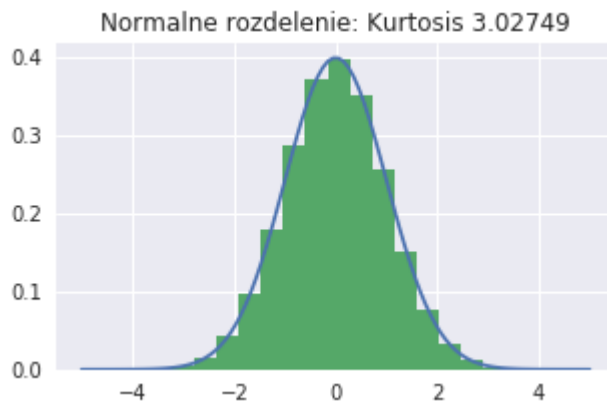
A teraz ukazka na nejakych rozdeleniach, kde vieme pekne kontrolovat mnozstvo dat v chvoste. Naprikl lognormal
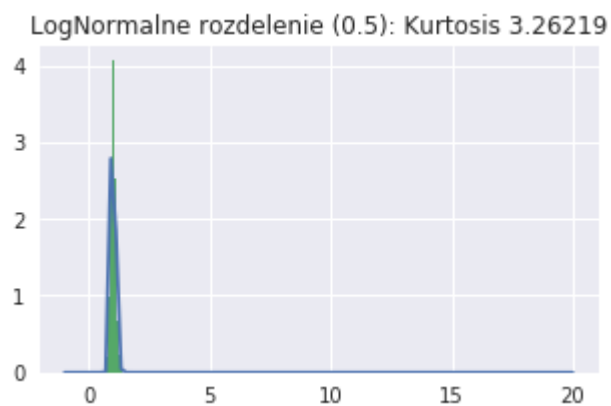
In [8]:
```
sample_size = 1000

lognorm = stats.lognorm(0.1)
x = np.linspace(-1, 20, 100)
sample = lognorm.rvs(sample_size)

plt.plot(x, lognorm.pdf(x))
plt.hist(sample, normed=True, bins=20)

plt.title("LogNormalne rozdelenie (0.5): ""Kurtosis %.5f" % (stats.kurtosis(sample, fisher=False), ))
```

Out[8]: <matplotlib.text.Text at 0x7fbb16ffabe0>

LogNormalne rozdelenie (0.5): Kurtosis 3.26219

In [9]:
```python
sample_size = 1000

lognorm = stats.lognorm(1)
x = np.linspace(-1, 20, 100)
sample = lognorm.rvs(sample_size)

plt.plot(x, lognorm.pdf(x))
plt.hist(sample, normed=True, bins=20)

plt.title("LogNormalne rozdelenie (1.0): ""Kurtosis %.5f" % (stats.kurtosis(sample, fisher=False), ))
```
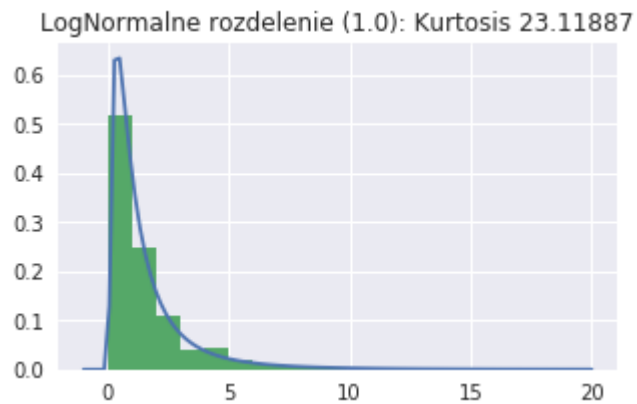
Out[9]: <matplotlib.text.Text at 0x7fbb16f2a860>

A teraz viacero rozdeleni v jednom obrazku aby sa to dalo dobre predstavit

In [10]:
```python
sample_size = 10000
x = np.linspace(-5, 50, 100)

dists = [
    ("Chi2(5)", stats.chi2(5).pdf(x), stats.chi2(5).rvs(sample_size)),
    ("Chi2(10)", stats.chi2(10).pdf(x), stats.chi2(10).rvs(sample_size)),
    ("Chi2(30)", stats.chi2(30).pdf(x), stats.chi2(30).rvs(sample_size)),
    ("50 - Chi2(5)", stats.chi2(5).pdf(50 - x), 50 - stats.chi2(30).rvs(sample_size)),
    ("Norm", stats.norm(0, 1).pdf(x), stats.norm(0, 1).rvs(sample_size)),
    ("lognorm(0.5)", stats.lognorm(0.5).pdf(x), stats.lognorm(0.5).rvs(sample_size))
]

labels = []

for name, dist, sample in dists:
    plt.plot(x, dist)
    labels.append("%s - Kurt: %.3f, Skew: %.3f" % (name, stats.kurtosis(sample, fisher=False), stats.skew(sample)

plt.legend(labels)
```
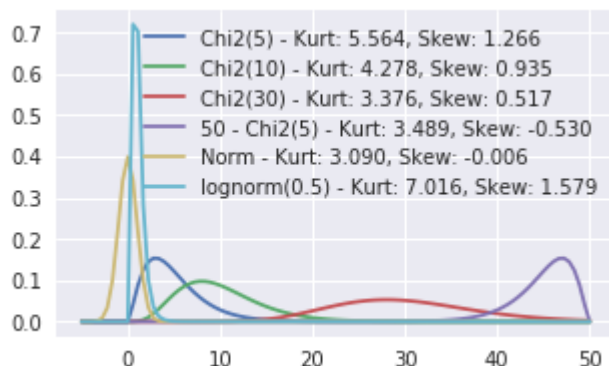
Out[10]: <matplotlib.legend.Legend at 0x7fbb16eaf390>
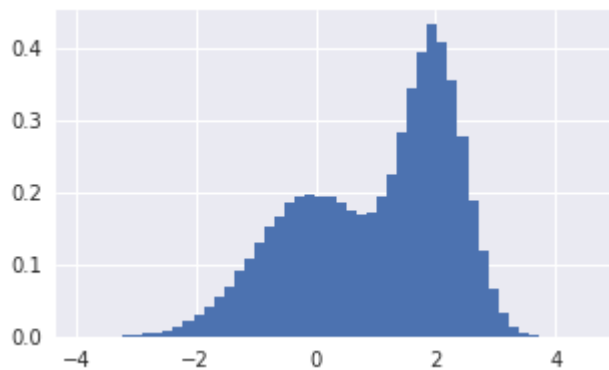

Analyza po jednom - spojite atributy - vizualizacia

Ako ste uz urcite pochopili, tak primarne sposoby vizualizacie su histogram a box plot
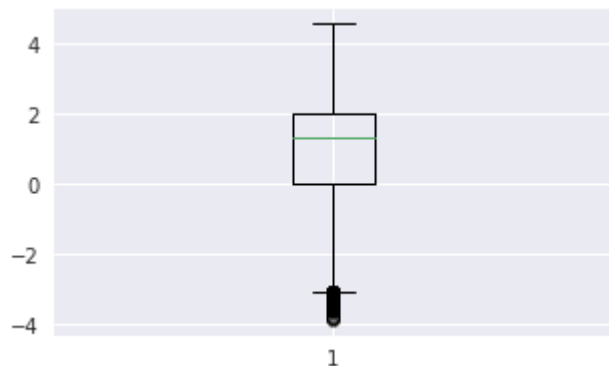
In [11]:
```python
sample_size = 100000

norm = stats.norm(0, 1)

x = np.linspace(-5, 5, 100)
sample = np.concatenate([
    stats.norm(0, 1).rvs(sample_size),
    stats.norm(2, 0.5).rvs(sample_size),
    ])
```

In [12]:
```python
_ = plt.hist(sample, normed=True, bins=50)
```


In [13]:
```python
plt.rc("lines", markeredgewidth=0.5)
_ = plt.boxplot(sample)
```


Osobne mam celkom rad spojenie boxplotu a histogramu do Violinplotu pretoze prehladne ukazuje tvar rozdelenia.

Castejsie sa ale pouziva vykreslenie dvoch obrazkov (aj histogram a aj boxplot). Spolu obsahuju viac informacii ako len jeden violinplot

```
In [14]: seaborn.violinplot(sample, orient='v')
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb16b77710>
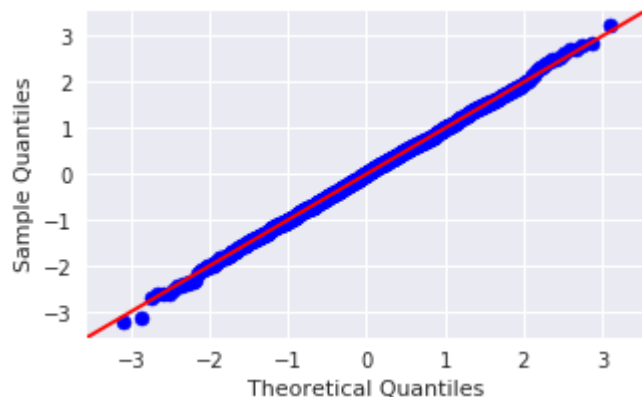

QQ-plot

```
In [15]: import statsmodels.api as sm
```

```
In [16]: sample_size = 1000

x = np.linspace(-5, 5, 100)
sample = stats.norm(0, 1).rvs(sample_size)
# sample = stats.norm(10, 5).rvs(sample_size)
```

In [17]: `_ = sm.ProbPlot(sample, fit=True).qqplot(line='45')`

QQ-plot je vizualna metoda na urcenie, ci dve datove sady pochadzaju z rovnakeho rozdelenia (probability plot porovnava datovu sadu s teoretickym rozdelenim).

Probability plot porovnava voci zvolenemu teoretickemu rozdeleniu. V tomto pripade normalnemu.

Porovnava kvantily rozdeleni.

Osy su v jednotkach porovnavanych datovych sad

Bod na obrazku zobrazuje hodnotu kvantilu v prvm a druhom porovnavanom datasete.

Ak su datasety rovnako velke, tak je to len vykreslenie usporiadanych datasetov pomocou scatterplotu. Ak je jeden mensi, tak sa ten pouzije na urcenie kvartilov a hodnoty z druheho (vacsieho) datasetu sa interpoluju

Na ake otazky vie odpovedat?

- Pochadzaju pozorovania z rovnakeho rozdelenia?
- Maju rozdelenia rovnaku skalu (priemer, standardnu odchylku)?
- Je tvar porovnavanych rozdeleni podobny (rovna ciara, bez ohladu na jej posunutie a sklon)?
- Maju rozdelenia podobne vlastnosti skewness a kurtosis?

In [ ]:

Pri zakladnom nastaveni porovnava s normalnym rozdelenim. Co nam velmi nepomoze v pripade, ak nase pozorovania su z uplne ineho

rozdelenia. Len nam to povie, ze je to nejake ine rozdelenie.

```
In [18]: x = np.linspace(-5, 5, 100)
         lognorm_sample = stats.lognorm(0.5).rvs(sample_size)
         _ = sm.ProbPlot(lognorm_sample, fit=True).qqplot(line='45')
```
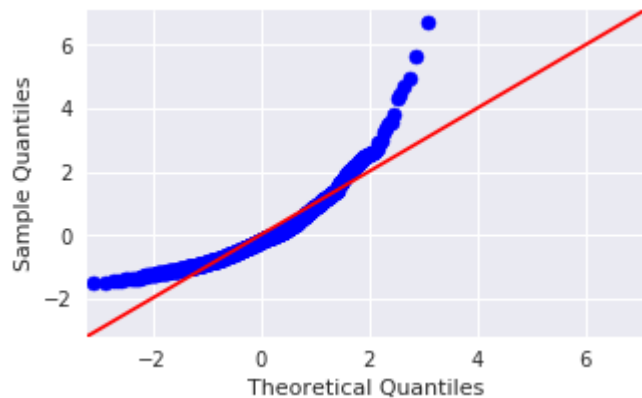

Ocividne je hovadina porovnavat tieto nase pozorovania s uplne inou distribuciou, ale moze sa to hodit, ked mame ine pozorovania

```
In [19]: _ = sm.ProbPlot(lognorm_sample, dist=stats.lognorm, fit=True).qqplot(line='45')
```
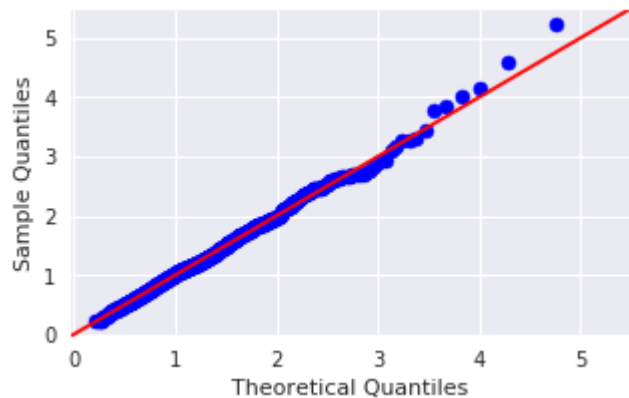

Nastastie vieme zmenit teoreticke rozdelenie a mozeme sa porovnan s nim

In [20]:
```python
import statsmodels.api as sm
x = stats.norm(8.25, 2.75).rvs(1000)
y = stats.lognorm(0.5).rvs(1000)
pp_x = sm.ProbPlot(x, fit=True)
pp_y = sm.ProbPlot(y, fit=True)
_ = pp_x.qqplot(line='45', other=pp_y)
```
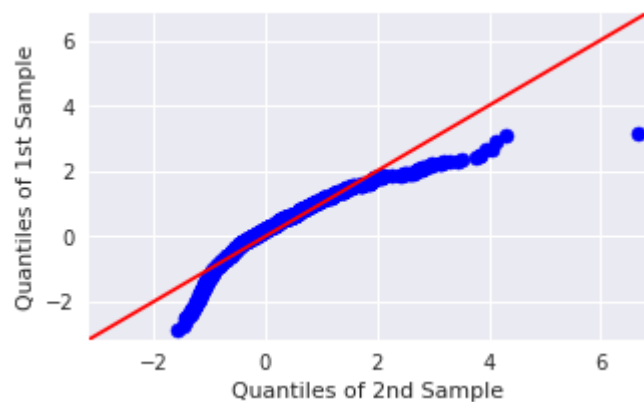

Velmi pekny prispevok o tom ako interpretovat QQ-plot

https://stats.stackexchange.com/questions/101274/how-to-interpret-a-qq-plot (https://stats.stackexchange.com/questions/101274/how-to-interpret-a-qq-plot)
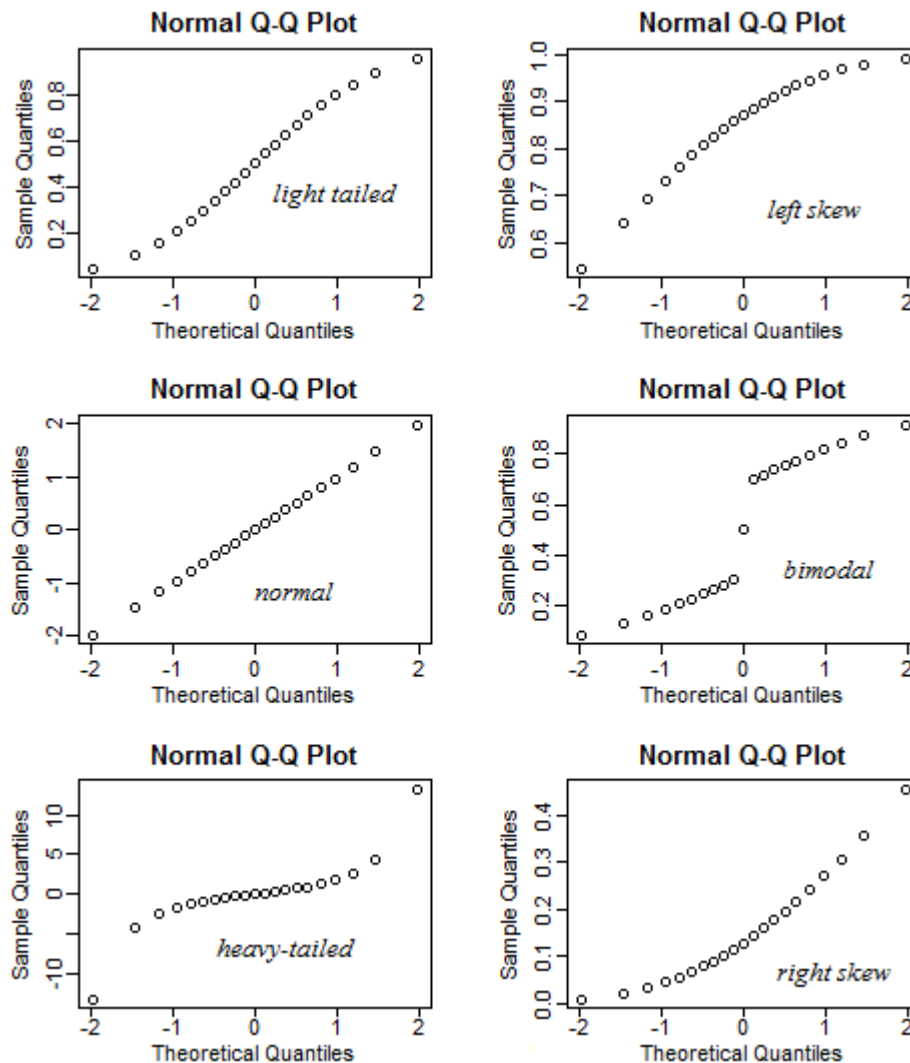
Analyza po jednom - kategoricke atributy

Tu je najcastejsim sposobom zobrazenia frekvencna tabulka zobrazujuca bud pocty pozorovani per unikatna hodnota atrinutu alebo pomer voci celkovemu poctu pozoorvani.

Graficka vizualizacia je bar plot

In [21]:
```python
diamonds = pd.read_csv('data/diamonds.csv')
diamonds.head()
```

Out[21]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [22]:
```python
diamonds.color.value_counts()
```

Out[22]:
```
G    11292
E     9797
F     9542
H     8304
D     6775
I     5422
J     2808
Name: color, dtype: int64
```

In [23]:
```python
diamonds.color.value_counts() / len(diamonds)
```

Out[23]:
```
G    0.209344
E    0.181628
F    0.176900
H    0.153949
D    0.125603
I    0.100519
J    0.052058
Name: color, dtype: float64
```

In [24]:
```
diamonds.color.value_counts().plot(kind='bar')
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb16eb8e10>

A samozrejme sa daju pouzit aj dalsie podobne typy na zobrazenie tychto istych dat. Napriklad kolacovy graf, aj ked ten sa cita trochu tazsie ako stlpcovy graf

Ak je atribut ordinalny, tak moze mat zmysel usporiadat hodnoty.

//asi nema zmysel usporiadavat farby, ale tie ich pomenovania ma k tomu nabadaju

In [25]:
```
diamonds.color.value_counts()[['D', 'E', 'F', 'G', 'H', 'I', 'J']].plot(kind='bar')
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb144d6438>
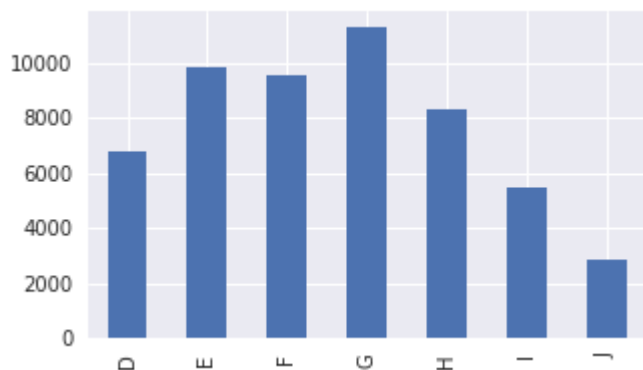
Ked mam atribut, v ktyorom su nejake zavislosti (casova naslednost, ine poradie), tak moze mat zmysel zobrazit to pomocou ciaroveho

grafu, ktory lepsie zobrazuje zmenu.

Pozor, pouzivat to len v pripade ak su tam zavyslosti medzi hodnotami.

```
In [26]: diamonds.color.value_counts()[['D', 'E', 'F', 'G', 'H', 'I', 'J']].plot(kind='line')
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb14198c88>
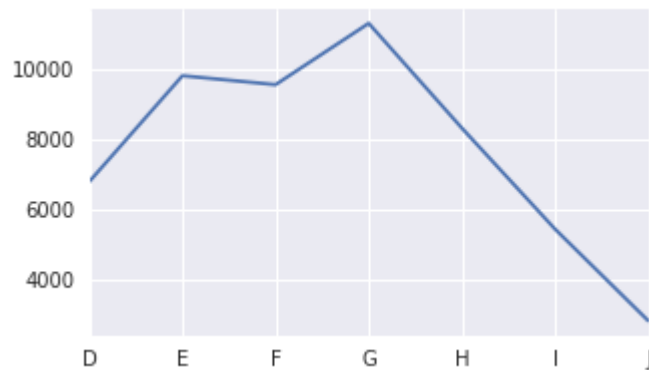
Najcastejsie sa tento graf pouziva pri casovych radoch.

Napriklad sa da pouzit na zobrazenie vyvoja nejakej meranej hodnoty v case. V tomto konkretnom pripade je to obsah NO2 vo vzduchu

In [27]:
```python
import airbase
no2 = airbase.load_data()

no2["2012-01-01 00:00": "2012-01-02 00:00"].FR04012.plot()
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb141904e0>
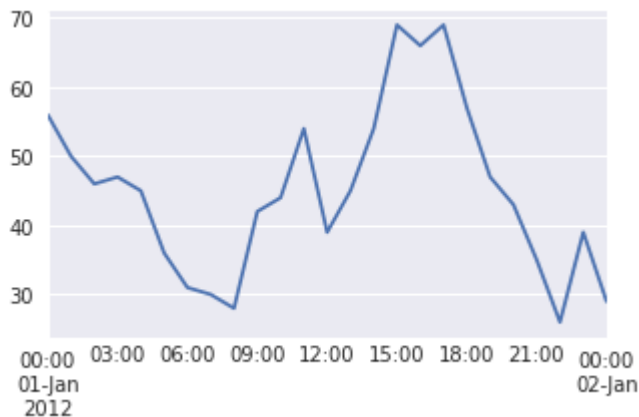
Obcas ma dokonca zmysel prekryvat viacere ciary cez seba, na to aby ste zobrazili viacero atributov / metrik / casovych obdobi ...

In [28]:
```python
no2["2012-01-01 00:00": "2012-01-02 00:00"].plot()
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb13fa2828>

In [29]: `no2.loc['2009':, 'FR04037'].resample('M').agg(['mean', 'median']).plot()`

Out[29]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fbb14033a58>`
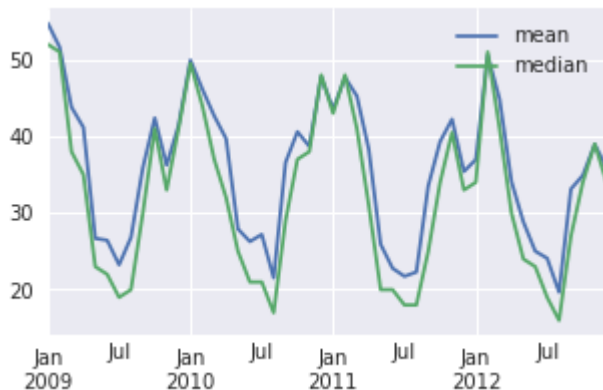
Analyza po paroch atributov (Bivariate analysis)

Analyza vztahov dvoch atributov.

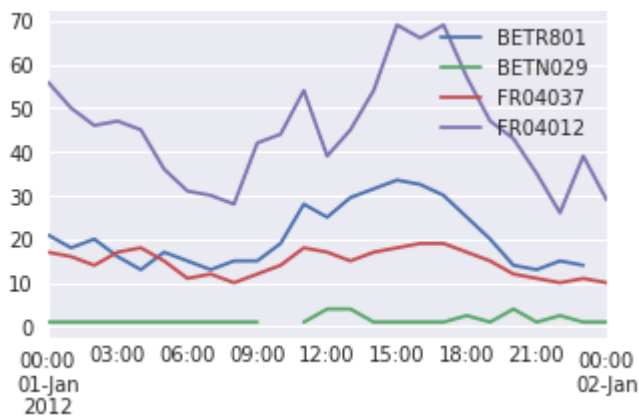Kedze mame spojite a kategoricke atributy, tak mozu vzniknut 3 rozne kombinacie. Pre kazdu z nich existuju metody, ktore mozeme pouzit na opisanie ich vztahov a na ich vizualizaciu.

- Spojite - Spojite
- Kategoricke - Spojite
- Kategoricke - Kategoricke

Spojity - Spojity

Scatter plot

Najcastejsi sposob ako vizualizovat vztah dvoch spojitych atributov. Zobrazuje rozmiestnenie v priestore hodnot.

Da sa pouzit aj na vizualizovanie skupin pozorovani. Typicky na to aby sme zistili, ci su v datach nejake prirodzene zhluky

In [30]:
```python
iris = seaborn.load_dataset("iris")

plt.scatter(iris.sepal_length, iris.sepal_width)
```

Out[30]: <matplotlib.collections.PathCollection at 0x7fbb16b47dd8>
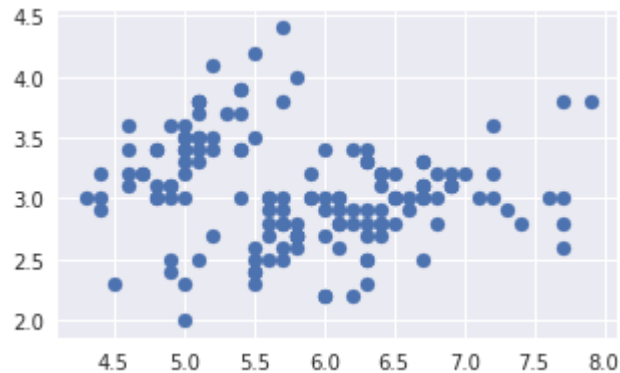
Ak ide o oznackovane data, tak ich vieme ofarbit pomocou znacky a pozriet sa na to, ci sa daju na zaklade tychto atributov rozdelit do skupin.

In [31]:
```python
for name, group in iris.groupby("species"):
    plt.scatter(group.sepal_length, group.sepal_width, label=name)

plt.legend()
```

Out[31]: <matplotlib.legend.Legend at 0x7fbb16cda748>
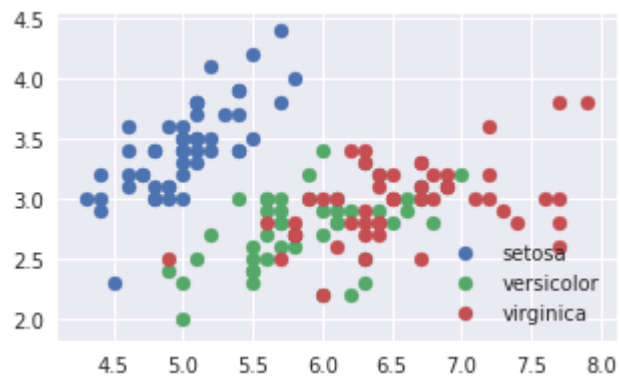
Ak chcem vizualizovat vztah vsetkych kombinacii atributov, tak mozem spravit pairplot

Pozor, pri velkom pocte atributov je to dost necitatelne a velmi dlho sa to vytvara

```
In [32]: seaborn.pairplot(iris)
```

Out[32]: <seaborn.axisgrid.PairGrid at 0x7fbb16da05c0>
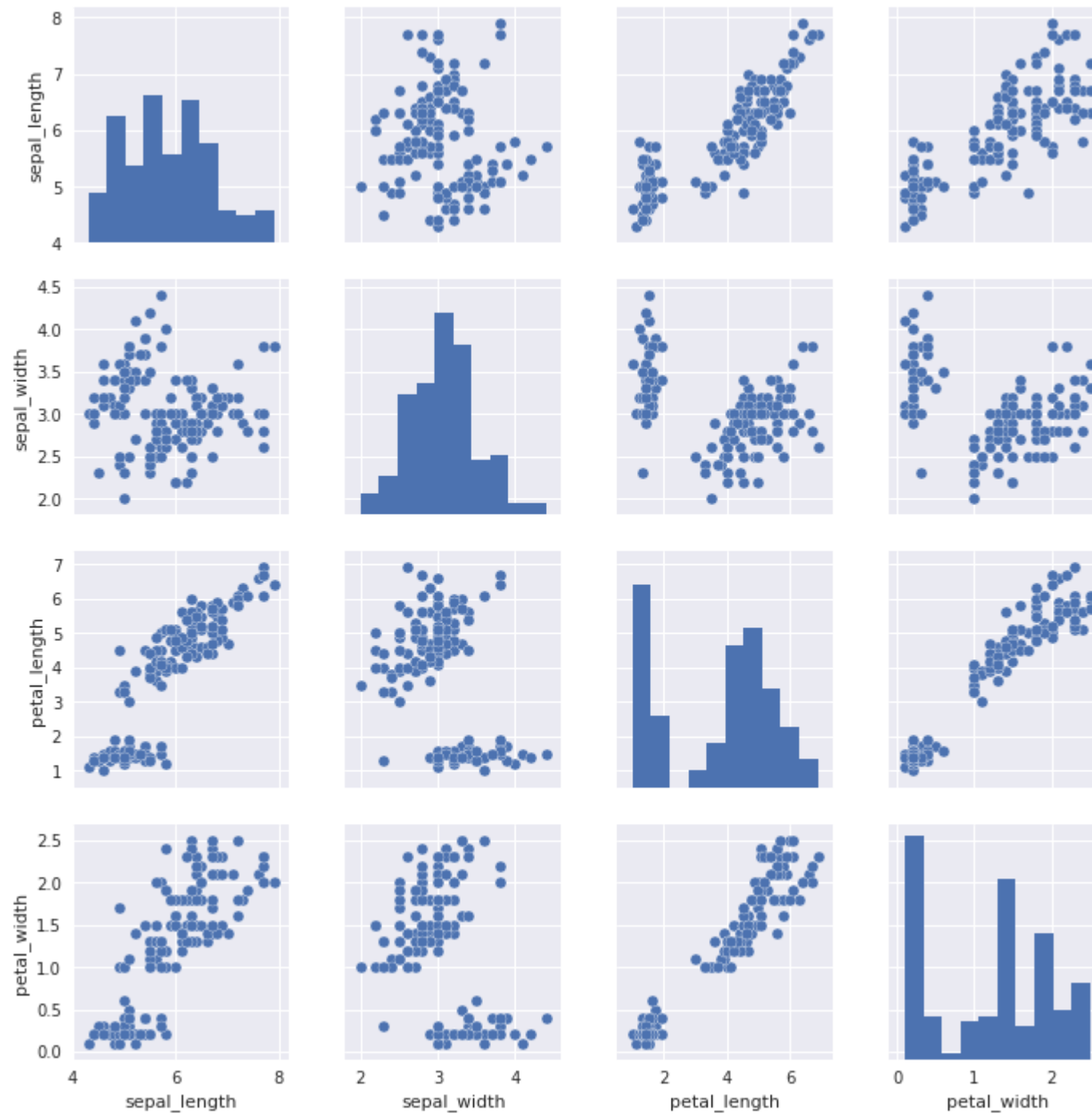
Samozrejme sa da aj pairplot ofarbit triedou

In [33]: `seaborn.pairplot(iris, hue="species")`

Out[33]: `<seaborn.axisgrid.PairGrid at 0x7fbb13aa8278>`
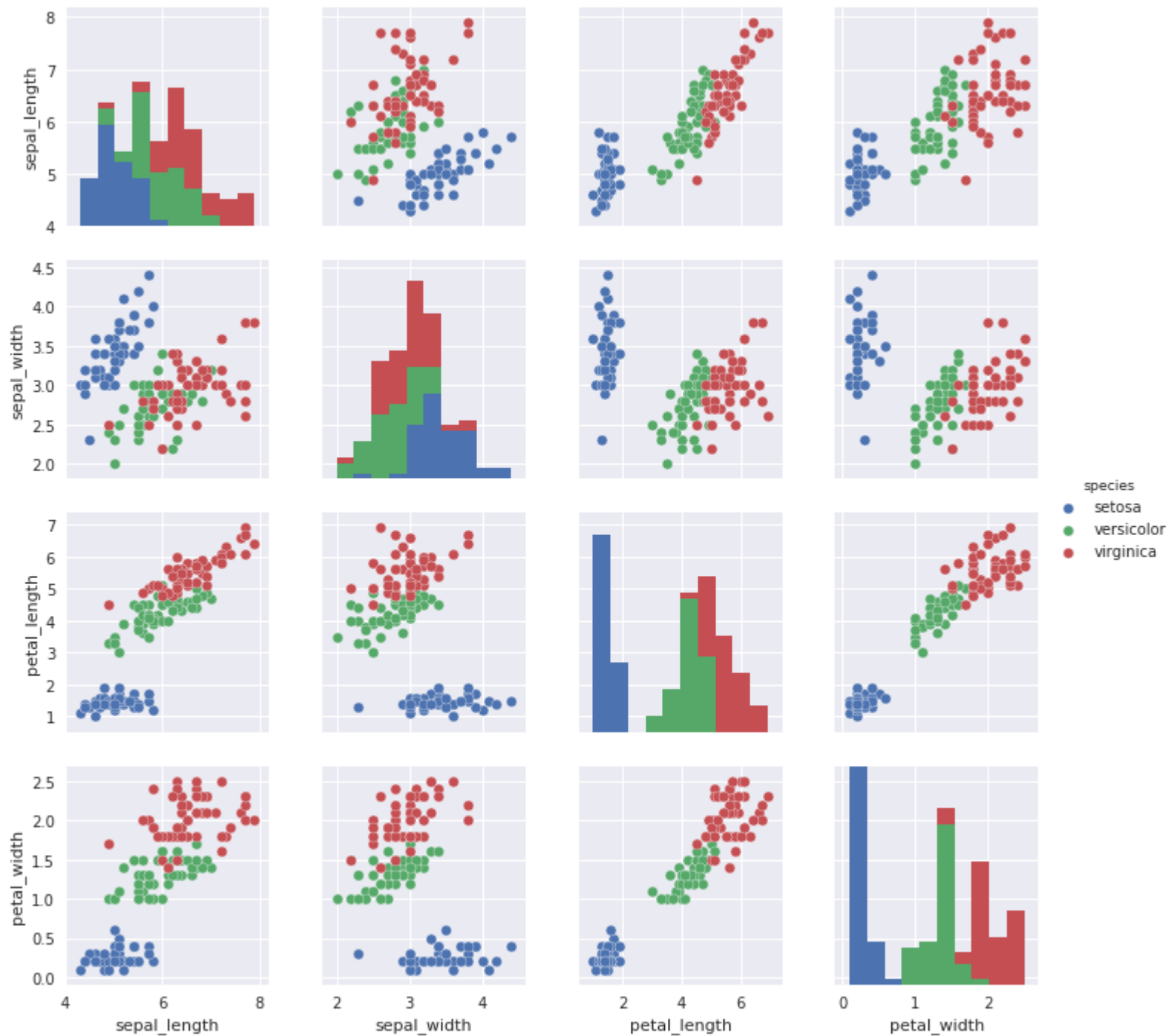
ak by som chcel manualne vytvarat nejake pravidla na klasifikaciu, tak z tohto obrazku by som uz nejake vedel vyrobit

Scatterplot sa da pouzit nie len na vizualizovanie zhlukov ale aj korelacii / zavislosti.
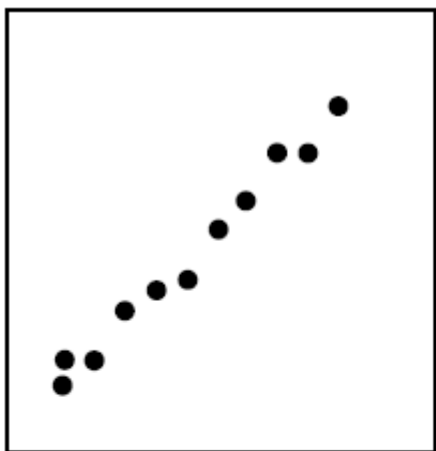
Strong positive correlation
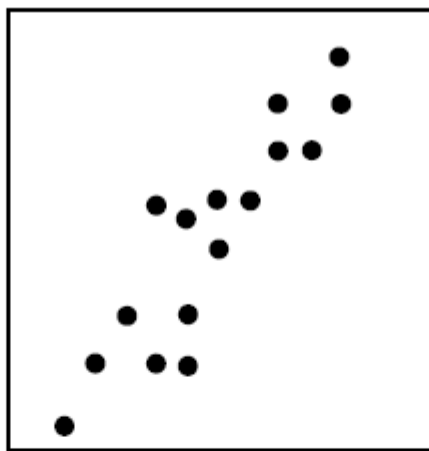
Moderate positive correlation
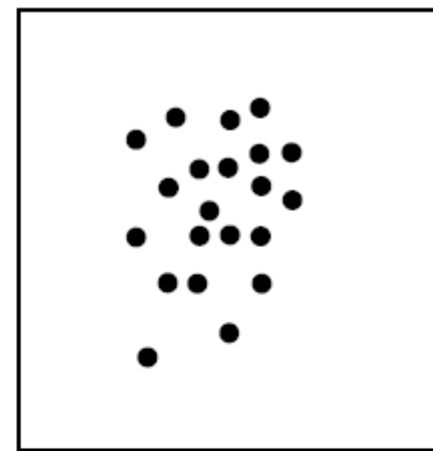
No correlation
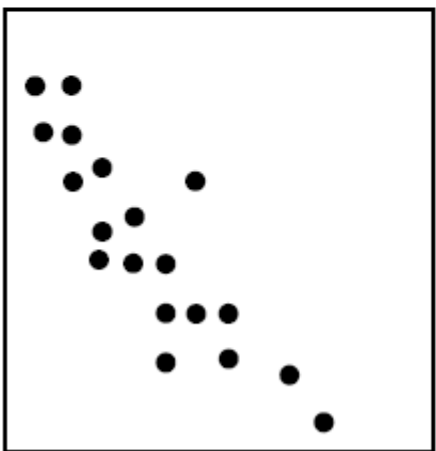
Moderate negative correlation
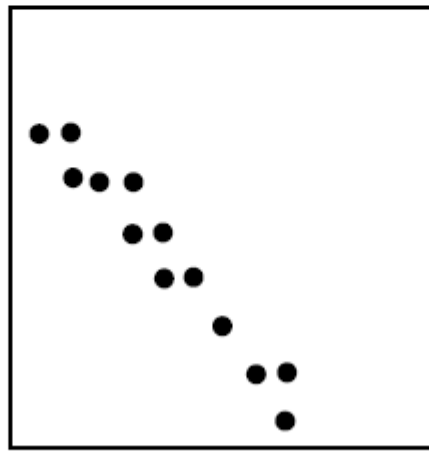
Strong negative correlation
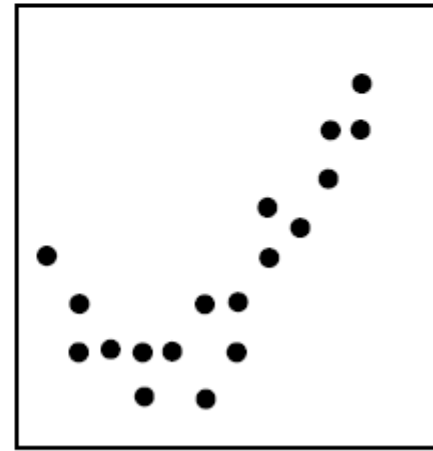
Curvilinear relationship

Nedokaze vsak kvantifikovat silu vztahu. Na to potrebujeme nejaku inu metriku - korelaciu.

Zdroj obrazku: https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#one (https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#one)

Korelacia

Hodnota v rozsahu [-1, 1], ktora hovori o tom, aky silny linearny vztah je medzi atributmi.

- -1 perfektna negativna korelacia
- 0 ziadna korelacia
- 1 perfektna kladna korelacia

Pearsnov korelacny koeaficient:

$$corr(X, Y) = \frac{cov(X, Y)}{E[X]E[Y]} = \frac{E[(X - E[X])(Y - E[Y])]}{E[X]E[Y]}$$

```
In [34]: car_data = pd.read_csv('data/auto-mpg.data', delim_whitespace=True,
                        names = ['mpg', 'cylinders', 'displacement','horsepower',
                                'weight', 'acceleration', 'model_year', 'origin', 'name'],
                        na_values='?'
                    )
        car_data.head()
```

Out[34]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino

In [35]: `car_data.describe()`

Out[35]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.000000

In [36]: `car_data = car_data.dropna()`

In [37]:
```
seaborn.regplot(x="horsepower", y="mpg", data=car_data)
print("Pearson correlation: %.3f" % car_data.horsepower.corr(car_data.mpg))
```

Pearson correlation: -0.778
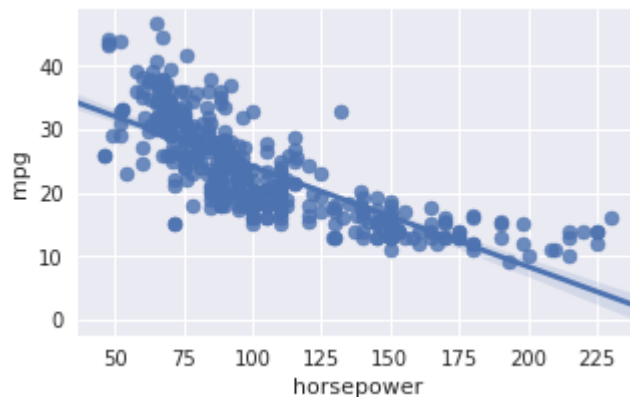
Pozor sklon regresnej ciary nehovori o sile korelacie. Len o smere.

In [38]:
```python
regr = lm.LinearRegression()
regr.fit(car_data.horsepower.values.reshape(len(car_data),1), car_data.mpg)

seaborn.regplot(x="horsepower", y="mpg", data=car_data)
print("Pearson correlation: %.3f, Regresion coefficient: %.3f" % (car_data.horsepower.corr(car_data.mpg), regr.co
```

Pearson correlation: -0.778, Regresion coefficient: -0.158
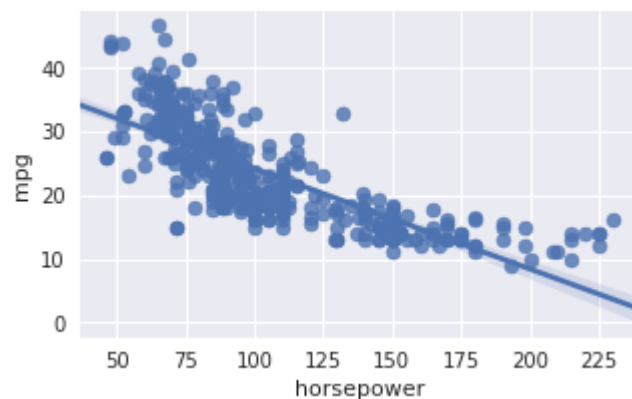
Sklon regresnej krivky je uplne iny ako velkost korelacie. Len znamienko indikujuce smer je rovnake.

In [39]:
```python
x = np.arange(100)
y = np.ones(100)

synth_data = pd.DataFrame({
    'x': x,
    'y': y
})

regr = lm.LinearRegression()
regr.fit(synth_data.x.values.reshape(100,1), synth_data.y)

seaborn.regplot(x="x", y="y", data=synth_data)
print("Pearson correlation: %.3f, Regresion coefficient: %.3f" % (synth_data.x.corr(synth_data.y), regr.coef_[0])
```

Pearson correlation: nan, Regresion coefficient: 0.000

Toto je extremny pripad, kde korelaciu nevieme spocitat, kedze jedna hodnota je len konstanta (nulova variancia), pricom je to pekna rovna ciara so sklonom 0.

In [40]:
```python
x = np.arange(100)
y = x * (-3.14) + 3

synth_data = pd.DataFrame({
    'x': x,
    'y': y
})

regr = lm.LinearRegression()
regr.fit(synth_data.x.values.reshape(100,1), synth_data.y)

seaborn.regplot(x="x", y="y", data=synth_data)
print("Pearson correlation: %.3f, Regresion coefficient: %.3f" % (synth_data.x.corr(synth_data.y), regr.coef_[0]))
```

Pearson correlation: -1.000, Regresion coefficient: -3.140
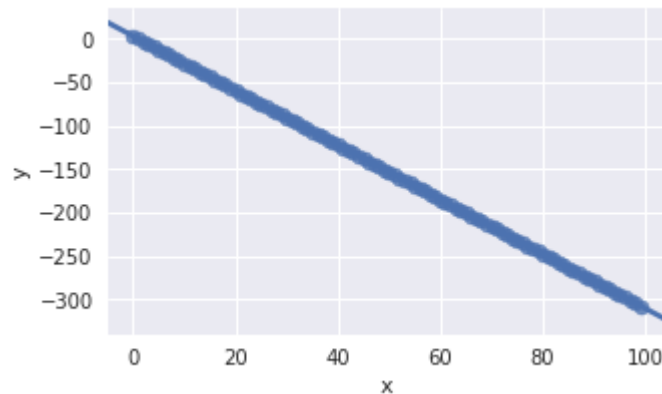
Tu je perfektne linearny vztah, kde je jasne vidiet, ze ten sklon je uplne iny.

In [41]:
```python
x = np.arange(100)
y = x + stats.norm(0,1).rvs(100)

synth_data = pd.DataFrame({
    'x': x,
    'y': y
})

regr = lm.LinearRegression()
regr.fit(synth_data.x.values.reshape(100,1), synth_data.y)

seaborn.regplot(x="x", y="y", data=synth_data)
print("Pearson correlation: %.3f, Regresion coefficient: %.3f" % (synth_data.x.corr(synth_data.y), regr.coef_[0]]
```

Pearson correlation: 0.999, Regresion coefficient: 0.998
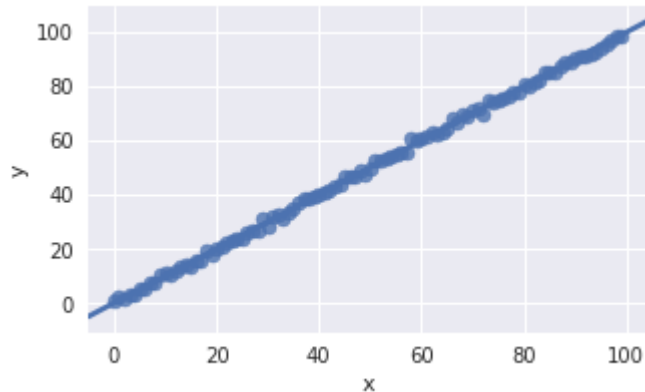
Aj ked pridame trochu sumu, tak je to velmi podobne
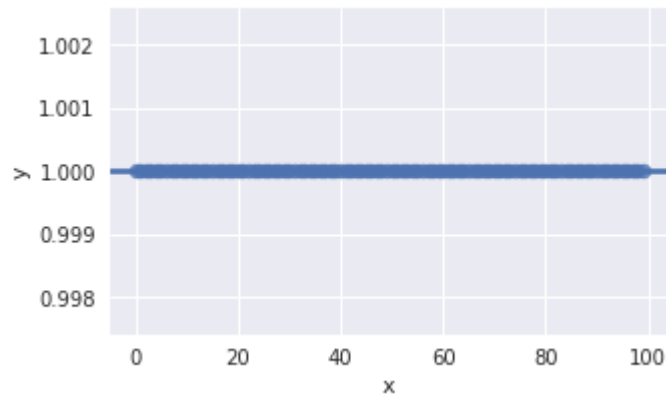
In [42]:
```python
x = np.arange(100)
y = x + stats.norm(0,30).rvs(100)

synth_data = pd.DataFrame({
    'x': x,
    'y': y
})

regr = lm.LinearRegression()
regr.fit(synth_data.x.values.reshape(100,1), synth_data.y)

seaborn.regplot(x="x", y="y", data=synth_data)
print("Pearson correlation: %.3f, Regresion coefficient: %.3f" % (synth_data.x.corr(synth_data.y), regr.coef_[0])
```

Pearson correlation: 0.699, Regresion coefficient: 1.109
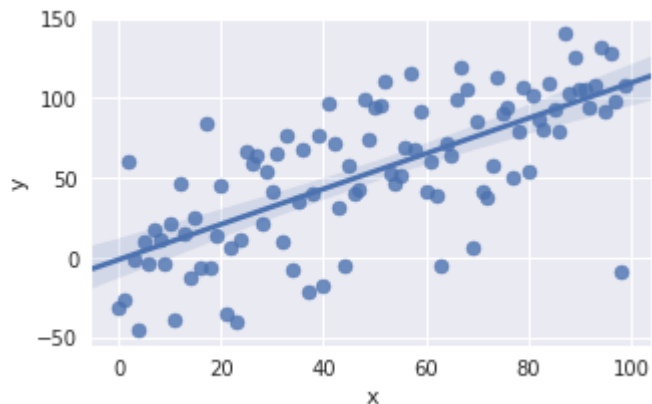
Ked ale pridame toho sumu trochu viac, tak sa nam ta korelacia zacne poriadne kazit a znova vidime, ze to s tym smerom nesuvisi.

Spat k datasetu o autach

Ak by sme sa chceli pozriet na korelaciu medzi vsetkymi dvojicami atributov, tak sa da pouzit takato korelacna matica

In [43]: `car_data.corr()`

Out[43]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.580541	0.565209
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647	-0.568932
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855	-0.614535
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361	-0.455171
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120	-0.585005
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316	0.212746
model_year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000	0.181528
origin	0.565209	-0.568932	-0.614535	-0.455171	-0.585005	0.212746	0.181528	1.000000

A da sa aj vykreslit pomocou teplotnej mapy aby sa nam lepsie citala

In [44]:
```
fig, ax = plt.subplots(figsize=(10,8))
seaborn.heatmap(car_data.corr(), ax=ax, annot=True, fmt=".3f")
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb114f9f98>
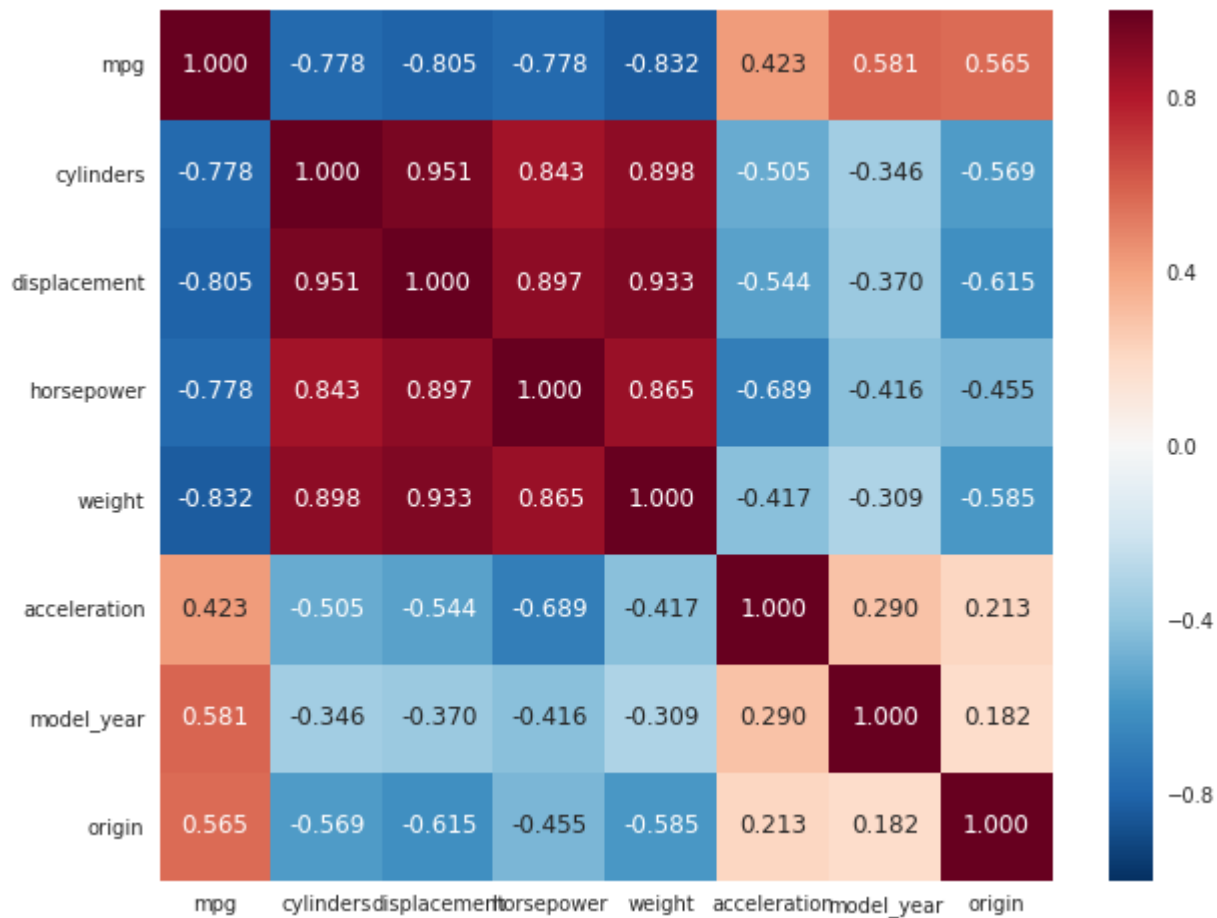
Kategoricky - Kategoricky

- Two-way table
- Heatmap
- Stacked bar plot
- Chi-kvadrat testy

In [45]:
```python
titanic = pd.read_csv('data/titanic/train.csv')
titanic.head()
```

Out[45]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [46]:
```python
# Frekvencna tabulka
titanic["Survived"].value_counts()
```

Out[46]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [47]:
```python
survived_class = pd.crosstab(index=titanic["Survived"],
                             columns=titanic["Pclass"])
survived_class.index= ["died","survived"]
survived_class
```

Out[47]:

Pclass	1	2	3
died	80	97	372
survived	136	87	119

```
In [48]: seaborn.heatmap(survived_class, annot=True, fmt="d")
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb11334588>
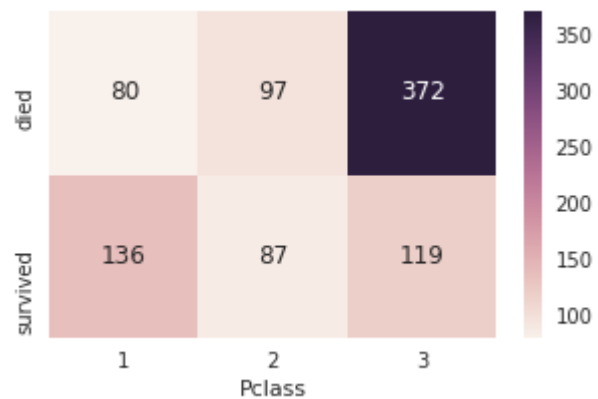
Ak by sme chceli zobrazit percentualny podiel, tak sa da normalizovat po riadkoch, stlpcoch, alebo vsetkych datach

In [49]:
```python
survived_class_perc = pd.crosstab(index=titanic["Survived"],
                                  columns=titanic["Pclass"],
                                   normalize='index') #'columns', 'all'
survived_class_perc.index= ["died","survived"]

seaborn.heatmap(survived_class_perc, annot=True, fmt=".4f")
survived_class_perc
```

Out[49]:

Pclass	1	2	3
died	0.145719	0.176685	0.677596
survived	0.397661	0.254386	0.347953
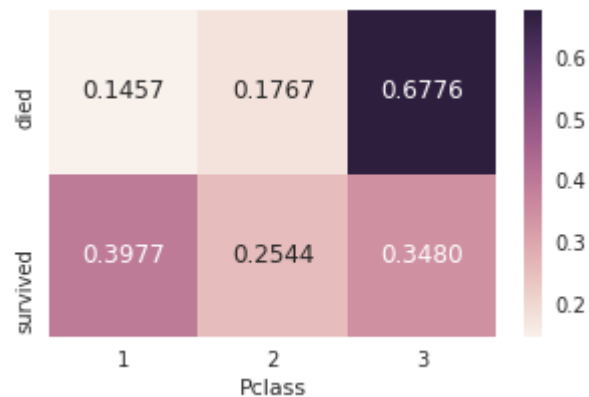
Daju sa tiez spravit aj tabulky s vyssimi dimenziami. Tam sa ale uz velmi rychlo straca prehladnost

In [50]:
```python
pd.crosstab(index=titanic["Survived"],
            columns=[titanic["Pclass"], titanic["Sex"]],
            margins=True)
```

Out[50]:

Pclass	1		2		3		All
Sex	female	male	female	male	female	male	
Survived							
0	3	77	6	91	72	300	549
1	91	45	70	17	72	47	342
All	94	122	76	108	144	347	891

In [51]:
```python
pd.crosstab(index=titanic["Survived"],
            columns=[titanic["Pclass"], titanic["Sex"], titanic["Embarked"]],
            margins=True)
```

Out[51]:

Pclass	1						2						3						All
Sex	female			male			female			male			female			male			
Embarked	C	Q	S	C	Q	S	C	Q	S	C	Q	S	C	Q	S	C	Q	S	
Survived																			
0	1	0	2	25	1	51	0	0	6	8	1	82	8	9	55	33	36	231	549
1	42	1	46	17	0	28	7	2	61	2	0	15	15	24	33	10	3	34	340
All	43	1	48	42	1	79	7	2	67	10	1	97	23	33	88	43	39	265	889

Stacked bar chart

```
In [52]: pd.crosstab(index=titanic["Pclass"], columns=titanic["Survived"]).plot.bar(stacked=True)
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb111abda0>
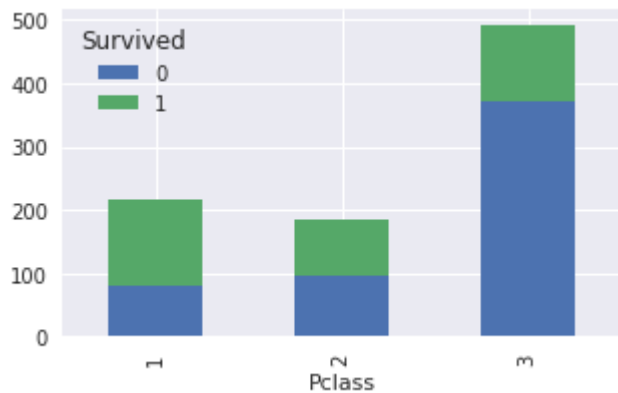
Chi-kvadrat (Chi-squared)

Tieto testy nie su zalozene na hodnotach atributov ako je to napriklad pri t-teste (kategoricka hodnota nema pre matematikov velky zmysel), ale na ich poctoch.

1. Chi-kvadrat test dobrej zhody (goodness-of-fit) Testuje, ci rozlozenie hodnot kategorickej premennej zodpoveda ocakavanemu rozdeleniu.
2. Chi-kvadrat test nezavislosti testuje, ci extistuje zavislost medzi dvoma kategorickymi premennymi

zdroj prikladov: http://hamelg.blogspot.sk/2015/11/python-for-data-analysis-part-25-chi.html (http://hamelg.blogspot.sk/2015/11/python-for-data-analysis-part-25-chi.html)

Chi-kvadrat test dobrej zhody

predstavte si, ze mame dve sady pozorovani a chceme urcit, ci su z rovnakeho rozdelenia.

Konkretny priklad: demograficke udaje pre cele USA a jeden stat v spojenych statoch

In [53]:
```python
national = pd.DataFrame(["white"]*100000 + ["hispanic"]*60000 +
                        ["black"]*50000 + ["asian"]*15000 + ["other"]*35000)


minnesota = pd.DataFrame(["white"]*600 + ["hispanic"]*300 +
                         ["black"]*250 +["asian"]*75 + ["other"]*150)
```

In [54]:
```python
national_table = pd.crosstab(index=national[0], columns="count")
national_table
```

Out[54]:

col_0	count
0	
asian	15000
black	50000
hispanic	60000
other	35000
white	100000

In [55]:
```python
minnesota_table = pd.crosstab(index=minnesota[0], columns="count")
minnesota_table
```

Out[55]:

col_0	count
0	
asian	75
black	250
hispanic	300
other	150
white	600

Vzorec na vypocet chi-kvadrat statistiky

$$\sum \frac{(observerd - expected)^2}{expected}$$

In [56]:
```python
observed = minnesota_table

national_ratios = national_table/len(national)   # pomery pre celu populaciu (referencne rozdelenie)
print('Ratios:', national_ratios)

expected = national_ratios * len(minnesota)    # ocakavane hodnoty ak by mala sledovana vzorka rovnake rozdelenie
print("Expected:", expected) # ak su z rovnakeho rozdelenia, tak toto by malo byt velmi podobne ako obsah premenr

chi_squared_stat = (((observed-expected)**2)/expected).sum()

print("Chi-squared", chi_squared_stat) # vysledna hodnota chi-kvadrat statistiky
```

```
Ratios: col_0      count
0
asian     0.057692
black     0.192308
hispanic  0.230769
other     0.134615
white     0.384615
Expected: col_0         count
0
asian       79.326923
black      264.423077
hispanic   317.307692
other      185.096154
white      528.846154
Chi-squared col_0
count     18.194805
dtype: float64
```

Nameranu statistiku musime porovnat s kritickou hodnotou

In [57]:
```
crit = stats.chi2.ppf(q = 0.95, # 95% confidence
                      df = 4)    # pocet stupnov volnosti testu = pocet kategorii - 1

print("Critical value:", crit) # kriticka hodnota by mala byt mensia ako hodnota chi-kvadrat statistiky, ktoru sm

p_value = 1 - stats.chi2.cdf(x=chi_squared_stat,  # aka je p-hodnota nasho testu
                            df=4)
print("P value:", p_value)
```

```
Critical value: 9.48772903678
P value: [ 0.00113047]
```

Kriticka hodnota je mensia ako namerana hodnota statistiky na nasej vzorke a zaroven p-hodnota je mensia ako 0.01 na zvolenej hranici istoty, takze mozeme povedat ze pozorovania su z rovnakeho rozdelenia

Pozor!

ak by p-hodnota nebola mensia ako 0.01, tak by sme nemohli povedat, ze mame statisticky dokaz toho, ze data su z rozneho rozdelenia. Mozeme povedat len to, ze nemame dostatok dokazov na to aby sme zamietli nulovu hypotezu a teda ze nevieme zamietnut hypotezu, ze su z rozneho rozdelenia.

In [58]:
```
# pre pohodlnost existuje pripravena funkcia, ktora to spocita za nas
# pozor na to, ze expected niesu namerane data pre celu populaciu ale ocakavane pocetnosti pri rovnakom pocte poz
# sledovana datova sada
stats.chisquare(f_obs= observed, f_exp=expected)
```

Out[58]: Power_divergenceResult(statistic=array([18.19480519]), pvalue=array([0.00113047]))

Chi-kvadrat test nezavislosti

Testujeme, ci existuje zavyslost medzi dvoma kategorickymi atributmi

In [59]:
```python
np.random.seed(10)

# Sample data randomly at fixed probabilities
voter_race = np.random.choice(a= ["asian","black","hispanic","other","white"],
                              p = [0.05, 0.15 ,0.25, 0.05, 0.5],
                              size=1000)

# Sample data randomly at fixed probabilities
voter_party = np.random.choice(a= ["democrat","independent","republican"],
                               p = [0.4, 0.2, 0.4],
                               size=1000)

voters = pd.DataFrame({"race":voter_race,
                       "party":voter_party})

voter_tab = pd.crosstab(voters.race, voters.party, margins = True)
voter_tab
```

Out[59]:

party race	democrat	independent	republican	All
asian	21	7	32	60
black	65	25	64	154
hispanic	107	50	94	251
other	15	8	15	38
white	189	96	212	497
All	397	186	417	1000

In [60]:
```python
observed = voter_tab.ix[0:5,0:3]
observed
```

Out[60]:

party race	democrat	independent	republican
asian	21	7	32
black	65	25	64
hispanic	107	50	94
other	15	8	15
white	189	96	212

In [61]:
```python
# Na zaklade sum po riadkoch a stlpcoch vyrobime ocakavane data
expected =  np.outer(voter_tab["All"][0:5],
                     voter_tab.ix["All"][0:3]) / 1000

expected = pd.DataFrame(expected)

expected.columns = ["democrat","independent","republican"]
expected.index = ["asian","black","hispanic","other","white"]

expected
```

Out[61]:

	democrat	independent	republican
asian	23.820	11.160	25.020
black	61.138	28.644	64.218
hispanic	99.647	46.686	104.667
other	15.086	7.068	15.846
white	197.309	92.442	207.249

toto vychadza z vety o sucine nezavyslich premennych:

Ak A a B su nezavysle premenne take ze P(A) > 0 a P(B) > 0, tak

$$P(A \cap B) = P(A) \times P(B)$$

expected obsahuje ocakavane pocetnosti v pripade ak by tato nezavyslost platila.

In [62]:
```
chi_squared_stat = (((observed-expected)**2)/expected).sum().sum()

print(chi_squared_stat)
```

7.16932128016

In [63]:
```
crit = stats.chi2.ppf(q = 0.95, # Find the critical value for 95% confidence*
                      df = 8)   # pocet stupnov volnosti je sucin poctu kategorii pre kazdu premenny - 1.
                                # Pocty su 3 a 5, teda 2 * 4 = 8

print("Critical value:", crit )

p_value = 1 - stats.chi2.cdf(x=chi_squared_stat,  # Find the p-value
                             df=8)
print("P value:", p_value)
```

Critical value: 15.5073130559
P value: 0.518479392949

In [64]:
```
# znova, existuje predpripravena funkcia, ktorej stacia pozorovane pocetnosti a vrati nam
# chi-kvadrat statistiku, p-hodnotu, pocet stupnov volnosti a ocakavane pocetnosti
stats.chi2_contingency(observed=observed)
```

Out[64]:
```
(7.1693212801620589,
 0.51847939294884204,
 8,
 array([[  23.82 ,   11.16 ,   25.02 ],
        [  61.138,   28.644,   64.218],
        [  99.647,   46.686,  104.667],
        [  15.086,    7.068,   15.846],
        [ 197.309,   92.442,  207.249]]))
```

Vysledok tohto testu je, ze namerana hodnota chi-kvadrat statistiky je mensia ako kriticka hodnota a nam sa nepodarilo doukazat zavislost medzi tymito dvoma premennymi.

Co dava celkom zmysel, kedze sme data generovali z dvoch uplne nezavislich nahodnych premennych.

Nepodaril osa nam teda vyvratit nulovu hypotezu a nemame dostatok dokazov na to, aby sme povedali, ze existuje zavyslost medzi tymito dvoma premennymi
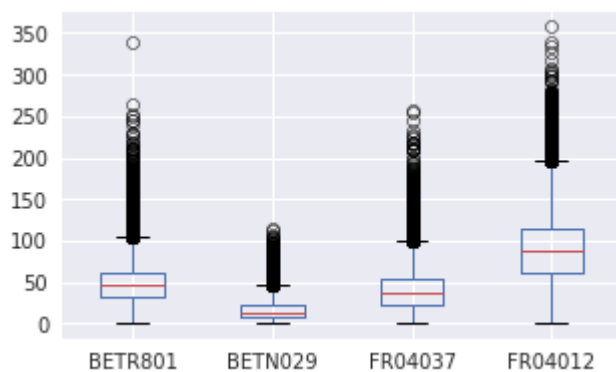
Spojity - Kategoricky

Tu sa najcastejsie pouziva rozdelovanie podla kategorickej hodnoty a zobrazovanie rozdeleni podmnozin numerickych hodnot napriklad pomocou histogramov alebo box-plotov.

Cize viacnasobne pouzitie vizualizacii, ktore sa pouzivaju na zobrazenie spojitych atributov

```
In [65]:  no2.plot(kind='box')
```

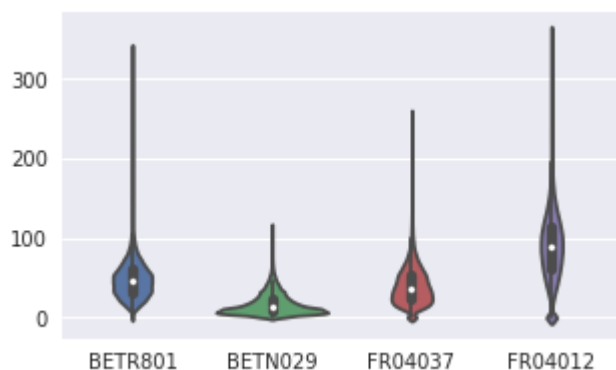Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbb11263ef0>

In [66]:   `seaborn.violinplot(no2)`

/usr/local/lib/python3.5/dist-packages/seaborn/categorical.py:2342: UserWarning: The violinplot API has been ch
anged. Attempting to adjust your arguments for the new API (which might not work). Please update your code. See
the version 0.6 release notes for more info.
  warnings.warn(msg, UserWarning)

Out[66]:   <matplotlib.axes._subplots.AxesSubplot at 0x7fbb111dac18>

Ak chceme overit, ci rozne podmnoziny maju rovnake/rozdielne vlastnosti, tak potrebujeme nejake statisticke testy.

Najcastejsie testy su tu t-test a Anova na overenie, ci jednotlive podmnoziny maju rozne priemery. Anova sa pouziva ak porovnavame viac ako dve podmnoziny.

T-test

Ak chceme porovnavat mnozinu s celou populaciou, tak potrebujeme jednovyberovy t-test

Ak chceme porovnavat dve nezavisle mnoziny, tak potrebujeme dvojvyberovy t-test.

Ak by sme chceli testovat rozdiely medzi dvoma vzorkami tej istej mnoziny, medzi ktorymi je napriklad casova zavyslost (zmena hodnoty po nejakom case/ukone (vysledky testu pred a po pouziti vyucbovej metody, vaha pred a po diete ...)), tak musime pouzit parovy t-test

Teraz chceme porovnavat dve mnzoiny, tak si dame ukazku Dvojvyberoveho a Paroveho t-testu

Dvojvyberovy t-test (Two sample t-test)

In [67]:
```python
np.random.seed(12)
ages1 = stats.poisson.rvs(loc=18, mu=33, size=30)
ages2 = stats.poisson.rvs(loc=18, mu=13, size=20)

print(ages1.mean(), ages2.mean())
```
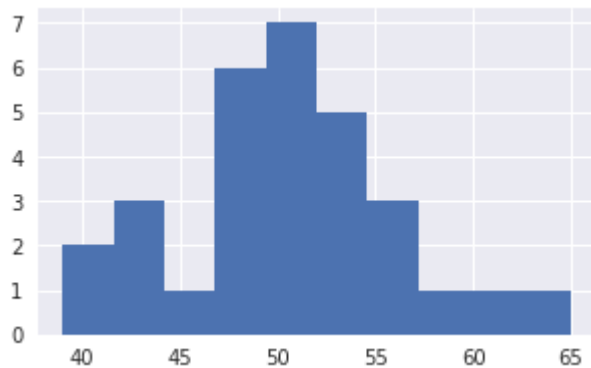
50.2666666667 31.6

In [68]:
```python
plt.hist(ages1)
```

Out[68]: (array([2., 3., 1., 6., 7., 5., 3., 1., 1., 1.]),
 array([39. , 41.6, 44.2, 46.8, 49.4, 52. , 54.6, 57.2, 59.8,
         62.4, 65.]),
 <a list of 10 Patch objects>)

In [69]:
```python
stats.ttest_ind(a= ages1,
                b= ages2,
                equal_var=False)
```

Out[69]: Ttest_indResult(statistic=14.275775319478337, pvalue=6.8197171100240744e-19)

p-hodnota je menej ako 0.01, takze mozeme povedat, ze sme nasli statisticky vyznamny rozdiel v priemernych hodnotach tychto dvoch vzoriek

Parovy t-test

```
In [70]: np.random.seed(11)
         before= stats.norm.rvs(scale=30, loc=250, size=100)
         after = before + stats.norm.rvs(scale=5, loc=-1.25, size=100) # Upravime povodnu vzorku.
                                                                       # Simulujeme tak nejaky proces, po ktorom chceme o
         weight_df = pd.DataFrame({"weight_before":before,
                                   "weight_after":after,
                                   "weight_change":after-before})

         weight_df.describe()
```

Out[70]:

|	weight_after	weight_before	weight_change
count	100.000000	100.000000	100.000000
mean	249.115171	250.345546	-1.230375
std	28.422183	28.132539	4.783696
min	165.913930	170.400443	-11.495286
25%	229.148236	230.421042	-4.046211
50%	251.134089	250.830805	-1.413463
75%	268.927258	270.637145	1.738673
max	316.720357	314.700233	9.759282

```
In [71]: stats.ttest_rel(a = before,
                          b = after)
```

Out[71]: Ttest_relResult(statistic=2.5720175998568284, pvalue=0.011596444318439857)

p-hodnota nieje mensia ako 0.01, takze mozeme povedat, ze sme nenasli dostatok dokazov na zamietnutie nulovej hypotezy a teda nevieme povedat, ci je tam rozdiel v strednych hodnotach

ANOVA

zdroj obrazku: https://xkcd.com/882/ (https://xkcd.com/882/)
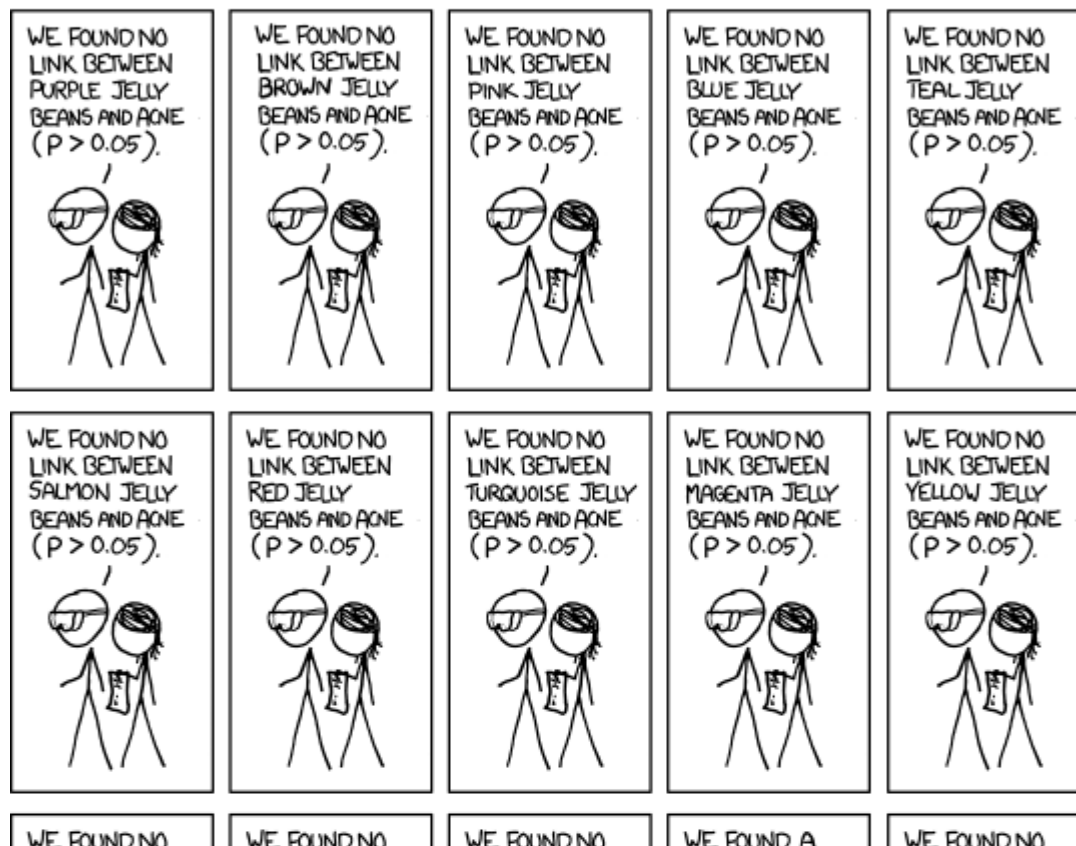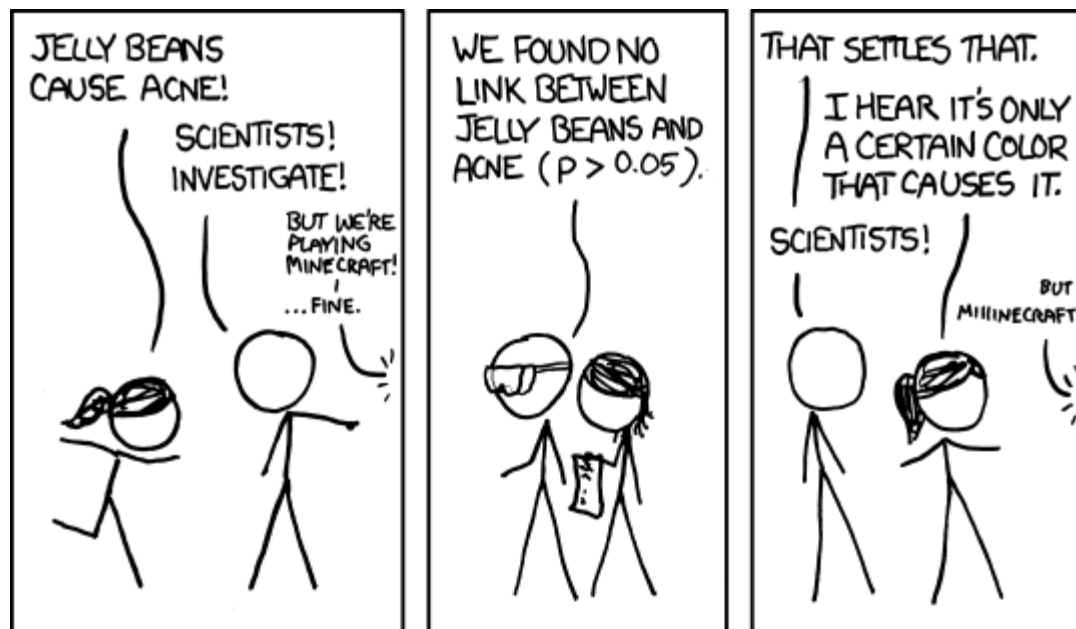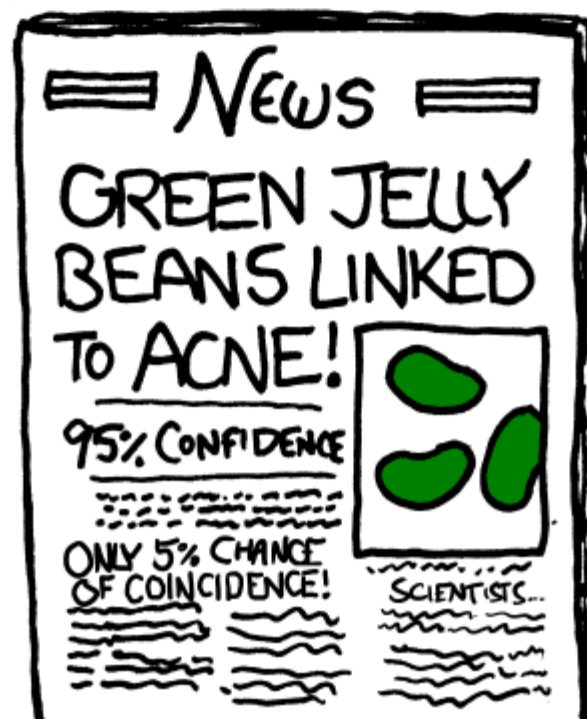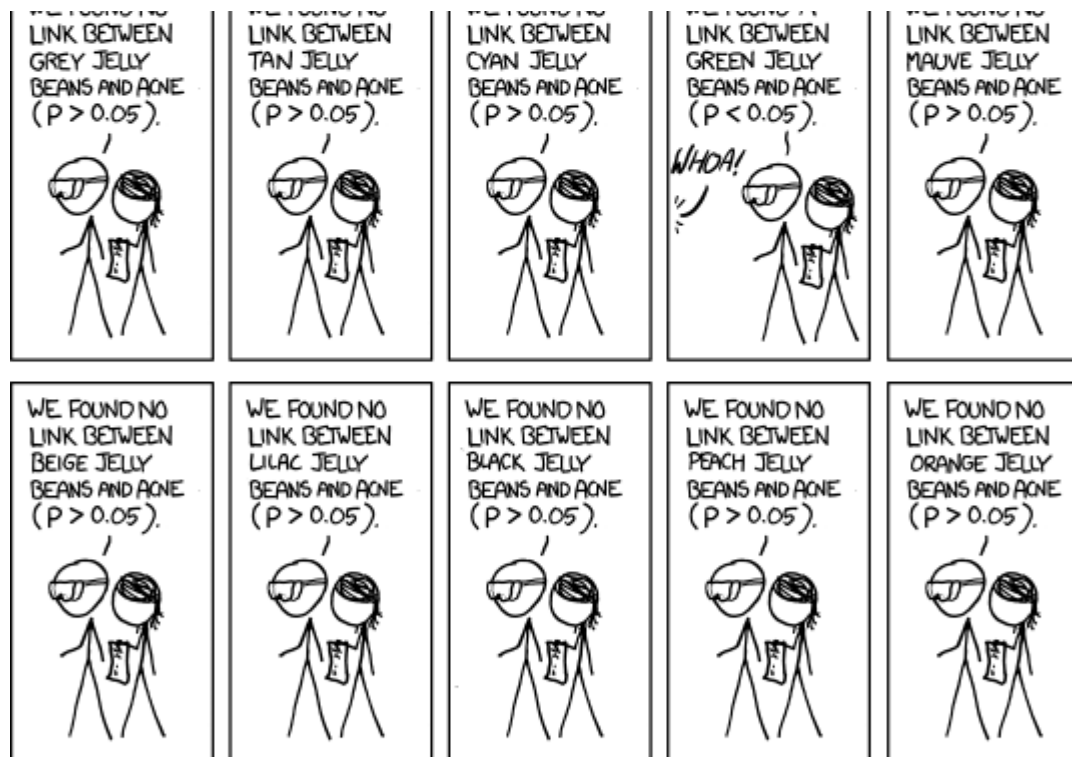
Na zistenie rozdielov medzi viacerymi skupinami by sme nemali robit opakovany t-test len tak. Hrozi nam, ze najdeme nevyznamny vysledok len kvoli nahode. P-hodnota znamena, ze s nejakou pravdepodobnostou dosiahnuty vysledok moze byt nahoda. Casta hranica je 5% alebo 1%. Ale aj toto je pravdepodobnost, ktora obcas nastane. Ak test opakujeme viac krat, tak sa nam to moze realne stat.

```
In [72]: np.random.seed(12)

races =    ["asian","black","hispanic","other","white"]

# Generate random data
voter_race = np.random.choice(a= races,
                              p = [0.05, 0.15 ,0.25, 0.05, 0.5],
                              size=1000)

# Use a different distribution for white ages
white_ages = stats.poisson.rvs(loc=18,
                               mu=32,
                               size=1000)

voter_age = stats.poisson.rvs(loc=18,
                              mu=30,
                              size=1000)

voter_age = np.where(voter_race=="white", white_ages, voter_age)

# Group age data by race
voter_frame = pd.DataFrame({"race":voter_race,"age":voter_age})
groups = voter_frame.groupby("race").groups

# Extract individual groups
asian = voter_age[groups["asian"]]
black = voter_age[groups["black"]]
hispanic = voter_age[groups["hispanic"]]
other = voter_age[groups["other"]]
white = voter_age[groups["white"]]

# Perform the ANOVA
stats.f_oneway(asian, black, hispanic, other, white)
```

Out[72]: F_onewayResult(statistic=10.164699828386366, pvalue=4.5613242113994585e-08)

Anova nam povedala, ze je tam vyznamny rozdiel medzi priemernym vekom medzi niektorymi skupinami. Nepovedala nam ale medzi ktorymi.

Jeden mozny sposob ako zistit medzi ktorymi je spustit t-test nad kazdou dvojicou.

Tu ale hrozi, ze oznacime nevyznamne rozdiely ako vyznamne (vid XKCD).

In [73]:
```python
race_pairs = []

for race1 in range(len(races)):
    for race2  in range(race1+1,5):
        race_pairs.append((races[race1], races[race2]))

for race1, race2 in race_pairs:
    print(race1, race2)
    print(stats.ttest_ind(voter_age[groups[race1]],
                           voter_age[groups[race2]]))
```

```
asian black
Ttest_indResult(statistic=0.83864469097479799, pvalue=0.4027281369339345)
asian hispanic
Ttest_indResult(statistic=-0.42594691924932293, pvalue=0.67046690042407264)
asian other
Ttest_indResult(statistic=0.97952847396359999, pvalue=0.32988775000951509)
asian white
Ttest_indResult(statistic=-2.3181088112522881, pvalue=0.020804701566400217)
black hispanic
Ttest_indResult(statistic=-1.9527839210712925, pvalue=0.051561971719525937)
black other
Ttest_indResult(statistic=0.28025754367057176, pvalue=0.77957701111176592)
black white
Ttest_indResult(statistic=-5.3793038812818352, pvalue=1.039421216662395e-07)
hispanic other
Ttest_indResult(statistic=1.5853626170340225, pvalue=0.11396630528484335)
hispanic white
Ttest_indResult(statistic=-3.5160312714115376, pvalue=0.00046412986490666839)
other white
Ttest_indResult(statistic=-3.7638093220778721, pvalue=0.00018490576317593065)
```

Aby sme zamedzili oznaceniu nevyznamnych rozdielov ako vyznamnych kvoli tomu, ze sme opakovali vela vyhodnoteni podmnozin, tak musime pouzit korekciu. Najcastejsie sa pouziva znizenie p-hodnoty. Ak hladame signifikanciu na urovni 5%, tak pri jednom teste musi byt p-hodnota < 0.05. Pri viacerych opakovaniach by sme mali tuto hodnotu vydelit poctom opakovani experientu. Cize nova hranica signifikancie je 0.05 / 10 = 0.005. Tato korekcia sa vola Bonferroniho korekcia.

Tato korekcia ale moze byt prilis konzervativna. Namiesto nej sa pouziva Tukeyho test.

```
In [74]:  from statsmodels.stats.multicomp import pairwise_tukeyhsd

          tukey = pairwise_tukeyhsd(endog=voter_age,
                                    groups=voter_race,
                                    alpha=0.05)

          tukey.plot_simultaneous()
          tukey.summary()
```
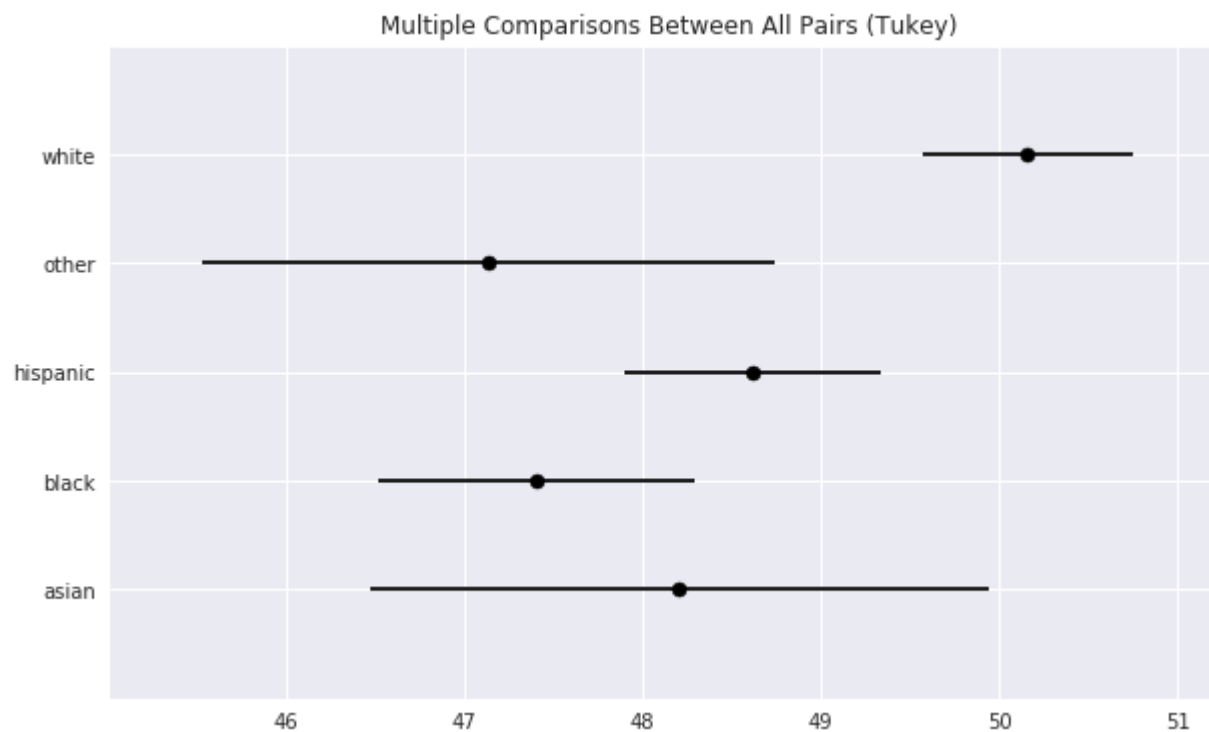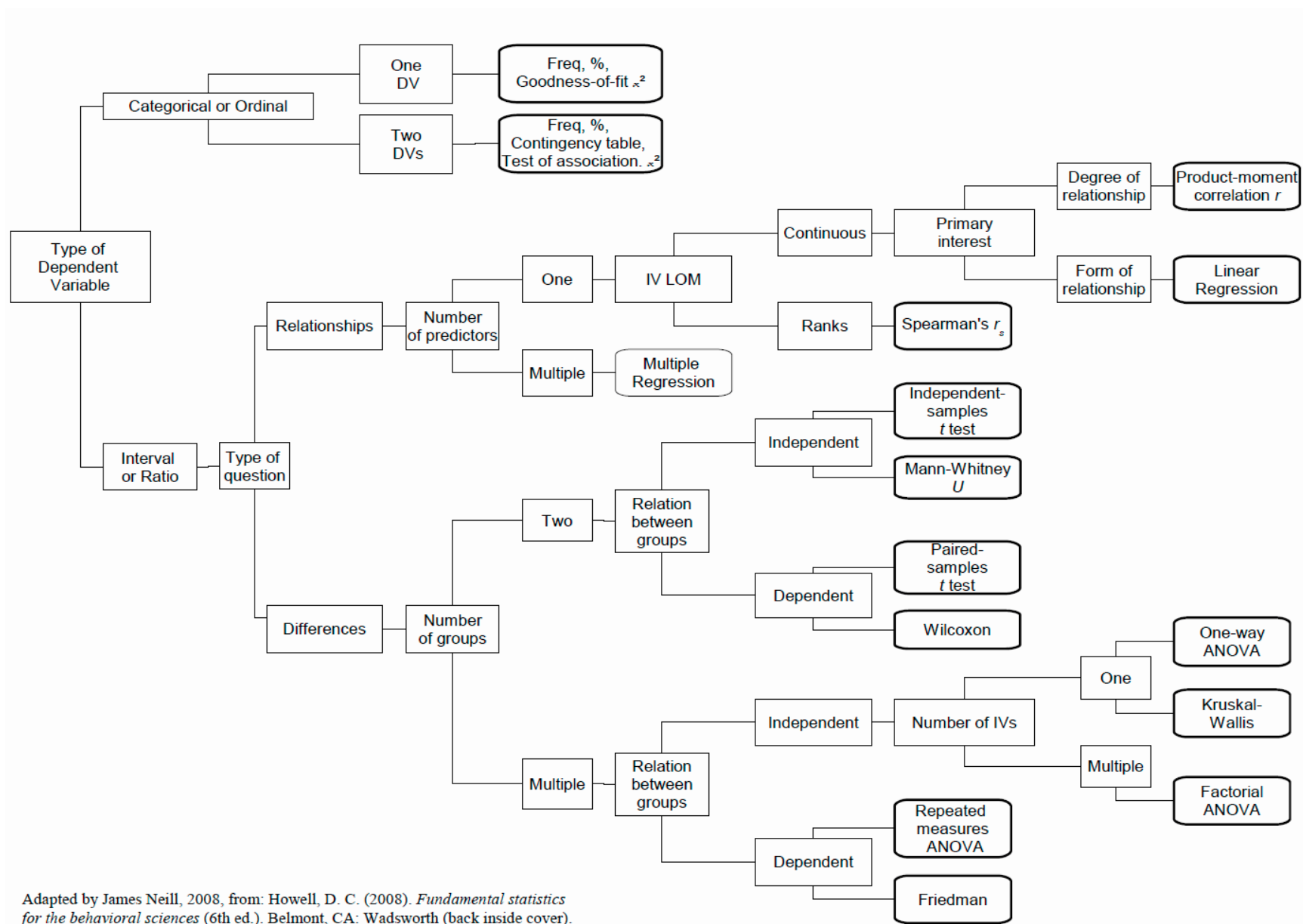
Out[74]:

Multiple Comparison of Means - Tukey HSD,FWER=0.05

group1	group2	meandiff	lower	upper	reject
asian	black	-0.8032	-3.4423	1.836	False
asian	hispanic	0.4143	-2.1011	2.9297	False
asian	other	-1.0645	-4.2391	2.11	False
asian	white	1.9547	-0.4575	4.3668	False
black	hispanic	1.2175	-0.386	2.821	False
black	other	-0.2614	-2.7757	2.253	False
black	white	2.7579	1.3217	4.194	True
hispanic	other	-1.4789	-3.863	0.9053	False
hispanic	white	1.5404	0.3468	2.734	True
other	white	3.0192	0.7443	5.2941	True

Multiple Comparisons Between All Pairs (Tukey)

Ked sa potrebujete rozhodnut aky test pouzit, tak moze pomoct takyto rozhodovaci strom

Adapted by James Neill, 2008, from: Howell, D. C. (2008). *Fundamental statistics for the behavioral sciences* (6th ed.). Belmont, CA: Wadsworth (back inside cover).

In [ ]: